Level Conversion Circuits

Transfer Knowledge

Carlos Gil Soriano
BE-CO-HT
carlos.gil.soriano@cern.ch

January 22, 2012

Abstract

Level Conversion Circuits project aims to replace old legacy systems targeted for pulse repetition, either in Blocking or RS485 format.

A complete list of legacy boards to be superseded can be found in the wiki page:

http://www.ohwr.org/projects/level-conversion/wiki

Revision history		
HDL version	**Module**	**Date**
0.1	Preliminary version	January 10, 2012
1.0	Final version	January 22, 2012

Contents

List of Tables

List of Figures

1 Level Conversion Circuit project structure

The project aims to substitute old boards by newer ones which will be able to cope with old functionalities and provide extra features.

Two sets of boards will be replaced, depending upon the pulse shape to be replaced:

- **Blocking pulse**
  The Blocking pulse is defined in the document [1].
- **RS485 pulse**
  The RS485 pulse has to be defined.

1.1 CONV-TTL-BLO replacement table

Board	EDMS ID	Functionality	
8 channel repeater	EDA-01490	TTL/$\overline{TTL}$/Blocking to Blocking	
16 channel repeater		TTL/$\overline{TTL}$/Blocking to Blocking	
CTDAC	EDA-01632	Blocking to TTL/$\overline{TTL}$	
LA-BLO-TTL	AB-001870	Blocking to TTL/$\overline{TTL}$	
LAF-BLO-TTL	AB-001871	Blocking to TTL/$\overline{TTL}$	
LASB-TTL-BLO	AB-001873	TTL/$\overline{TTL}$ to Blocking	
LA-GATE	AB-001876	TTL/$\overline{TTL}$ to Blocking	
LA-TTL-BLO	AB-001874	TTL/$\overline{TTL}$ to Blocking	
LAPF-TTL-BLO	AB-001872	TTL/$\overline{TTL}$ to Blocking	4μs pulse

Table 1: CONV-TTL-BLO replacement table

1.2 CONV-TTL-RS485 replacement table

Board	EDMS ID	Functionality
CTDAD	EDA-01600	TTL to RS485 transmitter
CTDCD	EDA-00925	TTL to RS485 transmitter
CTDCR	EDA-00948	RS422/RS485 receiver
CTDAR	EDA-00917	Optical E2000 to RS422
CTDLT	EDA-00916	TTL to Optical E2000 or RS422

Table 2: CONV-TTL-RS485 replacement table

1.3 Front and rear panel boards

Both CONV-TTL-BLO and CONV-TTL-RS485 projects consist on the same set of boards: a front board and a rear subsystem.

1.3.1 Front: ACTIVE

The front board contains all the active components. Is the only one receiving power from the crate. Consequently, it holds all the communication with the crate.
It offers I/O to front panel that are TTL compatible, and a SFP connector used for White Rabbit.
Status LEDs are offered, as well.

1.3.2 Rear: PASSIVE

The passive subsystem consist of two boards:

- **Motherboard**
  The moherboard connects the VME64x P2 interface with the piggyback connector. It is shared between Blocking and RS485 projects so as to reduce global BOM.

- **Piggyback board**
  The piggyback board is different in Blocking and RS485 projects. It offers I/O in the rear panels. The goal of the piggyback is to offer an easy-to-plug interface to operators.

2 Repositories structure

Both *CONV-TTL-BLO* and *CONV-TTL-RS485* share the same global structure:

- **doc**
- **hdl**
- **pcb**

2.1 CONV-TTL-BLO

- **doc**
  It holds all the generic design documentation of the board and files used in OHWR repo:
  - **OHWR**
    This subfolder contains pictures used both in the wiki pages of the issues section.

  Some extra subfolders are used: Blocking, HDL and HDLguide. The first contains the source files and output of the documentation for [2], [1] and [3].

- **hdl**
  As *CONV-TTL-BLO* and *CONV-TTL-RS485* will share a lot of IP cores for either basic or extended functionality, all of the IP cores are located in *CONV-TTL-BLO*. This repository is depicted below:

  - **IMAGES** It contains the top files for basic repetition in *CONV-TTL-BLO*:
    - **image0**
      This is the first bitstream with a very coarse basic repetition.
    - **image1**
      It corresponds to the image loaded into the *CONV-TTL-BLO V1* installed in PS facilities. The logic runs at 200 MHz and the outputs are glitch-free.
  - **basic_trigger**
    It contains the IP core with clocked repetition. It is the core used in image1 bitstream.

- **basic_trigger_async**
  It contains the IP core with lowest achievable jitter. No profiling in the place and route (it can lower the jitter value inyected by the FPGA).

- **ctdah_lib**
  This is the library that contains generic modules used along the rest of IP cores within this project.

- **i2c_slave_wb_master**
  The I2C slave IP core is located here.

- **m25p32**
  The M25P32 IP core is located here. It manages access to the Flash Memory thanks to the SPI core developed for this project.

- **multiboot**
  The multiboot IP core is able to access Xilinx's ICAP primitive to allow SPI Flash Memory reprogramming from a given set of locations.

- **rtm_detector**
  Just some files to check for proper RTM detection.

- **spi_master_multifield**
  The SPI master IP core can be found in this folder. It allows SPI communication to perform writes and read-backs. It is developed to be used together with M25P32 module.

- **trigger**
  This is the folder to work on the extended trigger functionality.

- **wr_core_demo**
  Some White Rabbit files to perform a test.

- **pcb**

  - **conv-ttl-blo**
    Files of *CONV-TTL-BLO V1.*

– **conv-ttl-blo-v2**
Files of *CONV-TTL-BLO V2*.

– **conv-ttl-rtm**
Files of *CONV-TTL-RTM V1*.

– **conv-ttl-rtm-blo**
Files of *CONV-TTL-RTM-BLO V1*.

– **doc**
It contains all the reports of the V2 board reviews and a report with all the issues found in V1 [4].

2.1.1 IP core structure

Every IP core developed (*i2c_slave_wb_master*, *m25p32*, *multiboot*,*spi_master_multifield* and *trigger*) follow the same structure:

- **doc**
It holds the source files for generating beautiful, LaTeXed documentation, like this document.

- **project**
It holds all the .xise, .gise files and all the temporary ones. Inside this folder it can be found a *wave.do* file (typically inside the *waveform* subfolder).

- **rtl**
It holds all the RTL files. All the IP cores follows the following design schema:

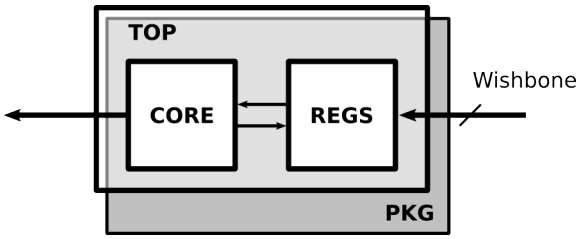

Figure 1: IP core structure

The *top* files interconnects the *core* (which has all the logic, controls the FSMs and the outputs) with the *regs*, responsible for controlling access to internal registers via *core* or a *wishbone* interface.
Additionally the *package* provides the definition of constants, register structures (defined as records) and translation functions for them to/from *std_logic_vector*.

All the cores follow the ESA and CERN@BE-CO-HT good VHDL practices, Alessandro Rubini's advices on how to write mantainable code and the comments are doxygenized.

- **test**
  It holds all the test files but the *wishbone_driver*, which is located in ctdah_lib folder. The test is written in VHDL as is as structured as follows:

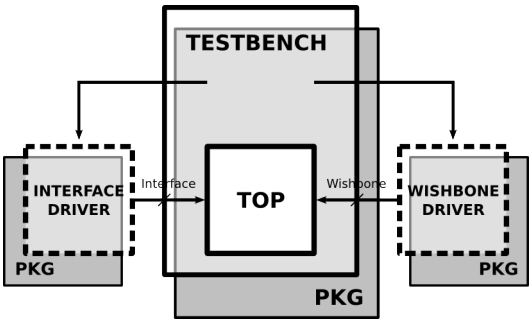

Figure 2: IP core testbench structure

The dashed lines of both drivers means that the cores are not intended to be synthesizable.

Some of the testbench generates a log file to quickly check the result of the test. Some others testbenches make use of *ModelSim SignalSpy* library to be able to read and check internal signals of given subcomponents, without the need of modifying files (either definition of VHDL ports or adding precompiler lines to the code).

2.2 CONV-TTL-RS485

It follows the same structure as *CONV-TTL-BLO*. The IP cores to be used in this board correspond to the ones developed for *CONV-TTL-BLO*.

3 Blocking boards replacement: CONV-TTL-BLO

3.1 Legacy boards

3.1.1 Problems faced

Old Blocking modules employed **three magnetically coupled inductors difficult to replace**. Because of this fact, the design of CONV-TTL-BLO started. Hence, been able to easily replace the components was of paramount importance since the very beginning of the designing stage.

Apart from this point, the need of **remote control and management** of the boards had an increased importance. Including a FPGA copes with this problem.

3.1.2 Jitter

Due the simplistic way for controlling the replication of the pulses, these modules have a low figure of jitter.

3.2 CONV-TTL-BLO design

The functional specifications of CONV-TTL-BLO can be found in [2]. We can divide between basic and extended functionalities.

3.2.1 Basic functionalities

Aimed to be fully compliant with legacy systems:

- Replicate Blocking pulses.
- Translate to/from TTL and $\overline{TTL}$ from/to Blocking.
- Provide galvanic isolation in the links.
- To be glitch-free.

The design of the Blocking driver follows a Flyback topology [5], which inherently provides galvanic isolation. So as to properly drain up the remaining built-up magnetic current in the inductor, a snubber circuit was added. The snubber circuit can be found in [3].
Because of the Flyback topology, the three-widing coupled inductor was replaced by a 1:1 coupled inductor. The selection of this coupled inductor follows the premise of minizing the chances of being out of stock in the future.

In the event of having to choose a different coupled inductor please check the correct values and margins for:

Manufacturer	Series
Pulse Electronics	PF0552 and PF0553
Coilcraft	MSD1278
Coiltronics	DR74 and DR125

Table 3: Coupled inductor series replacements

- *Saturation current*
- *Primary inductance*
- *Coupling ratio*
- *Footprint*

3.2.2 Extended functionalities

They provide increased control thanks to the FPGA and the use I2C line of the VME64x interface by means of *SERA* and *SERB* pins.

- Selectable pulse length.
- Time-tagging of repeated pulses via White Rabbit.
- Log of inputs and repetitions.
- Remote reprogramming of flash-stored FPGA bitstreams via I2C interface.

3.2.3 V1 schematics overview

CONV-TTL-BLO, namely CERN EDMS-02446 project, is described below.

- **TOP page**

  The top page shows the main blocks in which the project is divided into.

  Conceptually, the left part of the schematic corresponds to the front panel and some clocking resources, used in White Rabbit.
  In the center we can found the FPGA, a Spartan 6 LX45T, which is the logic core of the design. This model of Spartan is used is many others within OWHR and BE-CO-HT, making portability of IP cores seamless. Due to the fact of being a 6 series, SerDes functionality will be later on used in V2 to improve sampling jitter of repeated pulses.

All the blocks corresponding to the connections with VME64x backplane and rear transistion module are over the right part of the top schematic page.

- **Power supplies**

  Power supplies are covered in the next two pages.

  In the first one, the rails for all the logic ICs and FPGA power is provided. Decoupling capacitors for the FPGA are found in it. Values and sizes of the FPGA decoupling network are consistent with the recommendations of the vendor. All the I/O banks in the FPGA are tied to 3.3 volts.

  The second page corresponds the power supply needed for the generation of the 24 volts amplitude blocking pulses. Correct dimensioning for impulsive responses, gain and margin budgets were calculated and verified with Texas Instruments Switcher Pro. It should be noted here that the Power MOSFET in V1 was changed by a similar performance one, due to soon obsolescency of the part.

  Just as a reminder for new designers, the selection of critical capacitors was taken with the view of a long-lasting worklife of the boards. Hence, the use of conductive polymer capacitors is recommended. In this design, OS-CON capacitors are used instead in electrolitic ones. In designs in which tantalum capacitors are of expected use, a close look to POSCAP is recommended. In either case, improved conductance and better temperature behaviour is expected over electrolitic and tantalum, respectively.

- **FPGA**

  FPGA connections are splitted in two pages: *FPGA BANK* and *MGTX*.

  With regard to *FPGA BANK*, all I/O banks are powered to 3.3 volts, as previously indicated. Banks 0 and 1 hold clocking resources, VME64x I2C connections, LEDs and some extra features. Banks 2 and 3 plug all the pulses connections with no particular care in the connections of the pins. This was a limitation in the performance of V1, solved in V2. So as to much improve perfomance in the clocked pulse repetition, **SerDes functionality** should be used. By reading the advanced I/O resources guidelines from Xilinx, connections should be done as in V2:

two non differential input sources must not belong to _P and _N pins of a same differential FPGA input.

Obviously, clockless pulse repetition will have the lowest possible jitter figure.

- **White Rabbit clocking resources**

  Just copied from SPEC project, V2 indepently configurable DAC capability. All the decouplings are exactly the same as in SPEC.

- **JTAG**

  In this page the JATG connection, switches and external Flash memory is depicted. External reset circuitry is added in V2.

- **VME64X**

  VME64X has two connectors P1 and P2. P1 is used for I2C communication and P2 serves as a connection from front to rear: Blocking I/O signals, rear LEDs, motherboard and piggyback IDs.

  When attaching a V1 into a crate it can produce VME64x conflicts in other VME64x boards, due to not daisy-chaining *bus grant* and *interrupt ack* lines. This problem is solved in V2.

- **INPUT UNIT**

  The input unit uses an optocoupler preceeded by a DC rejection filter (a differentiator). The line is terminated and protected via a TVS. In V2 a Schimtt trigger is added right after the optocoupling to sharpen the change of the signal.

- **OUTPUT BLO**

  BCT25244 is used because of two reasons:

  - Powerful drive.
  - Reduce stocks cause it is already used in CTRs.

  A net is misconnected in V1. Issue solved in V2.

An important feature implemented in the design is being glitches-free. This is achived in different ways:

  - Careful startup of the FPGA.
  - Placing of pull-ups or pull-downs in open enable pins.

- **OUTPUT UNIT**

  The output unit consist of a flyback

3.2.4 Changes from V1 to V2

While debugging V1, issues appeared.

- **Power Supplies**

  - Changes to better adjust the voltage for outputting Blocking 24 volts:
    Issues **452, 455, 458**

  - Remove some noise and adding better protection:
    Issues **504, 517**

  - Change of MOSFET due to be obsolete soon:
    Issue **638**

- **Pins misconnections**

  - A floating BCT25244 pin:
    Issue **463**

  - VME64x Daisy chain misconnected:
    Issue **502**

- **Blocking stage**

  - Removal of speed-up diode (causes increased temperature dependency), use of 5V plane and including an output resistor to improve RC when no termination is used in a device:
    Issue **462**

- **Front panel**

  - Replacement of status LEDs for a bicolour LED array.

3.2.5 Modification to do on V2

- **Power Supplies**
  Change the Blocking power supply low resistor feedback to produce Blocking pulses at 24V and not 21V:
  Issues **679**

3.3 Piggyback design

CONV-TTL-RTM-BLO, namely CERN EDMS-02453 project, is described below.

- **TOP page**
  It connects the VME64X P2 connector with the double row, one hundred Semtech connector for the piggyback.

- **Panel and Leds**
  Just the connectors and nothing else. The TVSs are already in the motherboard.

3.4 VHDL design

The VHDL design has to achieve two milestones:

- **Basic functionality**
  It covers the repetition of the pulse as in the previous legacy boards.

- **Extended functionality**
  It improves the basic functionality by time-tagging via White Rabbit the pulses, allowing remote monitoring and reprogramming of the device.

3.4.1 Basic functionality

The basic functionality is covered in the design used for the *CONV-TTL-BLO V1* card installed in the test loop in PS facilities.

The IP core has to:

1. Detect and replicated received pulses once they are not considered as glitches.

2. Provide a safe and reliable working life. Not producing glitches at startup.

The IP core sampling frequency for the inputs is 200MHz. The internal 200MHz system clock is generated from the 125MHz clock. Then the induced sampled jitter will have a maximum value of 5 ns.

The following table shows the configuration of the *CONV-TTL-BLO V1* board installed in the test sage loop in PS facilities:

Parameter	**Value**
Pulse height	24 volts
Pulse length	1 μs
Pulse rise time	80 ns
Pulse fall time	100 ns
Minimum worst-case repetition jitter	5 ns
Channel LEDs blinking length	250 ms

Table 4: Configuration for *CONV-TTL-BLO V1* in PS safe loop test

3.4.2 Extended functionality

As said before, extended functionality consist of three main blocks:

1. Pulse time-tagging
2. I2C communication: access and monitoring
3. FPGA reprogramming functionality

As points 2 and 3 represented bottlenecks for starting with the PTS, design priority was given to them.

- **I2C communication**

  I2C module is of paramount importance for the development of the rest of functionalities. Being the link between FPGA and outside world, makes I2C the first module to develop before PTS can be carried out.

IP core	**Core**	**Test**	**Guide**
I2C_slave_wb_master	Link	Link	[6]

Table 5: *I2C_slave_wb_master* IP core files and documents

  The *I2C_slave_wb_master* is an I2C slave core which provides wishbone master and slave interfaces. The former one serves a bridge from I2C to internal wishbone modules. The second one lets the *I2C_slave_wb_master* core registers to be easilly accessed from the I2C interface by by-passing the I2C wishbone master interface to the slave. Providing these two interface lets the IP core be consistent and homogeneous with the rest of cores designed for *CONV-TTL-BLO* and *CONV-TTL-RS485*.

  More information about the IP core can be found in its guide [6].

- **SPI Flash Memory reprogramming**

  In order to reprogram the SPI Flash memory different IP cores are needed.

  To start off, a SPI master core is needed to place SPI transactions into the memory. It should be noted that the core developed for that has

taken into account parametric settings to meet timing.

Secondly a M25P32 memory handler was designed to write and read from memory in an easy and optimal way. Memories from the same family can be used upon some modifications in the package file. Flash pages will be written from an internal buffer. Thus, the way to program the Flash memory consist of writing the internal page buffer of the memory handler and acknowledge a write into a certain page. By doing this, in all the pages required by the raw bitstream, the new bitstream will be loaded correctly.

The last of the modules corresponds with the ICAP controller, called multiboot core. This core instanciates the Xilinx's ICAP primitive according the Xilinx's recommendation and let the user start a load bitstream process from the I2C via wishbone bridging.

IP core	**Core**	**Test**	**Guide**
spi_master_multifield	Link	Link	[7]
m25p32	Link	Link	[8]
multiboot	Link	Link	[9]

Table 6: SPI Flash Memory reprogramming IP cores

4 RS485 boards replacement: CONV-TTL-RS485

4.1 Legacy boards

4.1.1 Use

CONV-TTL-RS485 will be use as a more advance card that CONV-TTL-BLO. It will be able to replicate pulses as in its Blocking counterpart and redistribute GMT and timing information.

4.1.2 Additional features

With regard to legacy designs, CONV-TTL-RS485 will be able to detect insuficient differential level in the reception lines. Thus, defective and broken links can be detected.

4.2 CONV-TTL-RS485 design

- **TOP page**
  The division of this project is the same as CONV-TTL-BLO. The driver stage for outputting to the rear module has changed from Blocking to RS485.

- **Power Supply**
  In this project the Blocking power supply has been removed. The line filters added in CONV-TTL-BLO V2 have been added to CONV-TTL-RS485 V1.
  The decoupling FPGA capacitors are the same and the external reset is included in this page.
  It should be remarked that all FPGA I/O banks are powered at 3.3 volts.

- **FPGA BANK**
  When CONV-TTL-RS485 was designed, the sampling jitter problem was not found. Subsequently, a change in the I/O trigger FPGA pins for the pulse repetition should be carried out in the same fashion as it was from CONV-TTL-BLO V1 to V2.

- **CLOCKS**
  Two changes from CONV-TTL-BLO V1 have been done.
  The first corresponds to let individual configuration of DACs from the FPGA. The second is the use of cleaner power supply for the DACs.

Voltage drop in ferrites was measured and it is negligible.

- **VME64X**
  All the issues of bad daisy chaining in CONV-TTL-BLO V1 have been addressed.

- **MGTX**
  A four position microswitch to enable network identification (which accelerator/timing domain the card is attached to) is added to the board.

- **FRONT TTL**
  The problems of missconnections and bad power supplying from CONV-TTL-BLO V1 has been corrected in RS485. Same antiglitch features applies for the pull-ups and pull-downs.

- **FRONT PANEL**
  In this page the new bicolour LED array was introduced, it works together with Matthieu's IP core.

- **INPUT UNIT**
  By using same signed thresholds in the RS485 receivers, an bad/dead link detection can be carried out.

- **OUTPUT UNIT**
  Same transceivers as in reception are used. They are compliant with speed link for GMT replication.

- **JTAG**
  Changes needed in CONV-TTL-BLO V1 have been carried out here.

5 CONV-TTL-RTM: RTM Motherboard

The motherboard is shared between both the CONV-TTL-BLO and the CONV-TTL-RS485. It provides connection to piggybacks from the VME64X P2 connector.

5.1 CONV-TTL-RTM design

The RTM motherboard consist of a VME64X P2 connector and a two rows 100 pin male connector. The signals are bypassed from the P2 connector to the 100 pin connector and, for protection measures, TVS are included.

6 ELMA crate

An updated documentation of the ELMA crate System Monitor card can be found in [EDMS].

6.1 How to communicate

The way ELMA crate System Monitor communicate follows the instruction specified to Mihai Savu in [10]. Hence, to communicate effectively with both the CONV-TTL-BLO and CONV-TTL-RS485 will consist of:

1. A byte for I2C address and operation.
2. Two bytes for Wishbone Addressing.
3. Four bytes of data.

To send the I2C commands from ELMA crate, firstly it should be accessed via telnet. It should be noted that the crate must be connected to the CERN socket in which the SysMon is registered to, otherwise the device will not have access to the network. Once accessed via telnet, two commands can be issued *writereg* and *readreg*, both specified in [11].

A comprehensive explanation of the comunication is found in the documentation of I2C VHDL core specifically done for this project [6].

6.2 Problems

- I2C communication I2C in ELMA crates runs at low speed 400KHz to 100KHz depending on the capacitive load attached to the SERA, SERB pins. A better pull-up strategy could have been carried out to solve these problem.

  Issues **497, 500, 523**

  It should be kept in mind that in **high capacity crates, this problem can be more serious.**

- Chasis not so well-built 1U ELMA crate lacks of the robustness of other crates. When a crate lies on a table with a card plugged in, some badly cut through hole pins can touch the chasis, producing shortcircuits.

  Issue **537**

7 People involved in Level Conversion Circuits

Success in a project is due to the joint work of a team. A list of all the people involved in the project can be found in the table below. I would like to take advantage of this line to thank them all again for their help and work.

Design team	
Erik Van der Bij	Manager
Matthieu Cattin	Technical supervision - Installation
Tomasz Wlostowski	Technical help
Installation - operators	
Claude Dehavay	Installations manager
Emmanuel Said	Main Blocking/RS485 installator - cabling - front/rear panel
Olivier Barriere	Installation - front/rear panel
ELMA crate	
Magnus Bjork	ELMA crate CERN's responsible
Boehr Timo	ELMA crate links
Frank Weiser	ELMA crate links
Silviu Bodeanu	ELMA System Monitor responsible
Mihai Savu	ELMA I2C C developer
DEM	
Betty Magnin	Manager for PCB design and manufacturing
William Billereau	Manager for PCB design and manufacturing
Benoit Civel	Responsible for layout in *CONV-TTL-BLO V1*
Claude Andouillet	Responsible for layout in *CONV-TTL-RTM*,*CONV-TTL-RTM-BLO* and *CONV-TTL-RTM-RS485*
Bruno Recoldon	Responsible for layout in *CONV-TTL-BLO V2* and *CONV-TTL-RS485*

Table 7: People involved in the project

A Development roadmap

A.1 CONV-TTL-BLO

1. **Remote reprogramming**
   Remote reprogramming is the key to wind up the development of the IP core chain needed to perform the PTS in the boards.
   To achieve this, the *i2c_slave_wb_master*, *m25p32* and *multiboot* cores should be put together and thoroughly tested. An image has been provided and can be found in:

   - **CONV-TTL-BLO/hdl/IMAGES/image1**

   It should be noted that *image1_core.vhd* has been splitted from *image1_top.vhd* to allow *image1_core.vhd* be reused in RS485.
   Most of the work to be done in the VHDL will be testing that a new raw bitstream has been loaded into the FPGA. It should be noted that good understanding of Xilinx's ICAP is needed [12].

2. **PTS**
   After having a stable bitstream with the basic repetition plus the IP core chain indicated above, PTS will be "easilly" carried out. That will involve quite a lot of work.
   Unlike other projects, such as *SPEC*, in which the PTS could send commads directly from the host computer, now the commands should be by-passed over telnet to the SysMon.

3. **Pulse time-tagging**
   To precisely time-tagging a White Rabbit core must be included in the FPGA, then all the pulses will be logged and they will be accessible via I2C. *trigger* core should be worked on to finally meet the requirements.

4. **Reduce jitter**
   Reducing jitter means using either the asynchronous solutions in *basic_trigger_async* or writing an IP core using the SerDes functionality in Xilinx's Spartan 6. A good look on [13] is recommended.

A.2 CONV-TTL-RS485

1. **Strongly discuss about datagrams parsing**
   CONV-TTL-RS485 will offer more functionalities than *CONV-TTL-BLO*. Correct dimensioning of the application memory should be taken into account. The memory requirements will depend upon Jean-Claude's

need for parsing datagrams and place orders into the accelerators. External memories could be used to improve performance.

2. **Producing a V2**
   It should be kept in mind that, in order to improve jitter perfomance, same changes in FPGA I/Os as in *CONV-TTL-BLO V2* apply.

References

[1] C. Gil Soriano. Standard Blocking Output Signal Definition for CTDAH board, September 2011. http://www.ohwr.org/documents/109.

[2] C. Gil Soriano. Standard Blocking Repeater in VME64x Format, Functional Specifications, August 2012. http://www.ohwr.org/documents/104.

[3] C. Gil Soriano. A Triggered Monostable Blocking Oscillator Used in legacy Channel Repeaters, August 2012. http://www.ohwr.org/documents/94.

[4] C. Gil Soriano. Report of issues in CONV-TTL-BLO v1.0 EDA-02446-V1-0, October 2012. http://www.ohwr.org/projects/conv-ttl-blo/repository/show/pcb/doc/issues/conv-ttl-blo?rev=basic_trigger_addingGenerics.

[5] C. Gil Soriano. Flyback circuit outputting Standard Blocking for CTDAH, September 2011. http://www.ohwr.org/documents/110.

[6] C. Gil Soriano. I2C Slave to Wishbone Master module. Technical Report v0.9, CERN, BE-CO-HT, November 2012. http://www.ohwr.org/projects/conv-ttl-blo/repository/revisions/master/changes/hdl/i2c_slave_wb_master/doc/i2cSpecs.pdf.

[7] C. Gil Soriano. SPI master multifield HDL core. Technical Report v0.9, CERN, BE-CO-HT, October 2012. http://www.ohwr.org/projects/conv-ttl-blo/repository/revisions/master/changes/hdl/spi_master_multifield/doc/spiSpecs.pdf.

[8] C. Gil Soriano. m25p32 manager HDL core. Technical Report v0.9, CERN, BE-CO-HT, October 2012. http://www.ohwr.org/projects/conv-ttl-blo/repository/revisions/master/changes/hdl/m25p32/doc/m25p32.pdf.

[9] C. Gil Soriano. Multiboot HDL module. Technical Report v0.9, CERN, BE-CO-HT, January 2013. http://www.ohwr.org/projects/conv-ttl-blo/repository/revisions/master/changes/hdl/multiboot/doc/multiboot.pdf.

[10] S. Iglesias Gonsalvez. ELMA Crates Specification, Access to board data using SNMP and I2C. Technical report, CERN, BE-CO-HT, February 2011. http://www.ohwr.org/documents/227.

[11] NEW Sysmon Manual. Technical Report v1.11, ELMA. http://www.ohwr.org/documents/226.

[12] Spartan-6 FPGA Configuration User Guide. Technical Report UG380 v2.3, Xilinx Inc., July 2011. http://www.xilinx.com/support/documentation/user_guides/ug380.pdf.

[13] Spartan-6 FPGA SelectIO Resources. Technical Report v1.4, Xilinx Inc., December 2010. http://www.xilinx.com/support/documentation/user_guides/ug381.pdf.