****

***FPGA***

***SPECIFICATION***

**Project :** Iris3

**Document Title :** Profinet in FPGA specification

**Document Number and Revision :** 661-ARHW-002-0.02

**Author :** Jean-François Larin, ing #121322

**Date :** 2014-April-04

**Matrox Confidential information.** Do not circulate outside Matrox without the prior consent of the Matrox Engineering department. Advance Information. Contains information on product in the design phase of development.

**Revision history**

**This section lists all revisions for the current document and explains the modifications made for each new revision (the table below is only given as an example):**

|  |  |
| --- | --- |
| April/04/2014  (revision 0.01) | * Initial draft. |
| April/04/2014 | * Correction typo i211 vs i210 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Table of contents**

[1. Introduction 5](#_Toc384372646)

[1.1 The problem 5](#_Toc384372647)

[2. Block Diagram 5](#_Toc384372648)

[2.1 Software architecture 6](#_Toc384372649)

[3. Prototyping plan 7](#_Toc384372650)

[3.1 First Prototype 7](#_Toc384372651)

[3.1.1 First Prototype API 7](#_Toc384372652)

[3.2 Second prototype 8](#_Toc384372653)

[3.3 Open issues and other prototypes 8](#_Toc384372654)

[3.3.1 Third prototype 9](#_Toc384372655)

[3.3.2 Fourth prototype 9](#_Toc384372656)

[3.3.3 Fifth prototype 9](#_Toc384372657)

[4. APPENDIX A: RELATED DOCUMENTS 10](#_Toc384372658)

**Table of figures**

[Figure 1 proposed hardware architecture 5](#_Toc384372659)

[Figure 2 Profinet stack architecture 6](#_Toc384372660)

[Figure 3 Memory space map 8](#_Toc384372661)

[Figure 4 Real-Time part of the stack 9](#_Toc384372662)

# Introduction

The goal of this specification is to expose the design of a FPGA that will be used to implement Profinet acceleration through FPGA. We want to use that technology on Iris3 camera design and 4Sight-GPM Atom version.

## The problem

The main problem found in previous implementation of the Profinet protocol over Iris GT or 4Sight GP is related to the timing of packet transmission. Profinet requires real-time transmission of data. When the Profinet stack run on top of a Windows operating system, it becomes difficult to support the transmission of real-time traffic and it is impossible to guarantee that all deadlines will be met.

We want to solve that problem by offloading the real-time part of the Profinet protocol stack to a dedicated processor in an FPGA. That processor will have privileged access to the Ethernet port.

# Block Diagram

The following figure shows the hardware architecture of the proposed solution:

Microprocessor

B

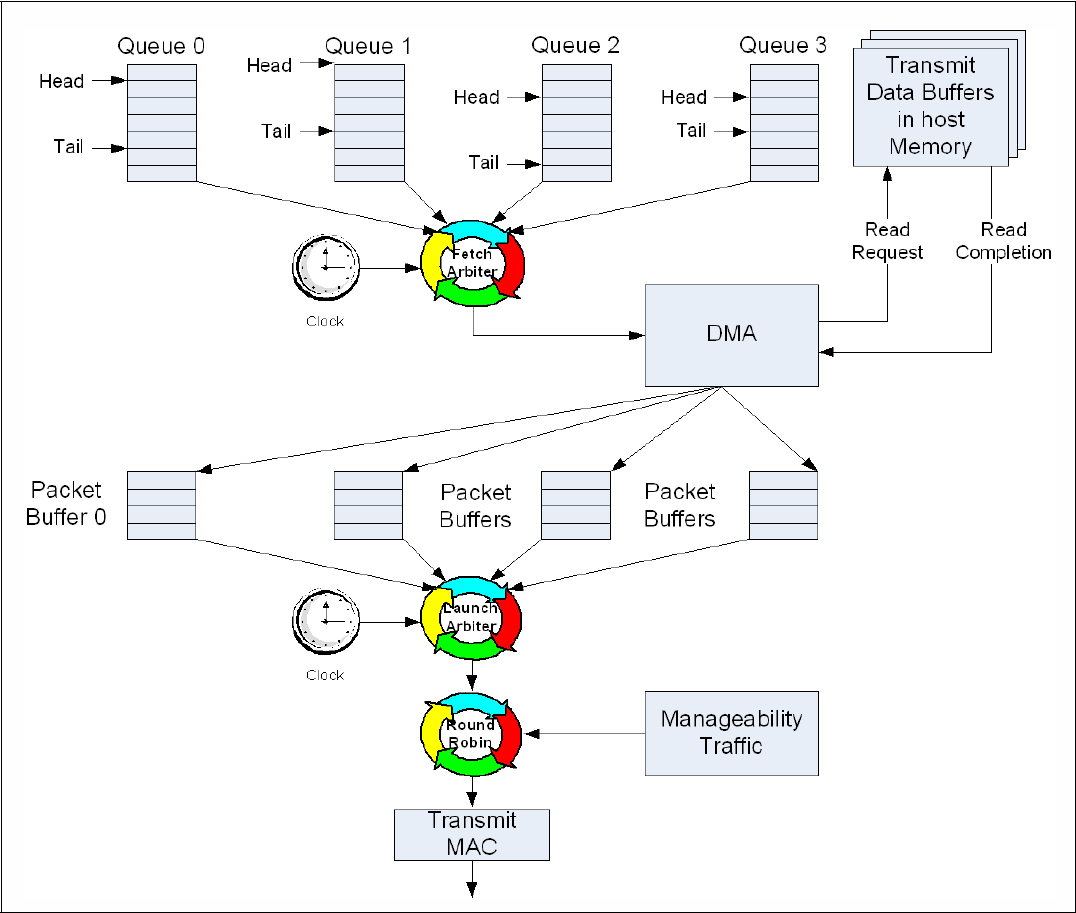
U

S

MAC

(rmii)

Host Interface



P

C

I

e

NCSI (100Mb)

 i210

GbE

Figure 1 proposed hardware architecture

On the left, the i210 Ethernet Controller is represented. That controller is connected to a PCIe link for normal transmit and receive operations. The Manageability Traffic bypasses all the queues and buffer, all the latency related to the PCIe bus and all the delays introduced by the software on the host (not represented in the figure). In our case, we use the NC-SI interface which runs a 100Mb in full-duplex.

On the right side, we have a FPGA. Inside the FPGA, we have a Microprocessor sub-system containing a MAC peripheral. On our prototype, we will use the Microblaze from Xilinx. Also, to communicate with the Profinet application on the host, we need an interface. In the case of the 4Sight-GPM, this interface will be the LPC bus. On the Iris3 case, this should be a PCIe interface.

## Software architecture

The figure below shows the implementation of Profinet stack Matrox purchased:



Figure Profinet stack architecture

Normally, the stack is made to be ported on new hardware and OS by modifying only the grey-green module with the « Adapt » tags. In our case, we have both a powerful, non-realtime host processor and a smaller but real-time fpga microprocessor, so we will need to review the division of the workload in the final implementation to optimize the performance vs software development time.

On the figure, there are 2 red arrows and adaptation layer through which packet transit. There is the Sock-Adapt layer, which resides on top of the TCP/IP stack. This traffic cannot be real-time because it is on top of the TCP/IP. We don’t see any need to route this traffic through the FPGA and the NCSI.

The second arrow between the BSP Adapt and the Telegram-Checker is used for the Profinet Real Time frames (using EtherType 0x8892). This traffic has to go through the NCSI.

# Prototyping plan

We plan on proving that Profinet can be ported onto the described hardware architecture and yield a reliable real-time implementation by building working prototype and evolving the prototypes toward the upcoming hardware platform.

## First Prototype

The first prototype will show that the NCSI can be used to bypass all the normal Ethernet-Queue-PCIe-NDIS-TCP/IP traffic. It will be built on the Y7449-01 PCB platform using the S6LX16 FPGA. On that platform, we have very limited memory internally to the FPGA, so the test we can do will be limited by the software we can put on the Microblaze.

For that prototype, the Microblaze simply has to send packet on the Ethernet. Reception of Ethernet packet will still be handled by the PCIe path to the NDIS in Windows. This also means that packet sent will use the same MAC address as the Ethernet Controller.

This test will enable us to show:

1. The usability of the NCSI for passthrough packet.
2. That the Management traffic can be sent in time even if there is heavy traffic between the host and the Ethernet (going through the PCIe bus).
3. The limitations of the LPC bus bandwith and the impact on the host CPU workload.

### First Prototype API

A dual-port memory will be used to send packets. Since we expect that the software will only need to send a single packet at every cycle time, the send interface will use a single buffer.

Handshake DW

Send packet buffer

(existing

memory

space)

0x0000

0x0800

0x0FFC

Figure Memory space map

The location of that space is hardcoded by both the BIOS and the FPGA implementation to 0xFEA00000. The existing Spider\_lpc memory space, containing the FPGA Build ID, is located in the first 2kbytes of the memory space. The second 2kbyte of memory is reserved for the send frame buffer. The software must write the buffer starting with the destination MAC address. Padding and CRC is added automatically by the hardware.

Software must write the packet length (excluding padding and CRC length) in the 11 LSB of the handshake DW. It should then set the next 5 bit (bit 15 to 11) to a non-zero value. This indicates that the software hands over the buffer to the hardware to send the packet. When the packet transmission is completed, the Microblaze will write back the Handshake DW to 0, thereby confirming that the packet transmission is completed. This means that the byte at 0xFFD must be the last byte written by the software. Also, software must poll that byte and verify it is set back to 0 before writing again in the Send packet buffer.

## Second prototype

For the second prototype, we are going to use the same platform as the first prototype, but it is going to add a precise time reference to the system. The goal is to generate an interrupt at fixed interval. The sleep functionality provided by Windows is imprecise and delay inaccuracy can add up and packet transmission misses their deadlines.

This test will enable us to eliminate the sleep delay in the packet transmission and find what the other causes of transmission delays are.

## Open issues and other prototypes

We envision that both enhancement described previously may not fix all problem in all cases. This section describes what we can plan as a final architecture to provide the best performance (lowest reliable cycle time).

It may be difficult or impossible to implement those models on the Y7449-01 prototype because of the limited internal memory, so we may have to wait for a new platform to prove those concepts.

### Third prototype

As another step to make the system more reliable, we should try to put the periodic packet generation in the Microblaze processor. Graphically, this means moving a part of the Profinet stack to the Microblaze processor. This would make sure we do not miss any real-time deadlines. However, it may be difficult to split the existing Profinet stack code this way. This idea as to be reviewed by the software team.

### Fourth prototype

The next step would be to move the whole real-time stack in the Microblaze. Graphically:

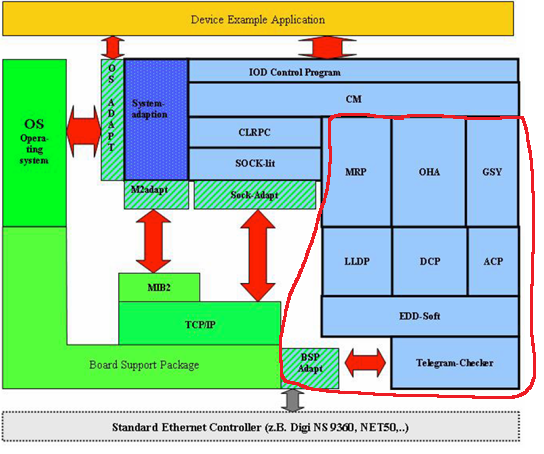


Figure Real-Time part of the stack

This would imply that all the EtherType 0x8892 traffic is redirected onto the NCSI. Then, the communication that has to go through the LPC bus is only the internal communication between some modules of the stack. If it is too complex to split the stack at that layer, it may be simpler to consider the next option shown below.

### Fifth prototype

We could also put the complete Profinet stack within Microblaze processor. The development environment already features TCP/IP stack and offers a POSIX kernel, with multi-priority task scheduling, interprocess communication, etc. This may be simpler to do since only modification to the adaptation layers of the Profinet stack would be required. The LPC interface would only need to transport the information between the Application and the IOD Control program shown on the figure above.

Note that in this scenario, more memory will be required, for both the code and the data of the Microblaze. If we consider this option because it will be more efficient (vs LPC bus and vs implementation time), we should considered the cost of larger FPGA or the cost of adding external memory to the FPGA, as an option. Also, we could consider executing code from the SPI flash, which may or may not be feasible.

# APPENDIX A: RELATED DOCUMENTS

Intel, Intel Low Pin Count (LPC) Interface Specification, (revision 1.1) August 2002, <http://www.intel.com/design/chipsets/industry/25128901.pdf> .

DMTF, Network Controller Sideband Interface (NC-SI) Specification, (Version 1.0.1) January 24, 2013,

<http://www.dmtf.org/sites/default/files/standards/documents/DSP0222_1.0.1.pdf>