ARCHES

Application, Refinement and Consolidation of HIC, Exploiting Standards

ESPRIT Project 20693

NOE SPECIFICATION

Version : 5.0

febuary 1999

Prepared By : Jean-Lou Desbarbieux

Approved By : Alain Greiner

Université Pierre et Marie Curie

NOÉ

SPECIFICATIONS

Draft 5.0 febuary 1999

Alain Greiner
Jean-Lou Desbarbieux Jean-Jacques Lecler Franck Wajsbürt
Cyril Spasevski Amal Zerrouki
Raphael Karabassis Jorge Negri

Content

CHAPTER 1
Introduction

1.1 Preamble

This document presents NOÉ, a component aimed to implement the interface between a PCI local bus and an HSL network (IEEE 1355). The objective behind NOÉ is to offer not only the necessary performance for data transfers but also a programmability ability to fit a wide range of message passing protocols.

This component is particularily adapted to build a cluster of workstations. It can be used on any CPU board offering a connection to a PCI bus.

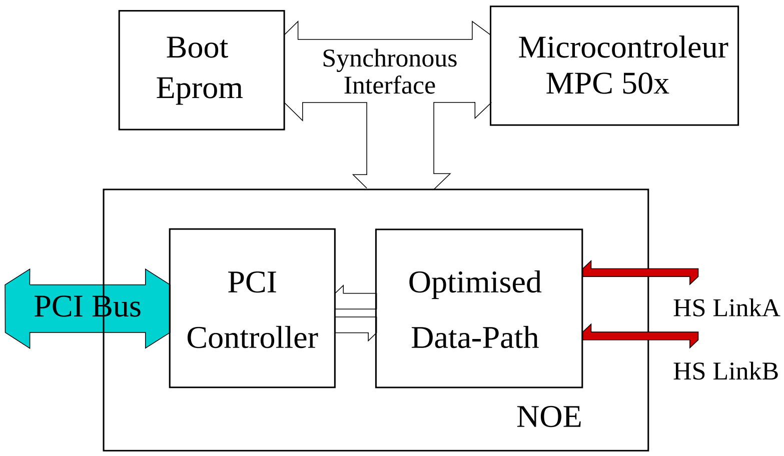


Figure 1.1: Overview of a NOÉ board

NOÉ is an initiator on the PCI bus, it reaches the data located in main memory without soliciting the host processor. It can also be a target on the PCI bus; the host processor performs memory writes to control the NOÉ component.

The host processor can read the NOÉ status registers through a read operation in main memory.

On the network side, NOÉ provides an interface with two HSL 1 Gbit/sec link.

1.2 Main characteristics

- Fully PCI 2.1 compatible (33 MHz, 64 bits).

- Fully IEEE 1355 compatible (2*HSL 1 Gbit/sec).
- A 2 Gbit/seconde peak throughput.
- A 6 Kbyte buffering space.

1.3 Applicable documents

IEEE 1355	Heterogeneous InterConnect (HIC) Standard	D1.4, Feb 1995
PCI Special Interest Group	PCI Specification	Rev 2.1, Jun 1995
UPMC-LIP6	RCUBE Specification	Ver 1.6, Feb 96
UPMC-LIP6	PCIDDC Specification	Ver 1.3a, Dec 96

CHAPTER 2
NOÉ overview

This chapter presents an overview of the internal architecture of the NOÉ component. This overview is aimed to help the reader identifying the data flows and how the local processor manages NOÉ resources. This chapter starts with the description of the internal blocks and busses of NOÉ. Then it shows, through an example implementing the PCIDDC protocol, how these elements work together.

2.1 Block Diagram

NOÉ is composed of 15 operators, two internal busses and 14 data fifos. It provides two independent bi-directionnal HSL ports and an interface with a 64-bit 33 MHz PCI bus.

Two datapathes can be identified in this architecture. A data path is associated to each HSL port. It contains a set of operators to handle HSL packets (e.g: PX_A, PB_A, HSL_MOD_A, ...) and a buffering resource made of fifos (e.g: DMAR2PB_A_FIFO, LP2PB_A_FIFO, ...).

Many operators are duplicated to garanty that the 2 pathes and thus the 2 ports are fully independent. However, given that the PCI bus interface is unique, the PCI-specific operators (DMAR, DMAW, R_BUFF, ...) should be shared by the 2 pathes.

2.2 Bus Description

NOÉ contains two internal busses : the Internal Interface Bus (I^2B) and the Local Processor Memory Bus (LPMB). This section will describe each of them.

2.2.1 Internal Interface Bus : I^2B

The I^2B bus connects the PCI controller (PCIC) to the different source and destination operators in NOÉ which are :

Sources .... DMAW, R_BUFF, SHREG and CONF.

Destinations ..... DMAR, W_BUFF, SHREG and CONF.

PCIC is the I^2B bus master. All operators connected to it are seen as slaves.

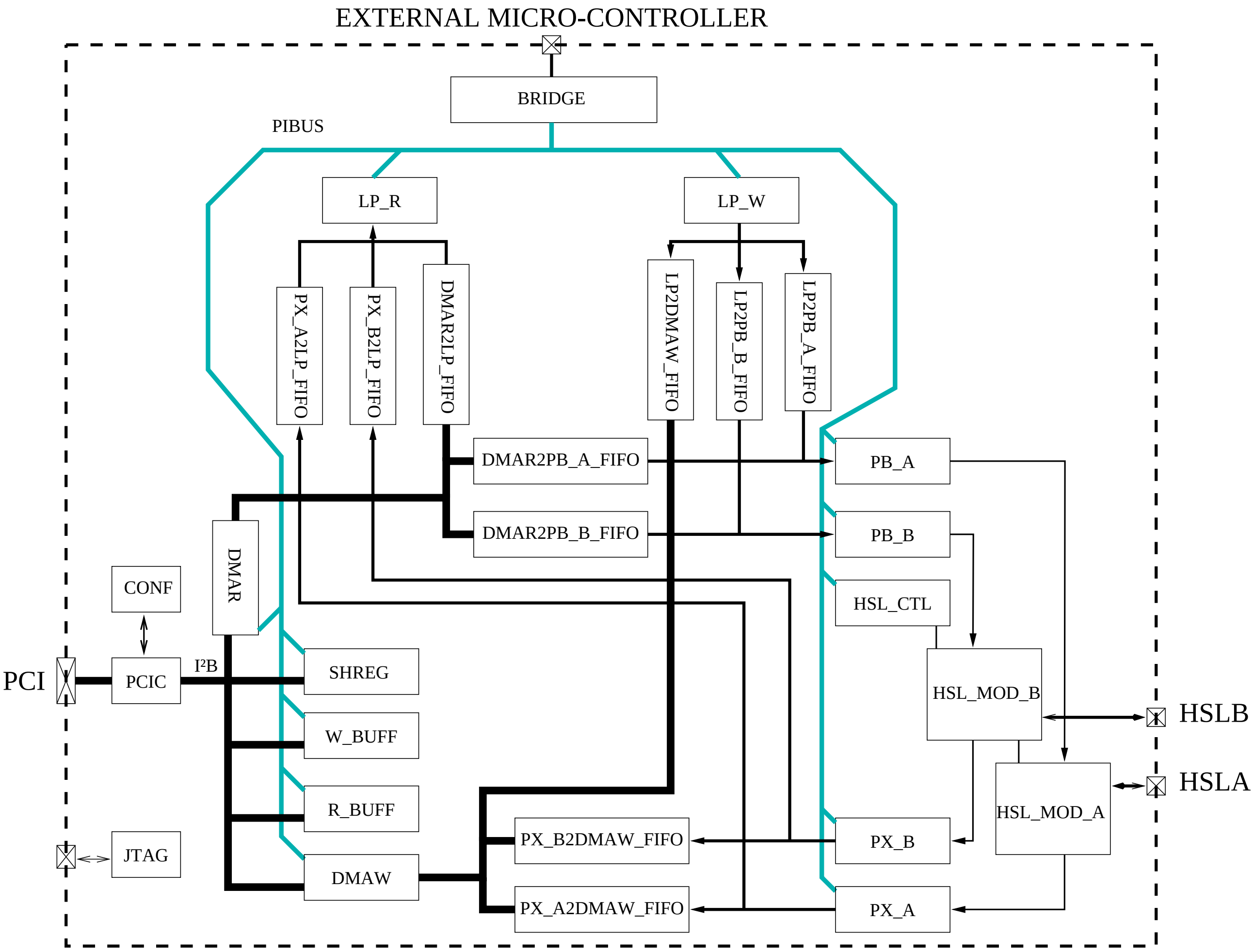


Figure 2.1: NOÉ block diagram

2.2.2 Local Processor Memory Bus : LPMB

This bus is a PI bus. It connects the different NOÉ operators to the Bridge which in turn connects NOÉ to the local processor memory bus. The different operators are seen on the bus at distinct addresses. The NOÉ status register address on the LPMB is known by all NOÉ operators. However, each operator is the exclusive owner of one or more bits in the NOÉ status register.

2.3 Block Description

2.3.1 PCI Controller : PCIC

This is the PCI 64 bits 33 MHz controller. PCIC is an initiator to perform memory read and write operations through the PCI bus.

It is also a PCI target that replies to both memory read or write requests and to configuration read or write operations.

Finally, it is also an initiator on the I^2B internal bus.

All address phases on the PCI bus are transferred on I^2B bus; PCIC does not handle target address decoding.

The CONF block decodes the address presented on the PCI bus to determine if it belongs to the NOÉ address space.

The R_BUFF indicates whether the requested address is currently available or if the local processor should be solicited to update the R_BUFF in order to satisfy the memory read request. If so, then, PCIC returns a RETRY indication to the initiator to delay the memory read ; the local processor will handle the memory read request when ready.

The DMAR and DMAW operators assert a set of signals to indicate to PCIC that memory read or write operations are to be performed. PCIC is also provided with a programmable arbiter that handles simultaneous memory read or write requests.

It is worth to mention that in the initiator mode, PCIC does not keep track of the addresses. As a consequence, when a DISCONNECT is requested by the target, DMAR and DMAW are responsible of keeping track of the addresses for which the DISCONNECT occured. They are also attributed the role of restarting later (resuming) the direct memory access operation.

2.3.2 Shared Registers : SHREG

The SHREG block consists of 4 32-bit generic registers. This block is used to ensure the communication between the host processor and the local processor. The registers can be read and written by both the local and the host processors.

These registers can be accessed rapidely by the two processors. But, given the number of registers provided (four), they cannot be used in a multi-users context.

These registers are necessary for the local processor boot.

The block contains also 2 32-bit interrupt registers. The first 32-bit register is used by the local processor to send interrupt requests to the host processor. The second one is used used by the host processor to send interrupt requests to the local processor.

2.3.3 Configuration Registers : CONF

This block contains the PCI configuration registers defined by the PCI 2.1 standard.

NOE does not contain any specific register. The circuit uses the SHREG registers during the initialisation phase (processor boot). Errors that might occure during normal operation are also reported using the SHREG registers.

	31 … 24	23 … 16	15 … 8	7 … 0
00 h	Device ID		Vendor ID	
04 h	Status		Command	
08 h	Base Class Code	Sub Class Code	Programming Interface	Revision ID
0C h	BIST	Header Type	Latency Timer	Cache Line Size
10 h 14 h	Base Address Register 0			
18 h 1C h	Base Address Register 1			
20 h 24 h	Unused Base Address Register			
28 h	Cardbus CIS Pointer			
2C h	Subsystem ID		Subsystem Vendor ID	
30 h	Expansion ROM Base Address			
34 h	Reserved			
38 h	Reserved			
3C h	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line

Figure 2.2: Type 00h configuration header space

NOÉ uses 2 base address registers of 64 bits wide each. The first one is used to access the SHREG registers and the second one is mainly used to control the local processor through memory accesses. These accesses are necessarily submitted via the PCI bus by the host processor and handled in NOÉ by the R_BUFF and W_BUFF operators which in turn will solicite the local processor. In some cases, the second base address register can also be used to perform read or write operations in the local memory.

The first base address register dedicated to SHREG can address a 4KByte array. This size is chosen to ease the host processor MMU operation. The SHREG block contains actually only few registers. However, each of them can have more than one image in the host memory.

The memory array that can be addressed using the second base address register must be programmable. It is always limited to 4GBytes.

In both cases, these memory arrays can be located anywhere in the 64-bit address space. The two memory arrays are prefetchable.

2.3.4 Direct Memory access Write : DMAW

The DMAW block manages DMA transfers from NOÉ internal fifos to the host memory. The DMA order is given to DMAW by the local processor. This order consists of a physical memory address, a data length in bytes and a source fifo. The orders are written in registers that will be described in the programming chapter (chapter 3).

Once the address low register is written, all registers are pushed in an order fifo. Such a behaviour allows the local processor to shorten an order whenever two successive orders have common parts (i.e. it is not necessary to indicate each time the upper memory address).

The DMAW block is a target on I^2B bus. PCIC has to detect the order and ask for the PCI-bus. Then, PCIC reads data from DMAW and tries to build a single PCI burst. If a DISCONNECT arises, the DMAW block maintains the host memory target address.

There are three source fifos, two from the HSL network, and the third from the local processor.

2.3.5 Direct Memory acces Read : DMAR

The DMAR operator is controlled by the local processor to perform memory read operations in the initiator mode. Whenever needed, DMAR indicates to PCIC that a memory read in the host memory is necessary. PCIC requests the PCI bus to start reading in the host memory. If a DISCONNECT arises, the DMAR block maintains the host memory target address.

When the DMA read operation is complete, the DMAR operator sets a specific bit in the NOÉ status register to indicate to the local processor that the data transfer is complete.

DMAR can push the data in three distinct fifos. The choice of the destination fifo depends on the use of the returned data. When the data returned is to be used by the local processor, it is pushed in DMAR2LP_FIFO. When used to feed one of the packet builders, it is pushed either in DMAR2PB_A_FIFO or DMAR2PB_B_FIFO depending of the target HSL port.

2.3.6 The Read/Write Buffer : R/W_BUFF

The read/write Buffer handles the PCI interface part that allows NOÉ to act as a target on the PCI bus. The PCIC operator transmits to CONF and Read/Write buffer the address made available on the PCI bus during the PCI address phase.

The CONF block decodes the address to determine whether the PCI address belongs to the addressing space attributed to NOÉ during the system boot.

If the address is recognised, then the Read/Write buffer indicates to PCIC if the corresponding data is available in the prefetch buffer. If yes, R_BUFF transfers the data to PCIC, else a RETRY notification is returned on the PCI bus to force the initiator to resend the request later.

The prefetch buffer is a 1-word line of 64 bits width. When a MISS happens, the memory access request is stored into the directory. This directory status is also set to REQUESTED to indicate that the request is currently in process.

The requested address is pushed once in the fifo with a special annotation to indicate that the address correponds to a read request. When ready, the local processor pops the read request from the fifo. It performs the local memory read, stores the requested data in the prefetch line of the Read/Write buffer, and declares the line valide to signal that the data is available.

NOÉ does not handle bursts. During a write operation if a MISS happens in the prefetch buffer, the address, the data and the byte enables are pushed in the fifo with an annotation to indicate that the request corresponds to a write operation. If a HIT is detected, the prefetch buffer line is invalidated and the request is pushed into the request fifo if the line has been already consumed (line status set to CONSUMED). If the request fifo is full or the line is valid but still not consumed yet, the request is rejected.

When ready, the local processor pops the write request from the fifo to perform the local memory write.

As a general rule, if ever the request fifo is full, then any new request causing a MISS in the Read/Write buffer will drive a RETRY notification on the PCI bus.

2.3.7 PCI Read Burst FIFO : DMAR2PB_A_FIFO, DMAR2PB_B_FIFO

These fifos are used to store the data read in the host memory in the intiator mode. Data is pushed in one of these fifos by DMAR and popped by the corresponding packet builder (PB_A

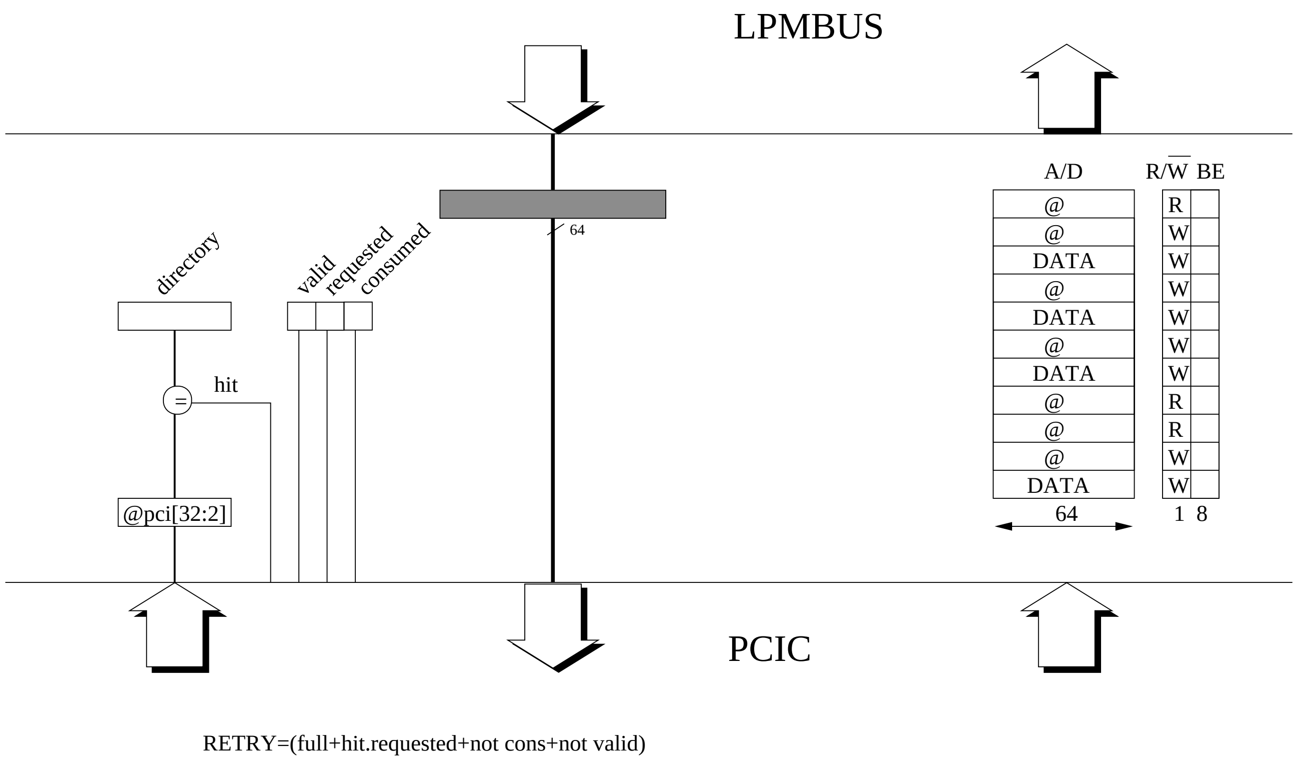


Figure 2.3: R/W_BUFF block

or PB_B).

Knowing that a write (push) in this fifo is faster than a read (pop), the fifo should be chosen large enough to satisfy the timing constraints introduced by PCI bursts. Therefore, both DMAR2PB_A_FIFO and DMAR2PB_B_FIFO contain 128 words of 64 bits each leading to a 2KByte overall size.

2.3.8 PCI Write Burst FIFO : PX_A2DMAW_FIFO, PX_B2DMAW_FIFO

These fifos are used to store the data extracted from the packets received on the HSL links. Data is pushed in each fifo by the corresponding packet extractor (PX_A, PX_B) and popped by the DMAR operator.

Given that a write (push) in these fifos is slower than a read (pop), the local processor can take advantage of the time interval in between to start interpreting the packet headers read on the HSL network.

To take advantage of the parallelism, this fifo must be large. Both PX_A2DMAW_FIFO and PX_B2DMAW_FIFO contain 192 words of 64 bits each (3KBytes).

2.3.9 PCI to Local Processor FIFO : DMAR2LP_FIFO

This fifo is used by the local processor to store the descriptors read in the host memory. The descriptor is pushed in this fifo by the DMAR operator and is popped by the local processor.

DMAR2LP_FIFO contains 32 words of 64 bits each.

2.3.10 Local Processor to PCI FIFO : LP2DMAW_FIFO

This fifo is used by the local processor to write data into the host memory. Typically, it will be used to write descriptors in a reception fifo to keep track of received messages (this mechanism is implemented by PCIDDC. For further details, refer to PCIDDC User manual).

LP2DMAW_FIFO contains 32 words of 64 bits each.

2.3.11 Packet Extractor to Local Processor FIFO : PX_A2LP_FIFO, PX_B2LP_FIFO

These fifos are used by their associated packet extractor (PX_A and PX_B) to write incoming message headers and packet trailers detected on the HSL network. These fifos are read by the local processor.

Both PX_A2LP_FIFO and PX_B2LP_FIFO contain 32 words of 32 bits each.

2.3.12 Local Processor to Packet Builder : LP2PB_A_FIFO, LP2PB_B_FIFO

The local processor writes in each of these fifos the words that will be sent on the HSL network. Typically, these fifos will be used to store the headers and the trailers that will be appended to data before a packet is sent on the HSL link.

Both LP2PB_A_FIFO and LP2PB_B_FIFO contain 32 words of 32 bits each.

2.3.13 Packet Builder : PB_A, PB_B

The packet builder is a NOE's operator which goal is to assembly data and control words to create a packet. These packets will be sent in the HSL network. As NOE presents two HSL ports, there are two packet builders (PB_A and PB_B), each one connected to one HSL port interface.

This operator has four main interfaces. The first one is a PI-BUS interface which allows communication with the local processor. The second one is a 9C's inteface (9 bits with Control) to the HSL port interface, for packet emission in the HSL network. The third one is an intrface to the PCI operator of NOE. This inteface connects the packet builder to a FIFO (DMAR2PB_A_FIFO and DMAR2PB_B_FIFO) that is the data source for buildin a packet. The fourth one is an interface to NOE's external signals, as *reset* and *loop-back*.

The packet builder is programmable. The local processor must post instructions in its instruction FIFO. The fact of writing in its FIFO starts a packet building operation.

2.3.14 Packet Extractor : PX_A, PX_B

The packet extractor is a NOE's operator which goal is to extract data from the packets detected on the HSL network. As NOE presents two bi-directional HSL ports, there are two independent packet-extractors (PX_A and PX_B), each one connected to one HSL port.

This operator presents five main interfaces. The first one is an interface with a PI-BUS. By this interface, the local processor can read the status words from the packet-extractor. The second one is an inteface with the NOE's PCI operator. This interface is connected to a FIFO (PX_X2DMAW_FIFO) that allows "data decorelation" between dta extraction (FIFO's writing by PX) and memory data writing(FIFO's reading by DMAW). The third one is a 9C's interface that connects the packet extractor to the HSL port interface operator. This interface is connected to a FIFO (HSL2PX_X_FIFO) that provides input data to the packet extractor. The fourth one is another 9C's interface that connects the packet extractor to the packet builder's *loop-back*. This interface is available for the *loop-back* working mode. The fifth one is a general interface to NOE's exernal signals, as reset and *loop-back*.

2.3.15 Packet Builder to HSL module FIFO : PB_A2HSL_FIFO, PB_B2HSL_FIFO

The packets built by the packet builders PB_A or PB_B are pushed into PB_A2HSL_FIFO or PB_B2HSL_FIFO depending of the target HSL port. The HSL module will then read the packets contained in this fifo to send them on the HSL serial link.

The main function of these fifos is to solve the frequency discripancy that exists between the Packet builders and the HSL modules.

Both PB_A2HSL_FIFO and PB_B2HSL_FIFO contain 32 words of 32 bits each.

2.3.16 HSL Module to Packet eXtractor FIFO : HSL2PX_A_FIFO, HSL2PX_B_FIFO

Each HSL module uses a fifo to store the packets detected on the the HSL link. Packets submitted by HSL_MOD_A are pushed into HSL2PX_A_FIFO to feed the packet extractor PX_A. Similarily, packets submitted by HSL_MOD_B are pushed into HSL2PX_B_FIFO to feed the packet extractor PX_B.

As for PB_A2HSL_FIFO and PB_B2HSL_FIFO, the main function of HSL2PX_A_FIFO and HSL2PX_B_FIFO is to solve the frequency discripancy that exists between the HSL modules and the packet extractors.

Both HSL2PX_A_FIFO and HSL2PX_B_FIFO contain 32 words of 32 bits each.

2.3.17 HSL Module Control : HSL_CTL

The HSL II module provides a maintenance port. The HSL_CTL block manages the communication between the local processor and the HSL modules through this port.

2.4 Typical Functionnal Description

This section introduces the packet transmission and reception principles to be implemented in NOÉ.

In this section, and for the sake of explanation, we assume that a message passing protocol based on remote writes is to be implemented using NOÉ. It is important to mention that NOÉ can support remote reads as well with no cost increase. Moreover, given that the protocol is handled by the firmware, NOÉ can satisfy the requirements of a wide range of protocols at variable costs.

The example presented in this section corresponds to the protocol implemented by hardware on PCIDDC, a PCI Direct Deposit Component (cf Sec. 1.3). This example requires one HSL port only. Port 0 will be used and port 1 will be ignored. As a consequence, all the operators dealing with Port 1 will be ignored in the 2 subsequent sections.

To send a message, the data is made available by the application in a contiguous and locked physical memory array. Then, the application describes the transfer (using a descriptor) in a specific table in user memory. The descriptor contains the address in the local memory of the data to be transferred, the data size in bytes, the remote node identifier as well as the remote node address. Then, the application indicates to PCIDDC that a new message has been added to the table of the messages to be sent.

The data transfer is then fully taken in charge by the circuit. The local and remote applications can be informed that the data has effectively left the local memory and/or that a new data has been placed in the destination node memory by asserting an interrupt.

2.4.1 Transmission

Here are the differents actions undertaken by the operators of NOÉ during the transmission of a message from one node to another connected through an HSL network :

- The program executed by the host processor performs a write operation in memory to signal a new descriptor.
- PCIC identifies itself as a target. If the W_BUFF fifo is not full, PCIC acknowledge the PCI transaction.
- The data as well as its address transferred during the PCI write operation are written in a fifo of the W_BUFF.
- Periodically, the local processor polls the NOÉ status register. As soon as a new write operation is pushed into the W_BUFF fifo, the NOÉ status register is updated so that the local processor can detect the arrival of a new request in the write buffer.
- The local processor reads then in the W_BUFF fifo the data and its address. This write operation is interpreted by the local processor as a write of a new descriptor.
- The local processor then requests from DMAR a memory read to acquire the new message descriptor stored in the host memory. The local processor indicates also that the returned descriptor should be pushed in DMAR2LP_FIFO.
- DMAR in turn, indicates to PCIC that data (the descriptor) should be read from the host memory and thus that a PCI transaction should be initiated.
- As soon as the PCI bus gets available, PCIC acquires the bus as an initiator to read the descriptor in the host memory. The received descriptor is then transferred to DMAR.
- DMAR pushes the descriptor into DMAR2LP_FIFO and sets a bit in the NOÉ status register to indicate to the local processor that the memory read is complete and done.
- By polling the NOÉ status register, the local processor detects that the descriptor is now available in DMAR2LP_FIFO. The processor reads then the descriptor in the fifo to identify the address of the data in the host memory and the size of the data to be transferred on the HSL network. The processor requests then from DMAR a memory read in the host memory. It indicates also to DMAR that this time, the data read in the host memory should be pushed into DMAR2PB_A_FIFO.
- DMAR in turn, indicates to PCIC that data (to be sent on the HSL network) should be read in the host memory and that a PCI read transcation should be initiated.
- As soon as the PCI bus gets available, PCIC acquires the bus as an initiator to read the effective data in the host memory. The received date is then transferred to DMAR.

- DMAR pushes the received data into DMAR2PB_A_FIFO and set a specific bit of the NOÉ status register to indicate that the memory read is complete.
- When the local processor detetcs in the NOÉ status register that the read is finished, it can assert an interrupt to the host processor by setting the appropriate bit in the SHREG block. The host processor will then be informed that the data has finally been sent ont the HSL network.
- In the mean time, the local processor continues decoding the remaining part of the descriptor to extract the information that will compose the message header. Then, the resulting header is pushed into LP2PB_A_FIFO.
- The local processor indicates to the Packet Builder (PB_A) the number of header bytes to pop from LP2PB_A_FIFO. It indicates also the number of data bytes to pop from DMAR2PB_A_FIFO. Finally, it allows the packet builder to send the packet on the HSL network.
- The packet builder composes the packet by popping both LP2PB_A_FIFO and DMAR2PB_A_FIFO and pushes the resulting packet in PB_A2HSL_FIFO.
- HSL_MOD_A reads the packet to be transmitted from PB_A2HSL_FIFO and finally sends it on the HSL link.

2.4.2 Reception

- HSL_MOD_A reads on the HSL link an incoming packet and pushes it into HSL2PX_A_FIFO.
- The packet eXtractor (PX_A) extracts the message header from HSL2PX_A_FIFO ans pushes it into PX_A2LP_FIFO.
- The packet extractor extracts also the data field from HSL2PX_A_FIFO and pushes it into PX_A2DMAW_FIFO.
- The local processor pops the header from PX_A2LP_FIFO once it detects in the NOÉ status register that a new packet has been extracted by PX_A and pushed into PX_A2LP_FIFO.
- The local processor extracts also the data size as well as the destination address at which the incoming data should be stored in the local memory.
- The local processor transmits then these information (data size and the destination address) to the DMAW block. It then authorises the DMAW to start reading the data.
- The DMAW block pops then the data from PX_A2DMAW_FIFO.
- DMAW indicates to PCIC that a memory read has to be performed.
- As soon as the PCI bus is available, PCIC extracts the data from DMAW and handles the PCI transaction a an initiator.
- As soon as the DMAW finishes popping the incoming data from PX_A2DMAW_FIFO, and PCIC finishes reading the data from DMAW, DMAW sets the appropriate bit in the NOÉ status register to indicate that the memory write is complete.
- The local processor can then assert an interrupt by setting the appropriate bit in the interrupt register provided in the SHREG block of NOÉ. This will generate an interrupt transmitted on the PCI bus to the host processor to signal that the data received on the HSL link has been finally stored in memory.

One should note that in order to minimise the latency during a transfer, it is recommended whenever applicable, to take advantage of the parallelism of NOÉ which offers the ability of controlling more than one operator simultaneously.

2.5 Pin Description

2.5.1 Pin grouping

NOÉ is a 208 pins QFP circuit of which 54 are dedicated for power supplies.

Group	Pins
PCI Bus	88
Reset from PCI	1
Microprocessor Port	49
Clock Management	3
HSL Port	4
Link Status LED Driver	3
JTAG and Probe Port	6
Power and Ground	54
Total	**208**

2.5.2 Signal Type definition

in . . . . Input is a standard input-only signal.

out . . . . Totem Pole Output is a standard active driver.

t/s . . . . Three-state is a bi-directional, three-state input/output pin.

s/t/s . . . . Sustained three-state is an active low three-state signal driven by only one bus agent at a time. When ownership is passed to another agent, the signal is driven high for one clock, and then undriven for an additional clock before being driven by the new owner.

o/d . . . . Open-drain signals allow multiple devices to share the pin as a wired-OR. A pull-up resistor should be used.

2.5.3 Reset from PCI

RESET# (in) . . . . . Schmitt-Triggered, Active Low. When asserted, RESET# asynchronously disables any internal or external tri-stated gated, in order to prevent any power-up damage. All PCI signals are tri-stated. RESET# also restores all internal memory points to their default state. This operation is done asynchronously to PCI clock. There is non timing constraint between RESET# and EXTAL : internal reset logic insures that reset exit state will be coherent. RESET# must be asserted during power-up.

2.5.4 Clock

NOÉ needs an internal high speed clock for HSL module (up to 80 MHz).

The 80 MHz clock signal is provided by an internal clock generator controlled by an external low-speed clock. This low-speed clock signal can be externally provided by a clock generator, or by a simple resonator exited by NOÉ.

In the first case the oscillator must be connected to the EXTAL pin, on the other hand the resonator must be connected between EXTAL and XTAL.

EXTAL (in) . . . . . . Clock Input. EXTAL can either be fed by external oscillator, or tied to a resonator.

XTAL (out) . . . . . . External Resonator Excitor. XTAL must be left floating if an external oscillator is used, or tied to the resonator.

CLKOUT (out) . . Internal Clock used by the HSL module.

2.5.5 HSL Ports

The two following pins represent the IEEE 1355/HSL port. Each pin (SDOUTA, SDOUTB, SDINA, SDINB) must be connected to another IEEE 1355/HSL port to build HS-Links. This connection can be made :-

- directly on PCB
- via a 2x50Ω coaxial cable (5 m max.)
- via an optical fiber, with external transcievers (70 m max.)

SDOUT (out) . . . . Serial Data Output. SDOUT pin includes power driver for a 5 meter long cable. Signal contains high frenquency components (up to 1 Gbit/s), pay attention at designing clean tracks.

SDIN (in) . . . . Serial Data Input. SDIN pin does not include a 50 Ω impedance adaptor. Two externals resistors (100 Ω to ground and 100 Ω to Vcc) are required to adapt the impedance line. Signal contains high frequency components (up to 1 Gbit/s), pay attention at designing clean tracks.

2.5.6 Link Status LED Driver

NOÉ integrates a link status LED driver, to help an user spotting link failures (bad connections, broken cables). The two pins can be connected via two opposite LEDs (a red one and a green one), with a resistor for current limiting. If both LEDs are integrated in the same package, it becomes a 4 state indicator:

- Shut : Link not enabled.
- Red : Link enabled, not calibrated.
- Green : Link enabled, calibrated, no data.
- Yellow : Link enabled, calibrated, with data.

CLED (out) . . . . . . Common LED Driver. This signal gives a low speed (about 1 kHz) clock.

CLED# (out) . . . . . Common LED Driver. This signal is the negation of the CLED signal.

LLED (out) . . . . . . Link LED Driver. This pin commands both green and red LED, wether the CLED signal is low or high respectively.

2.5.7 Microprocessor Port

From the microprocessor point of view, NOÉ behaves as a 4KByte, 32 bit wide SRAM. The Mapping of the address space is presented in chapter 3.

D[31:0] (t/s) . . . . . . Memory Data Bus. If CS# and RD# are asserted and WR# is desasserted, the Data Bus is an output reflecting data present at the address given by the address bus.

If CS# and WR# are asserted, the Data Bus is an input. Data will be latched at desassertion of WR#.

A[11:0] (in) . . . . . . Memory Address Bus. The address bus gives the byte address of the cell to be read or written.

CS# (in) . . . . Memory Chip Select. Active Low. CS# must be asserted to activate the configuration port.

OE# (in) . . . . Output Enable. Active Low. OE# assertion let the Data Bus to exit the high impedance state, and to export data specified by the address bus.

WE# (in) . . . . Write Enable. Active Low. WE# assertion let the Data Bus to enter the high impedance state. Data present on the Data Bus is latched at WE# desassertion. WE# have priority on OE#.

IRQ# (o/d) . . . . Interrupt Request. Active Low. NOÉ can generate interrupt request on a set of programmable events. In this case it asserts the IRQ# signal. The interrupt handler must then acknowledge the interrupt, which desasserts IRQ#.

CLK . . . . Local processor clock is use for NOÉ core (about 40 MHz).

2.5.8 JTAG and Probe Ports

This test port complies with the IEEE 1149.1 standard. It provides access for internal scan-path and boundary-scan logic.

TRST# (in) . . . . . . Test Reset. Resets the internal test logic.

TCK (in) . . . . Test Clock. Transaction on the JTAG port are synchrounous to this external provided clock.

TMS (in) . . . . Test Mode Select. Determines wether the device is functional or in test mode.

TDI (in) . . . . Test Data In. Serial Test Data Input.

TDO (out) . . . . Test Data Out. Serial Test Data Output.

PROBE (out) . . . . . Internal Signal Probe. This can be used to probe a set of internal signals, for chip test and debug.

2.5.9 Power and Ground

VCCS Serial Link Power. Must be tied to all other power pins externally. User must provide adequate decoupling capacitors.

GNDS Serial Link Ground. Must be tied to all other ground pins externally. User must provide adequate decoupling capacitors.

VCCE Pad Driving Power. Must be tied to all other power pins externally. User must provide adequate decoupling capacitors.

GNDE Pad Driving Ground. Must be tied to all other ground pins externally. User must provide adequate decoupling capacitors.

VCCI Internal Logic Power. Must be tied to all other power pins externally. User must provide adequate decoupling capacitors.

GNDI Internal Logic Ground. Must be tied to all other ground pins externally. User must provide adequate decoupling capacitors.

2.5.10 PCI Signals

CLK (in) . . . . is the CLOCK. This is the PCI bus clock and is the timing reference for all PCI bus transactions. The NOE chip can operate at a frequency up to 33 MHz.

RST# (in) . . . . stands for RESET. This signal is the PCI bus reset. It is one of the few PCI signals which may be asserted or negated asynchronously to the PCI bus clock (CLK).

AD [63:0] (t/s) . . . . stands for Address and Data. These signals represent the PCI bus address and data signals multiplexed on the same PCI pins. Information on these pins is identified as an address during the clock cycle in which the signal FRAME# is asserted. This is termed the "address phase" of a bus transaction. Information on these pins represents valid read or write data when both IRDY# and TRDY# are asserted, based on the current cycle type as defined on the C/BE lines during the address phase. This condition is termed the "data phase" of a bus transaction.

C/BE#[7:0] (t/s) . . . stands for Command and Byte Enables. These pins are used with the AD[63:0] pins. During the address phase of a bus operation, they identify the bus command to be performed. During the data phase of a bus operation, they identify the bytes involved.

PAR (t/s) . . . . stands for Parity. This PCI bus pin represents the even parity across the AD[31:0] and C/BE#[3:0] pins and is generated with a one clock delay.

PAR64 (t/s) . . . . . . Parity upper DWORD is the even parity bit that protects AD[63:32] and C/BE#[7:4].

FRAME# (s/t/s) . . stands for Cycle Frame. This PCI bus pin is asserted by the current bus master to signify the beginning of a bus transaction. Data transfers may continue in burst mode while FRAME# remains asserted. When FRAME# is de-asserted it indicates that the transaction is in the final data phase.

IRDY# (s/t/s) . . . . . stands for Initiator Ready. This signal is driven by the current bus master (initiator) and asserted to indicate its ability to complete the current data phase. Data is transferred when both IRDY# and TRDY# are asserted. Otherwise wait cycles will occur.

TRDY# (s/t/s) . . . . stands for Target Ready. This signal is driven by the selected bus target and asserted when that target is ready to complete the current data phase. Data is transferred when both IRDY# and TRDY# are asserted, otherwise wait cycles will occur.

STOP# (s/t/s) . . . . The STOP# signal is driven by the selected bus target and is asserted when it wishes to cease the current data transaction. IDSEL in Initialization Device Select This signal is used to gain access to the PCI configuration register space of a given PCI bus agent.

DEVSEL# (s/t/s) . stands for Device Select. The DEVSEL# signal is driven and asserted by the currently selected PCI bus target based on the current address and that target's assigned address range. Bus masters examine this signal to determine whether the desired device is present.

REQ# (t/s) . . . . stands for Request. This signal indicates to the bus arbiter that this device wishes to use the bus. It is a point-to- point signal that may be driven only when RST# is not asserted.

REQ64# (s/t/s) . . . stands for Request 64-bit Transfer. The bus master indicates it desires to transfer data using 64 bits. It has the same timing as FRAME#.

GNT# (t/s) . . . . stands for Grant. This point-to-point signal indicates that the bus has been granted to the requester. This signal may be driven only when RST# is not asserted and is ignored during the assertion of RST#.

ACK64# (s/t/s) . . . stands for Acknowledge 64-bit Transfer. It indicates the target is willing to transfer data using 64 bits. It has the same timing as DEVSEL#.

PERR# (s/t/s) . . . . stands for Parity Error. This signal indicates that a parity error has occurred. It is driven by the target or master that is the receiver of data during the clock cycle after the PAR signal becomes valid.

SERR# (o/d) . . . . . stands for System Error. This open-drain signal is driven by any device that detects odd parity during an address phase.

INTA# (o/d) . . . . . stands for Interrupt A. This signal is asserted when interrupt servicing of the NOE device is required. The INTA# pin is a shared PCI bus signal and uses an open-drain element to allow a wired-OR.

CHAPTER 3
NOÉ programming

This chapter describes how the NOÉ component can be programmed and used.

3.1 NOÉ command registers

This section describes the command registers necessary to control the various operators included in NOÉ. These registers can be accessed by the local processor through the onboard memory bus. NOÉ requires 4 KBytes in local processor address space. It can be located anywhere in the local processor memory address space. The local processor is in charge of generating the corresponding chip select.

Address	31 — 0
000h-00Ch	NOÉ Status and Control Register
100h-10Ch	Write Buffer Register
200h-20Ch	Read Buffer Register
300h-314h	Shared Register
400h-408h	DMAW Register
500h-508h	DMAR Register
600h	Packet Builder A Register
700h	Packet Builder B Register
800h	Packet eXtractor A Register
900h	Packet eXtractor B Register
A00h	PX_A2LP_FIFO Output Register
A04h	PX_B2LP_FIFO Output Register
A08h	DMAR2LP_FIFO Output Register
B00h	LP2PB_A_FIFO Input Register
B04h	LP2PB_B_FIFO Input Register
B08h	LP2DMAW_FIFO Input Register
E00h-E1Fh	HSL Control and Status Registers

Figure 3.1: NOÉ Adress Map

3.1.1 Register access modes

The registers are 32-bit wide but maybe split in fields. Each field has an independent access mode:

(R) . . . . Register may be read, and write a value does not matter.
(W) . . . . Register may be written and, a read value is undefined.
(R/W) . . . . Register may be read and written.
(RA) . . . . Register may be read. A read operation performed by the local processor in this register causes the selected operator to start an action.
(WA) . . . . Register may be written. A write operation performed by the local processor into this register causes the selected operator to start an action.

(R/C) . . . . Register may be read and cleared.
Write a 0 does not change the field, write a 1 clears it.

Access to an undefined register, read a **W** type register or write a **R** type register does not matter. In those cases, write has no effect and read returns back an undefined value.

The sequence in which WA and RA type registers are accessed is important. In some cases, accessing a given register may cause the change of other registers :

The local processor handles 32-bit words. Hence, it must actually perform 2 memory read operations to read a 64-bit word from a fifo. In such a case, the fifo pop will be activated if and only if the second read, returning the lower 32-bit word, has been performed.

3.1.2 NOÉ status register

This register is an aggregate of the status registers of the NOÉ operators. An operator status can be encoded or not, using 1 or more bits. The NOÉ status register cannot be read in the RA mode. Therefore, no action can be activated by reading this register.

Periodically, the local processor polls the NOÉ status register. If each bit is reset (equal to 0), none of the operators requested the local processor attention.

Here are the details of this register :

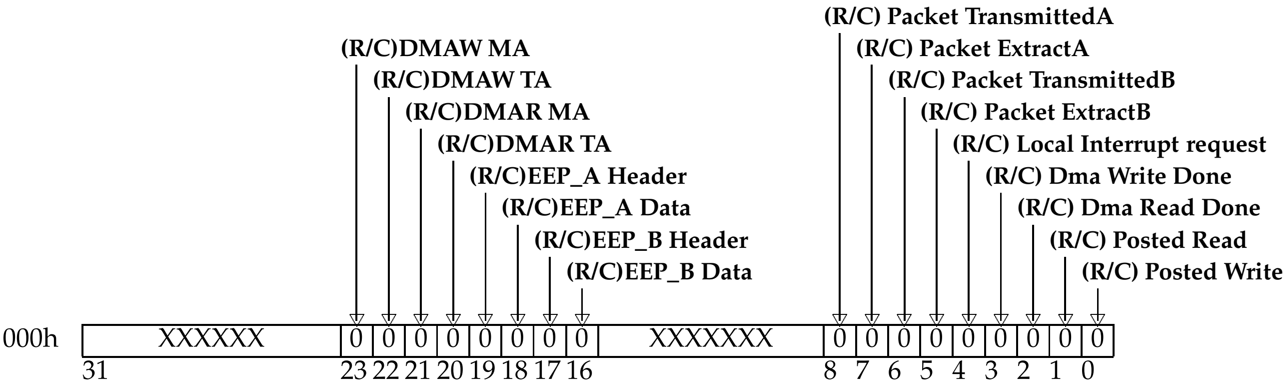


Figure 3.2: NOÉ Status Register

DMAW MA .... A master abort occur durring a DMAW operation.

DMAW TA .... A target abort occur durring a DMAW operation.

DMAR MA .... A master abort occur durring a DMAR operation.

DMAR TA .... A target abort occur durring a DMAR operation.

EEP_A Header .... PX detect an EEP ending a header section. PX is frozen until this bit is not clear by local processor.

EEP_A Data .... PX detect an EEP ending a data section. PX is frozen until this bit is not clear by local processor.

EEP Header .... PX_B detect an EEP ending a header section. PX is frozen until this bit is not clear by local processor.

EEP_B Data .... PX detect an EEP ending a data section. PX is frozen until this bit is not clear by local processor.

Packet TransmittedA . Each time the packet builder has built a packet and pushed it into PX_A2HSL_FIFO, a counter is incremented and Packet TransmittedA bit is set to indicate to the local processor that the packet has been sent to the HSL module.

Packet ExtractA ...... A packet header has been extracted by PX_A from HSL2PX_A_FIFO and has been pushed into PX_A2LP_FIFO. Reading the status register will acknowledge the extraction of an incoming packet header. If ever a second header is extracted from the network and pushed into PX_A2LP_FIFO, Packet ExtractA bit will remain set until the status register is read twice.

Packet TransmittedB . Each time the packet builder has built a packet and pushed it into PX_B2HSL_FIFO, a counter is incremented and Packet TransmittedB is set to indicate to the local processor that the packet has been sent to the HSL module.

Packet ExtractB .... A packet header has been extracted by PX_B from HSL2PX_B_FIFO and has been pushed into PX_B2LP_FIFO. Reading the status register will acknowledge the extraction of an incoming packet header. If ever a second header is extracted from the network and pushed into PX_B2LP_FIFO, Packet ExtractB will remain set until the status register is read twice.

Local Interrupt request This bit is set when the value of the interrupt request register in SHREG is different from zero.

Dma Write Done ..... DMAW is provided with a fifo that keeps track of the write orders given by the local processor. As soon as a write request has been satisfied, DMAW sets this bit. Reading the NOÉ status register will reset this bit if one DMA write order has been fullfiled. A counter is also provided to keep track of multiple DMA orders. The counter will be incremented each time a DMA order is fullfield and not acknowledged yet by the local processor. According to the counter value, reading the NOÉ status will reset this bit if all the orders satisfied have been acknowledged.

Dma Read Done ..... DMAR is provided with a fifo that keeps track of the read orders given by the local processor. As soon as a read request has been satisfied, DMAR sets this bit. Reading the NOÉ status register will reset this bit if one DMA read order has been fullfiled. A counter is also provided to keep track of multiple DMA orders. The counter will be incremented each time a DMA order is fullfield and not acknowledged yet by the local processor. According to the counter value, reading the NOÉ status will reset this bit if all the orders satisfied have been acknowledged.

Posted Read .... There is at least one memory read request pending in the R_BUFF fifo.

Posted Write .... There is at least one memory write request pending in the W_BUFF fifo.

3.1.3 NOÉ Command register

This register is used to generate interrupt to local processor.

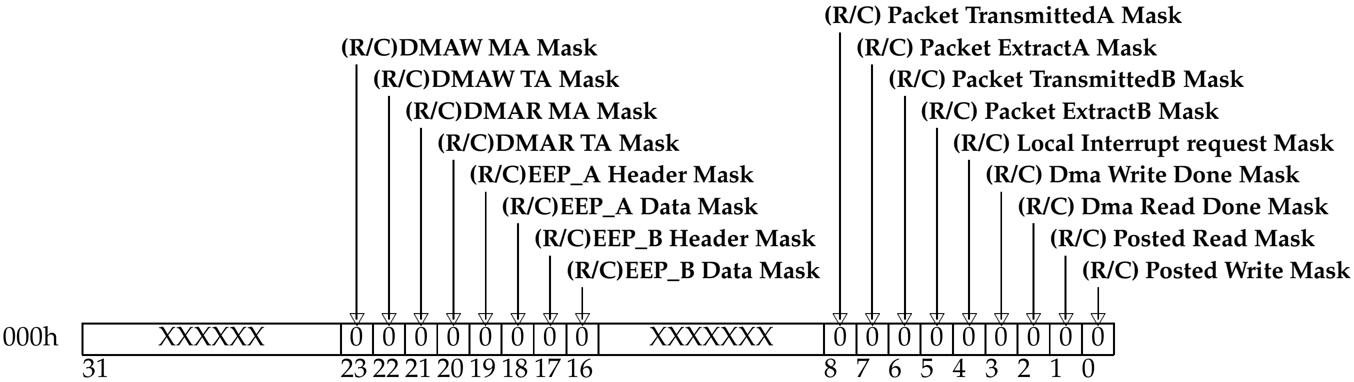


Figure 3.3: NOÉ Command Register

DMAW MA Mask .... When this bit set, NOÉ generates an interrupt to the local processor when DMAW MA in the status is set.

DMAW TA Mask ..... When this bit set, NOÉ generates an interrupt to the local processor when DMAW TA in the status is set.

DMAR MA Mask . . . . When this bit set, NOÉ generates an interrupt to the local processor when DMAR MA in the status is set.

DMAR TA Mask . . . . . When this bit set, NOÉ generates an interrupt to the local processor when DMAR TA in the status is set.

EEP_A Header Mask . When this bit set, NOÉ generates an interrupt to the local processor when EEP header in the status is set.

EEP_A Data Mask . . . . When this bit set, NOÉ generates an interrupt to the local processor when EEP data in the status is set.

EEP_B Header Mask . When this bit set, NOÉ generates an interrupt to the local processor when EEP header in the status is set.

EEP_B Data Mask . . . . When this bit set, NOÉ generates an interrupt to the local processor when EEP data in the status is set.

Packet TransmittedA Mask When this bit set, NOÉ generates an interrupt to the local processor when Packet TransmittedA in the status register is set.

Packet ExtractA Mask When this bit set, NOÉ generates an interrupt to the local processor when Packet ExtractA in the status register is set.

Packet TransmittedB Mask When this bit set, NOÉ generates an interrupt to the local processor when Packet TransmittedB in the status register is set.

Packet ExtractB Mask When this bit set, NOÉ generates an interrupt to the local processor when Packet ExtractB in the status register is set.

Local Interrupt request Mask When this bit set, NOÉ generates an interrupt to the local processor when Local Interrupt request in the status register is set.

Dma Write Done Mask When this bit set, NOÉ generates an interrupt to the local processor when Dma Write Done in the status register is set.

Dma Read Done Mask When this bit set, NOÉ generates an interrupt to the local processor when Dma Read Done in the status register is set.

Posted Read Mask . . . When this bit set, NOÉ generates an interrupt to the local processor when Posted Read in the status register is set.

Posted Write Mask . . . When this bit set, NOÉ generates an interrupt to the local processor when Posted Write in the status register is set.

3.1.4 NOÉ Data Fifo Level register

Before pushing any data into a fifo, the local processor must check that the fifo is not full. Similarily, before popping a data from any fifo, the local processor should check that the fifo is not empty.

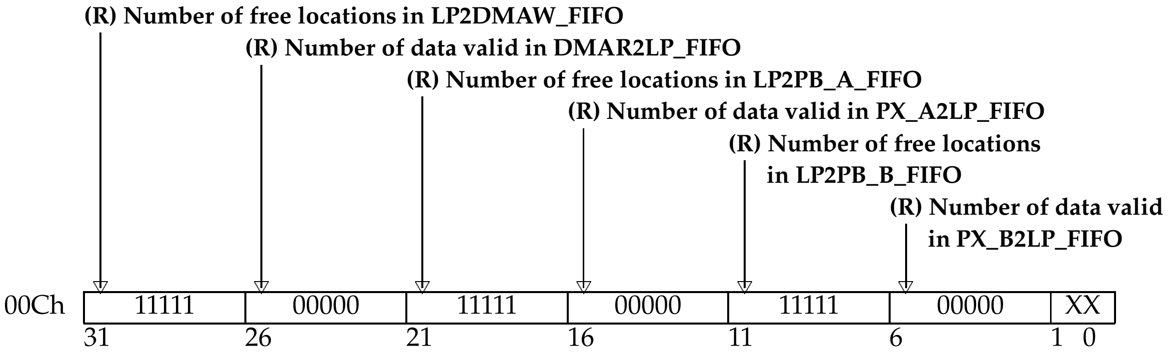


Figure 3.4: NOÉ Data Fifo Level Register

Number of free locations in LP2DMAW_FIFO
This register gives the number of free locations in LP2PB_FIFO.

Number of data valid in DMAR2LP_FIFO
This register gives the number of words that are currently pushed into DMAR2LP_FIFO.

Number of free locations in LP2PB_A_FIFO
This register gives the number of free locations in LP2PB_A_FIFO.

Number of data valid in PX_A2LP_FIFO
This register gives the number of words that are currently pushed into PX_A2LP_FIFO.

Number of free locations in LP2PB_B_FIFO
This register gives the number of free locations in LP2PB_B_FIFO.

Number of data valid in PX_B2LP_FIFO
This register gives the number of words that are currently pushed into PX_B2LP_FIFO.

3.1.5 NOÉ Command Fifo Level register

Before pushing any command into a fifo, the local processor must check that the fifo is not full. This register allows the local processor to push multiples command simultaneously.

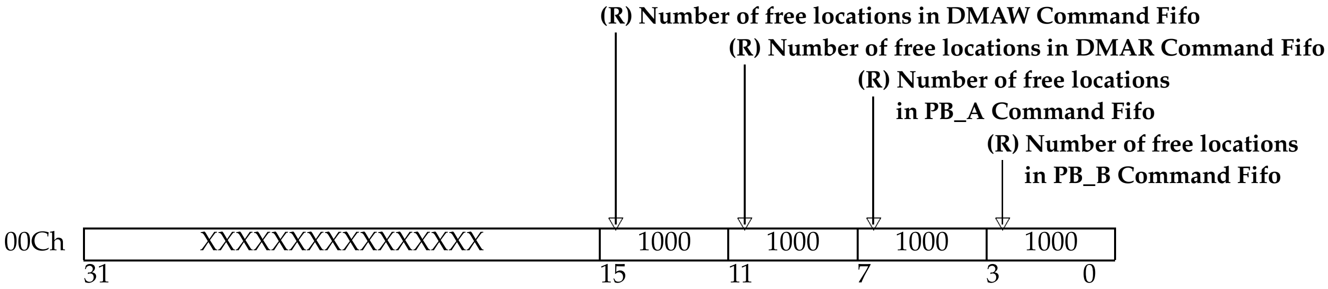


Figure 3.5: NOÉ Data Fifo Level Register

Number of free locations in DMAW Command Fifo
This register gives the number of free locations in DMAW Command Fifo. One location corresponds to a single command.

Number of free locations in DMAR Command Fifo
This register gives the number of free locations in DMAR Command Fifo. One location corresponds to a single command.

Number of free locations in PB_A Command Fifo
This register gives the number of free locations in PB_A Command Fifo. One location corresponds to a single command.

Number of free locations in PB_B Command Fifo
This register gives the number of free locations in PB_B Command Fifo. One location corresponds to a single command.

3.1.6 The Write Buffer

The write buffer is used to submit PCI write requests to the local processor. This latter needs the data and the address (in the PCI address space) at which it should be written.

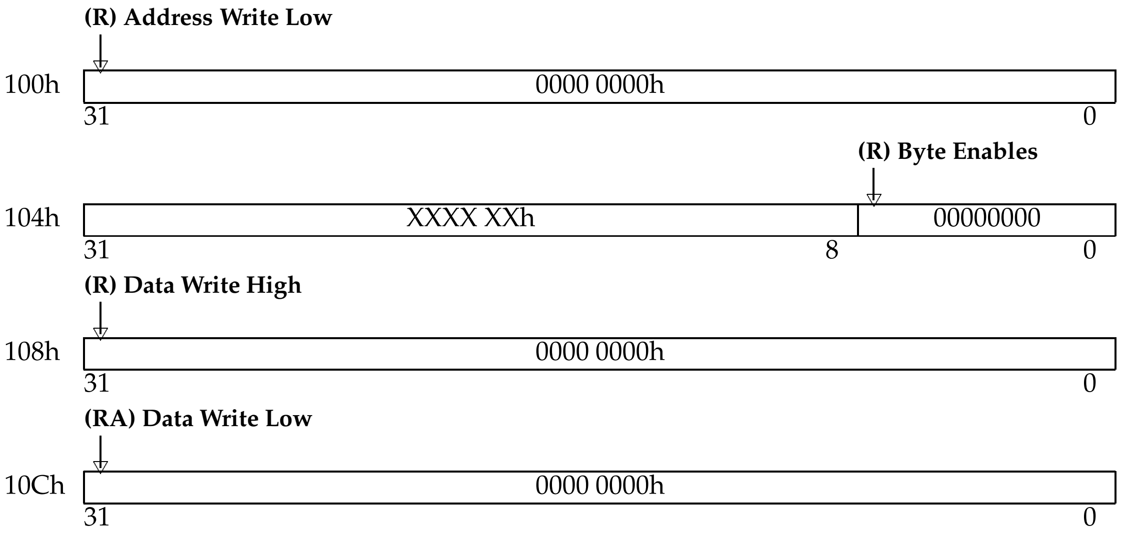


Figure 3.6: Write Buffer Registers

Address Write Low . . . These are the 4 lower address bytes presented on the PCI bus.

Byte Enables . . . . These 8 bits are used to identify the valid bytes of the data field. Each bit is active low.

Data Write High . . . . . . These are the 4 upper data bytes presented on the PCI bus.

Data Write Low . . . . . . These are the 4 lower data bytes presented on the PCI bus.

3.1.7 The Read Buffer

The read buffer is used to satisfy PCI read operations in the target mode. The prefetch buffer is managed by the local procesor.

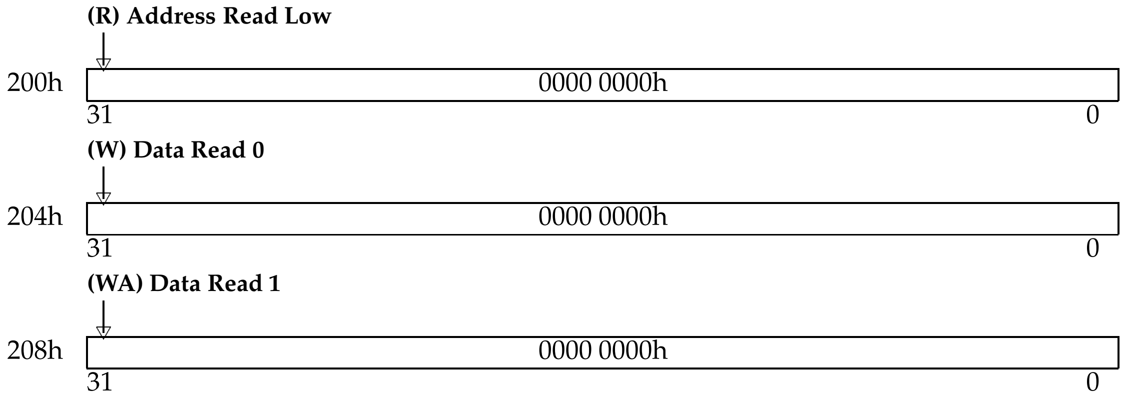


Figure 3.7: Read Buffer Registers

Address Read Low . . . These are the 4 lower address bytes presented on the PCI bus.

Data Read [1:0] . . . . This is the data returned by the local processor in reply to the read request posted on the PCI bus by the host processor.

3.1.8 Shared registers

This block is composed of 4 generic 32-bit registers used to support the exchange of information between the local processor and the host processor. These registers can be updated by the local processor. They can also be read and written by the host processor.

This block contains also 2 32-bit interrupt registers. The first register is used by the local processor to generate interrupt requests to be transmitted on the PCI bus. Similarily, the second 32-bit interrupt register is used by the host processor to submit interrupt requests to the local processor. Setting one or more bits of these registers will indicate that the interrupt signal is active.

These two interrupt registers can be read and written by both the local and the host processor.

Offset	Register	Value (bits 31–0)
300h	**(R/W) Register 0**	0000 0000h
304h	**(R/W) Register 1**	0000 0000h
308h	**(R/W) Register 2**	0000 0000h
30Ch	**(R/W) Register 3**	0000 0000h
310h	**(R/WA) Local IT**	0000 0000h
314h	**(R/WA) PCI IT**	0000 0000h

Figure 3.8: SHREG Registers

k

Register 0 . . . . In case of simultaneous write requests, the local processor access request is ignored.

Register 1 . . . . In case of simultaneous write requests, the local processor access request is ignored.

Register 2 . . . . In case of simultaneous write requests, the local processor access request is ignored.

Register 3 . . . . In case of simultaneous write requests, the local processor access request is ignored.

Local IT . . . . Setting one of the bits of this register generates on the local bus an interrupt request for the local processor.

PCI IT . . . . Setting one of the bits of this register generates on the PCI bus an interrupt request for the host processor.

3.1.9 The DMA Write

The DMAW operator allows NOÉ to act as an initiator on the PCI bus to write into the host memory a data flow coming from the network. This operator needs the destination address and eventually the data size expressed in bytes. If the specified size is 0, then the whole data flow will be written in memory. The specified address is then the base address and the following bytes will be written at subsequent memory locations.

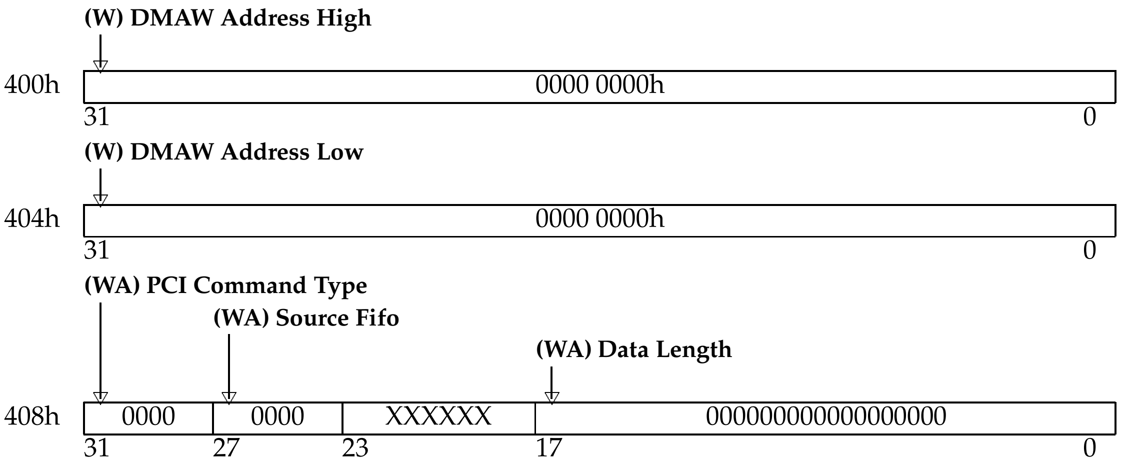


Figure 3.9: DMA Write Registers

DMAW Address High These are the 4 upper address bytes of the destination address (in the host memory) at which data should be written.

DMAW Address Low These are the 4 lower address bytes of the destination address (in the host memory) at which data should be written.

Data Length . . . . This field expresses the data size in bytes. It cannot exceed 64 KBytes.

PCI Command Type . . This field specify the PCI Command Type :

0011 I/O Write
1011 Configuration Write
0111 Memory Write
1111 Memory Write and Invalidate

Source Fifo . . . . This field indicates the data flow source which can be ;

0000 LP2DMAW_FIFO
0001 PX_A2DMAW_FIFO
0010 PX_B2DMAW_FIFO

Data Length . . . . This field expresses the data size in bytes. It cannot exceed 256 KBytes.

3.1.10 The DMA Read

The DMAR operator allows NOÉ to act as an initiator on the PCI bus to read data located in the host memory. This operator needs the address of the first byte to be read in the host memory as well as the expected data size expressed in bytes. The data size must be different from zero. Given that DMAR can feed three distinct fifos in NOÉ ((DMAR2LP_FIFO or DMAR2PB_A_FIFO, DMAR2PB_A_FIFO), the destination fifo must also be indicated.

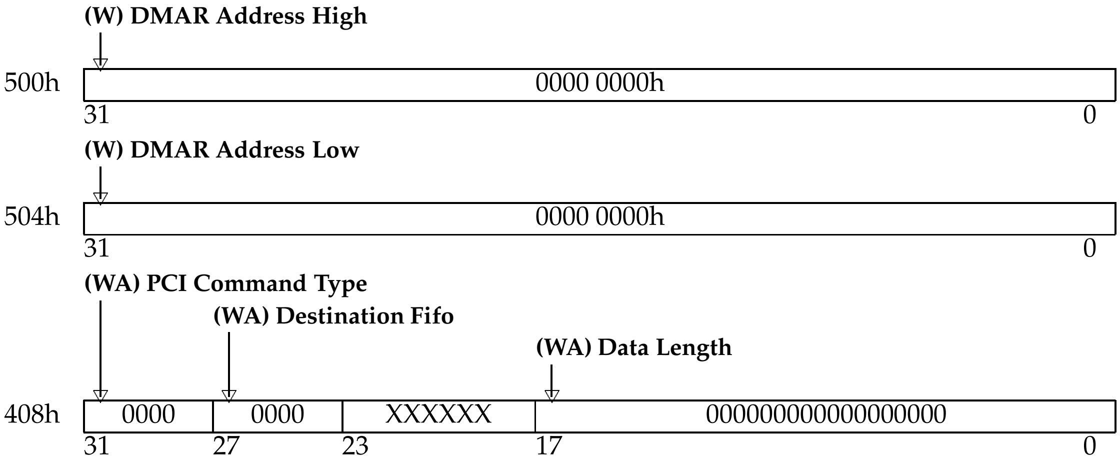


Figure 3.10: DMA Read Registers

DMAR Address High These are the 4 upper address bytes of the data in the host memory.

DMAR Address Low . These are the 4 lower address bytes of the data in the host memory.

Destination Fifo . . . . . . This field selects the destination fifo in which the incoming data will be pushed :

0000 DMAR2LP_FIFO
0001 DMAR2PB_A_FIFO
0010 DMAR2PB_B_FIFO

PCI Command Type . . This field specifies the PCI Command Type :

0010 I/O Read
1010 Configuration Read
0110 Memory Read
1100 Memory Read Multiple
1110 Memory Read Line

Data Length . . . . This field expresses the data size in bytes. It cannot exceed (256 Kbytes).

3.1.11 The Packet Builder

To build a packet the packet builder needs some information from the local processor, as the data length and the data source (DMAR2PB_X_FIFO or LP2PB_X_FIFO). Each packet builder has an instruction FIFO, where these informations are available. The local processor can write a packet builder instruction at the addresses 0x600 for the packet builder A and 0x700 for the packet builder B. The packet builder instruction has the following structure.

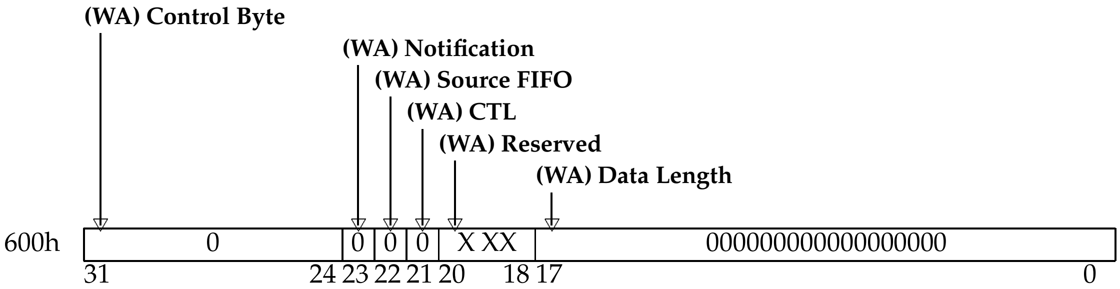


Figure 3.11: PB Instruction Registers

Control Byte . . . . This field has the control word that must be sent if CTL's value is 1.

Notification . . . . Indicates if the local processor must be notified when the packet is sent.

Source FIFO. . . . . Indicates the data source FIFO :

- **0** . . . LP2PB_X_FIFO.
- **1** . . . DMAR2PB_X_FIFO.

CTL . . . . Indicates if a control word must be inserted into the data path.

- **EP** . . . . End of Packet(0x2b).
- **EEP** . . . . Exceptional End of Packet(0x2d).
- **ES** . . . . End of Section(0x47).
- **CRCWR** . . . . . Write CRC(0x33).
- **CRCRST** . . . . . Reset CRC(0x35).

Data Length . . . . Length, in bytes, of the data that must be read in the source FIFO (DMAR2PB_X_FIFO or LP2PB_X_FIFO). If this field's value is 0 and the bit CTL is 1, it means that there's no data to be read in the source FIFO, but the packet builder must introduce a control word, as ES, EP or EEP into the data path.

The packet builder status can be read by the local processor at the same addresses (0x600 for the packet builder A and 0x700 for the packet builder B). The packet builder's status register has the following structure.

		(R) Inst_FIFO_Capa		(R) Mic_Ins_Reg
600h	X	0000	XXXX	0000
	31 … 12	11 … 8	7 … 4	3 … 0

Figure 3.12: PB Status Registers

Inst_FIFO_Capa. . . . . . Counter that shows the number of executed micro-instructions that asked for notification (Notification bit is 1 in the instruction register). This counter is decremented each time the local processor writes a 1 in the corresponding bit at the global status register (Packet TransmitedA or Packet TransmitedB).

Misc_Ins_Reg. . . . . Counter that shows the instruction fifo's free capacity. It should be read by the local processor before posting an instruction to the packet builder.

3.1.12 The Packet eXtractor

As NOE is fully programmable, the packet extractor must be able to process different kinds of packets. There are no informations exchanged between the packet extractor and the local processor about the communication protocol. The only kind of direct communication between these units is the packet extractor' status reading. The packet extractor status register is mapped at the address 0x800 for the packet extractor A and 0x900 for the packet extractor B. Writing at these addresses has no effect. The packet extractor' status register has the following structure .

	31 … 4	3 … 0
		(R) PX_COMPT
800h	XXXXXXXXXXXXXXXXXXXXXXXXXXXX	0000

Figure 3.13: PX Status Registers

PX_COMPT . . . . This field indicates the number of packet headers extracted from the HSL network, for whom a reception notification is required. This value is decremented each time the local processor writes a 1 in the corresponding bit at the global status register (Packet ExtractA or Packet ExtractB).

3.1.13 Header and descriptor fifos accesses

Four data fifos can be written or read by the local processor. These fifos can be accessed at different addresses of the local processor addressing space. If these fifos are not empty, a read operation of any fifo will induce a pop on the fifo. If these fifos are not full, a write operation in any fifo will result in a push in the fifo.

Before any fifo access, the local processor checks first the target fifo state. Reading an empty fifo will freeze the local processor until another operator starts filling the fifo. Writing in a full fifo will also freeze the local processor until a third operator starts popping the fifo.

A field of the NOÉ status register is dedicated to give a lower bound of the number of words contained in the fifo. Similarily, a specific field of the NOÉ status register is dedicated to give a lower bound of the number of free locations in the fifo.

Therefore, checking these bounds in the NOÉ status register before any fifo access is a way of avoiding the local processor freezing.

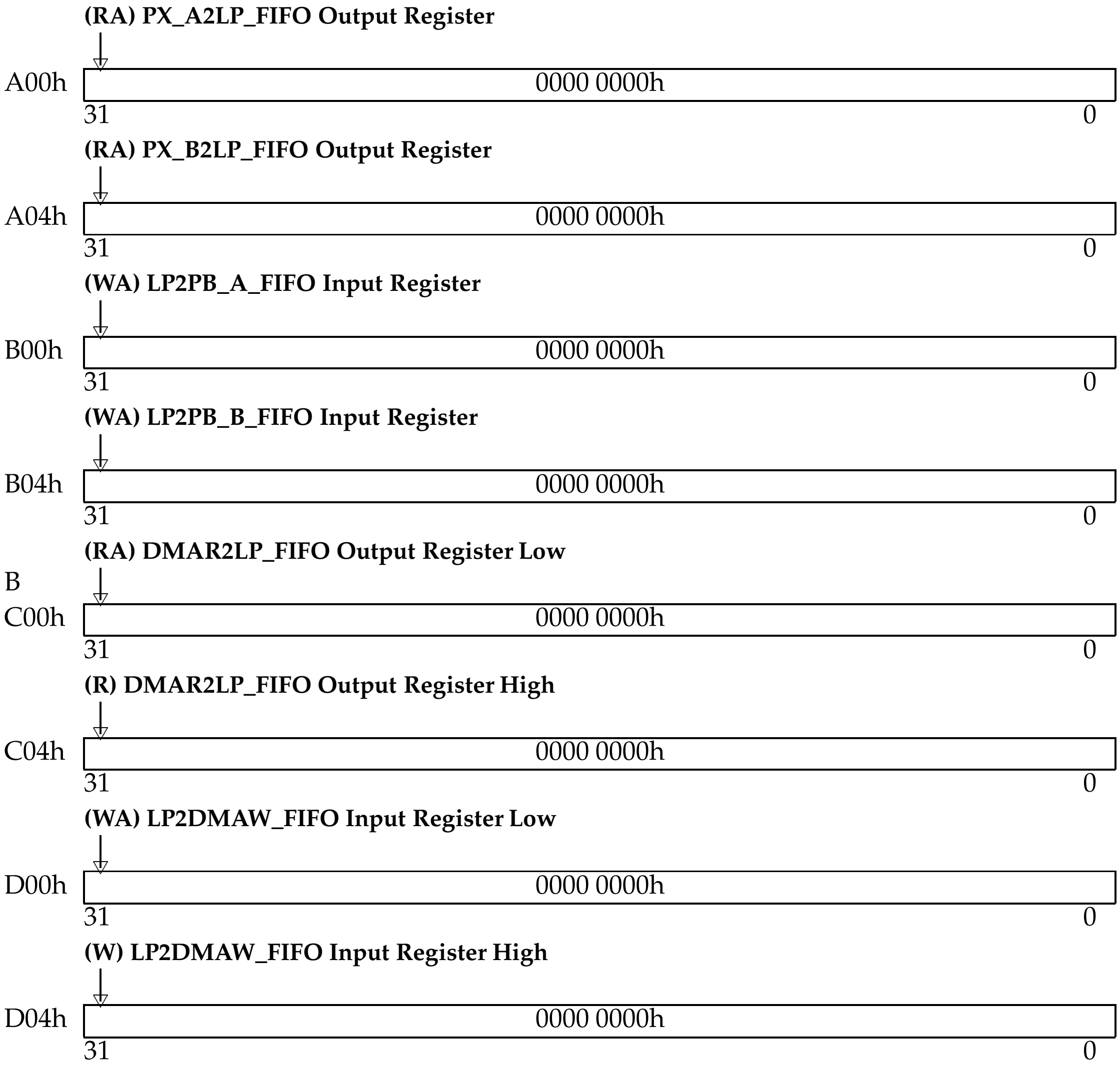


Figure 3.14: FIFO Access Registers

PX2LP_FIFO(A/B) Output Register
Output register of the fifo containing the packet headers detected on the HSL link. Reading this register starts a pop.

LP2PB_FIFO(A/B) Input Register
Input register necessary to write into the fifo the packet headers that are to be sent on the network. A write action in this register results in a push in the fifo. If the fifo is full, the write freezes the local processor.

DMAR2LP_FIFO Output Register high/low
This register is used to read the output of the fifo containing the data read in memory.

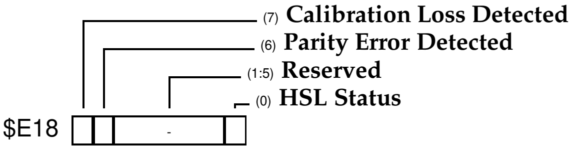
LP2DMAW_FIFO Input Register high/low
This input register allows the local processor to write a set of data in the fifo so that it can be moved to the host memory.

3.2 HSL Control and Status

HSL Control and Status Registers are located in the $E00 to $E1F address range.

3.2.1 HSL Status Register

HSL Status Register reflects the current state of the link.

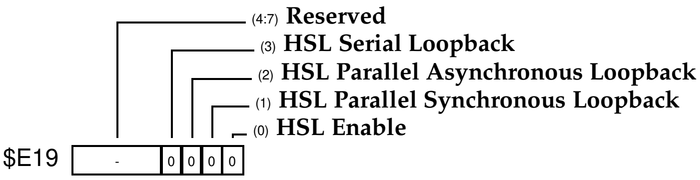


HS . . . . HSL Status. Read Only.
Set when the HSL becomes functional.

PED . . . . Parity Error Detected. Read and Clear
Set if at least one Parity Error has been detected, in reception on the link. That is, the Parity Error Counter Register is not null. Writing a '1' in PED decrements the Parity Error Counter by one. Thus every parity error can be detected by software.

CLD . . . . Calibration Loss Detected. Read and Clear.
Set if at least one calibration loss has been detected on the link. That is, the Calibration Loss Counter Register is not null. Writing a '1' in CLD decrements the Calibration Loss Counter Register by one. Thus every calibration loss can be detected by software.

3.2.2 Link Control Register



HE . . . . HSL Enable. Read/Write.
If this bit is reset, the link is totally shut down. When LE is set, the link enters its boot phase and tries to calibrate. If the boot phase is successful, the LS bit of HSL Status Register is set.
HE is reset at chip reset.

HPSL . . . . HSL Parallel synchronous Loopback. Read/Write
If this bit is set, the data sent by the link are internally synchronously looped back in the Serial Macrocell Interface Block (SMI). This can be used for test purpose only.
HPSL must not be changed while HE is set.
HPSL is reset at chip reset.

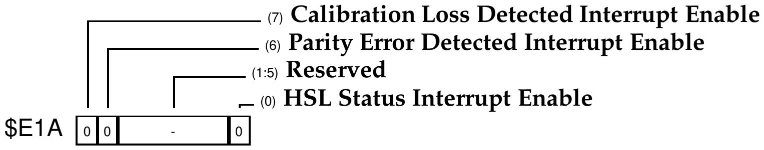
HPAL . . . . HSL Parallel Asynchronous Loopback. Read/Write
The same as the precedent bit but asynchronously. HPAL must not be

changed while HE is set.
HPAL is reset at chip reset.

HSL . . . . HSL Serial Loopback. Read/Write.
If this bit is set, the data sent by the link are internally looped back in the Serial Macrocell block. This can be used for test purpose only.
HSL must not be changed while HE is set.
HSL is reset at chip reset.

3.2.3 HSL Interrupt Register

This register is enables or not the IRQ# signal to be asserted when the corresponding bit of the Status Register is set.



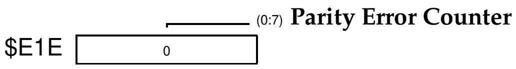
HSIE . . . . HSL Status Interrupt Enable. Read/Write.
IRQ# is asserted if both HS and HSIE are set.
HSIE is reset at chip reset.

PEDIE . . . . Parity Error Detected Interrupt Enable. Read/Write.
IRQ# is asserted if both PED and PEDIE are set.
PEDIE is reset at chip reset.

CLDIE . . . . Calibration Loss Detected Interrupt Enable. Read/Write.
IRQ# is asserted if both CLD and CLDIE are set.
CLDIE is reset at chip reset.

3.2.4 Parity Error Counter

Parity Error Counter is designed to monitor link performances. Read only.

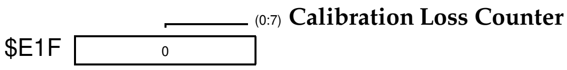


Parity Error Counter is incremented each time a character is received with a parity error. It is decremented each time a '1' is written in the PED bit of the Status Register.

Note that every parity error is reported, even if this error has no consequences, e.p. when the parity error affects an idle character between to packets.

3.2.5 Calibration Loss Counter

Calibration Loss Counter is designed to minitor link performances. Read only.



Calibration Loss Counter is incremented each time the LS bit of the HSL Status Register goes from '1' to '0'. This correspond to a calibration loss (link has been physically broken, or there is too much noise on the link). It is decremented each time a '1' is written in the CLD bit of the Status Register.

Note that every calibration loss is reported, even if this loss has no consequences, e.p. when the calibration has been lost and recovered while there was no traffic on the link.

CHAPTER 4
The local processor

This chapter decribes the local processor interface and its programming.

The local processor performs two independent functions :

- It controls the various NOÉ operators to ensure data transfers.
- It manages and implements the protocol defined by the user (e.g: Remote Write, RWU, ...)