

A 10 Gbps OTN Framer Implementation Targeting FPGA Devices

Guilherme Guindani[‡], Frederico Ferlini[‡], Jeferson Oliveira[‡],

Ney Calazans[‡], Daniel Pigatto^{*}, Fernando Moraes[‡]

[‡]PUCRS, Av. Ipiranga 6681, P. 32, Porto Alegre, Brazil

^{*}DATAKOM, Av. França 735, Porto Alegre, Brazil

{guilherme.guindani, ney.calazans, fernando.moraes}@pucrs.br, {fferlini, jeferson.co}@gmail.com, daniel@datacom.ind.br

Abstract—Integrated circuits for very high-speed telecommunication protocols often use ASICs, due to their strict timing constraints. This scenario is changing, since modern FPGAs, implemented in 65 or 45 nm technologies achieve high operating frequencies, and serializer/deserializer hardwired modules enable the reception of high speed aggregated rates (e. g. 10 Gbps or more), spanning the input stream for internal parallel computation. This paper presents a complete solution for an Optical Transport Network framer using FPGA devices. The framer receives a 10 Gbps stream originated from optical fiber medium, extracts its payload information, and transmits payload data at 10 Gbps. A working prototype was implemented in Virtex-4 and Virtex-5 devices. (*Abstract*)

Keywords—FPGA, OTN (Optical Transport Network), Telecommunication Circuits, Framer (key words)

I. INTRODUCTION

Due to the large scale globalization process active nowadays, information exchange has become a critical issue, requiring the elaboration of faster, flexible and reliable computer networks. The migration of network technologies to faster protocols (Gigabit Ethernet and 10 Gb Ethernet) forces the utilization of optical fiber links in both local (LAN) and metropolitan (MAN) network backbones.

To meet the increasing demand for increased bandwidth using optical fiber links, and to support 2.5 Gb, 10 Gb and 40 Gb broadband services, a new optical transport network layer was developed, the Optical Transport Network (OTN) [1]. OTN is the only standard capable of transporting 10GbE LAN PHY entirely.

Different from SONET-SDH that is time multiplexed (TDM), the OTN protocol is multiplexed in wavelength (WDM), lowering the costs of the network. However, the main characteristic of the OTN standard is the presence of an error correction structure, based on the Reed-Solomon (255, 239) algorithm. This structure may correct up to 128 bytes in burst for each frame, enabling the use of longer optical links.

Xelic XCO2 [2] is a commercial soft core suited for either FPGA or ASIC implementation. The system side can accept the following client signals: (i) ODU frames; (ii) synchronous or asynchronous CBR10G, ATM or GFP; (iii) non-specific client bit streams. At the system side, data transfer takes place at a nominal rate of up to 9.953 Gbps using a 64-bit data bus, operating at 155.52 Mbps. At the OTN line side data is transferred at an OTU2 rate,

using a 64-bit data bus operating at 167.33 Mbps, or at an ODU2 rate using a 64-bit data bus operating at 156.83 Mbps. XCO2 has an embedded scrambling and FEC encoding machines but no embedded decoding unit.

Broadcom BCM8512 [3] is a 10 Gbps transport integrated processor on a single chip that operates with SONET/SDH or 10 Gbps Ethernet on the client side. It is implemented using 0.13 μ m CMOS technology. This product includes FEC encode and decode modules using the Reed-Solomon RS (255,239) algorithm. However, no core version of this chip is available, and the vendor mentions no port to FPGA.

The two examples of commercial OTN framers imply buying a license or a chip. In addition, commercial solutions for FPGAs as the one presented do not have a complete solution integrated in a single device, requiring external components e.g., Reed-Solomon FEC decoder.

The objective of this paper is to present a *complete* solution for an OTN framer (also called an OTN transponder), targeting FPGA devices. The solution includes the OTN frame reception and transmission, frame disassembly, reception/transmission of client data, the Reed-Solomon FEC circuitry, and an interface to manage the system through embedded software.

The OTN design described here is a joint development effort involving an academic research group and a telecom company, with the main objective to incorporate the OTN technology into products of the company.

The rest of this paper comprises four sections. Section 2 describes the OTN protocol. Section 3 presents an overview of the proposed OTN framer architecture, while Section 4 describes the process of validation for the proposed architecture. Section 5 presents conclusions and directions for future work.

II. THE OTN PROTOCOL

The ITU-T is a branch of the International Telecommunication Union (ITU) responsible for analyzing and organizing groups to study and create recommendations for the telecommunication field. The Optical Transport Network (OTN) standard is described on the G.709 ITU-T recommendation, which defines an OTN interface as a set of elements for optical networks capable of providing transporting functionality, multiplexing, routing, management and supervision of optical channels. The OTN interface must have the ability to carry signals from different types of clients, as shown in the Figure 1.

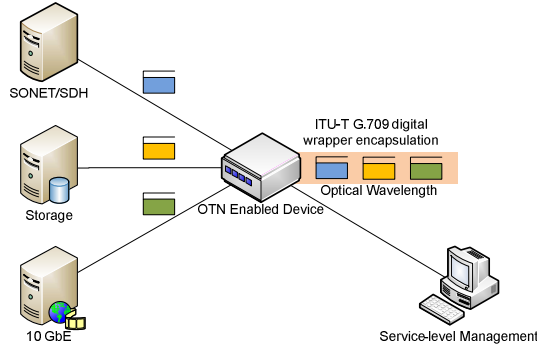


Figure 1 – Distinct signal sources transported over OTN.

According to [4], the OTN frame is composed by 16 lines of 255 bytes, and is divided in three main blocks: overhead (16 bytes), payload (3808 bytes, in 238 columns) and FEC (256 bytes in 16 columns). The OTN transmission does not follow the logic structure of the frame. It is transmitted column by column as depicted in Figure 2.

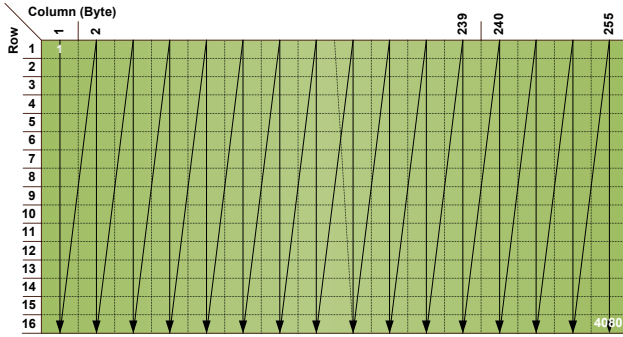


Figure 2 – The OTN Frame transmission sequence.

The OTN standard uses clock regeneration hardware on its receivers, therefore, long sequences of “0”s or “1”s can compromise the clock regeneration process and should be avoided. To avoid those long sequences, OTN transmitters use a scrambling process on the OTN frames before transmission. The scrambling process operates conceptually as a Linear Feedback Shift Register (LFSR), using the generating polynomial $1 + x + x^3 + x^{12} + x^{16}$. The output of the scrambling process (Figure 3) is added to each bit of the multiframe.

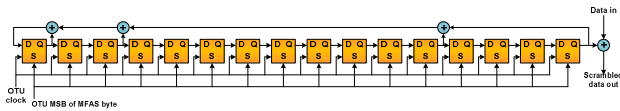


Figure 3 – Conceptual RTL implementation of the OTN scrambling process.

Recommendation G.709 defines the OTN multiframe (Figure 4), which contains 4 frames (4080 bytes lines, totalizing 16320 bytes). The OTN multiframe is organized in lines, and is composed by the overhead, payload and FEC for each line. The OTN multiframe is transmitted line by line.

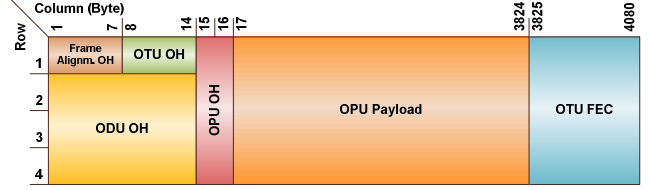


Figure 4 – Structure of the OTN multiframe.

Scrambling is applied after the FEC calculation for all multiframe bytes with the exception of the FAS (Frame Alignment Signal) bytes. This process is symmetric, i.e., the same process used for scrambling the transmission signal, is used during the receiving process to obtain the original descrambled signal.

III. ARCHITECTURE OVERVIEW

This Section presents an overview of the proposed transponder architecture. Figure 5 shows the transponder architecture block diagram, where the architecture main modules are apparent. The next Sections discuss each of these modules.

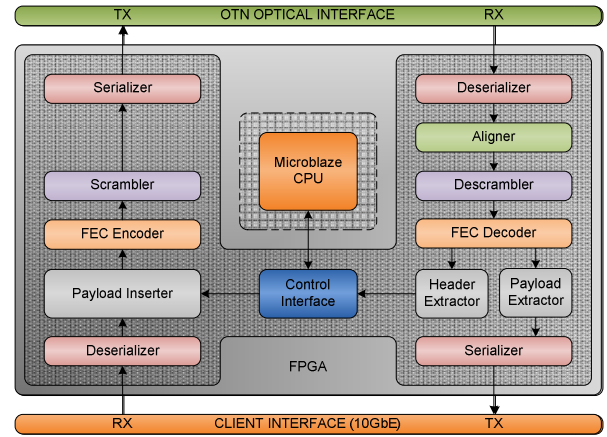


Figure 5 – Block diagram for the proposed OTN transponder architecture.

A. High-speed OTN and Ethernet optical interfaces

The proposed architecture uses 4 serializer/deserializer modules, two at each external interface connections (OTN optical interface and 10 Gbps Ethernet interface). These modules employ specific Xilinx FPGA primitives devised for double data rate (DDR) communication. These components include internal devices such as DDR registers and differential buffers (IDDR/IBUFDS and ODDR/OBUFDS pairs). More information about these devices and their use is available in [5]. Combined with the external interfaces, these modules are responsible for receiving the serial signal from the optical interface or the 10 GbE interface and then send it, in parallel, to the FPGA.

The optical signal is received at approximately 10 Gbps using 16 channels operating at 669.327 Mbps as determined in the OTU2 standard. The conversion from 10 Gbps optical to 16 copper channels at 669.327 Mbps is produced externally to the FPGA. However, even this

lower frequency is unreachable in modern FPGA core logic. Each copper channel is thus transformed into four 167.332 Mbps channels, generating a parallel traffic, which operates with 64-bit words. This is the signal, which the FPGA processes. The converse process is also possible, and the FPGA can supply an OTN stream for the OTN frame transmission.

The operating frequencies mentioned here in fact set the basic timing constraints the OTN circuit must respect as a whole. The parallelization of the optical flow sets the need to work with 64-bit words at approximately 200MHz. Since many bit manipulations take place during OTN processing, large circuit architectural challenges arise during the design. The next Sections cover how the design process faced and overcome the main challenges.

B. Frame Aligner

The frame aligner module is responsible to identify the FAS (Frame Align Sequence) sequence. The FAS includes the 6 first bytes of a multiframe. Figure 6 depicts the proposed architecture. This module has to handle 64-bit word bursts at 167.332MHz, and the FAS sequence may start in any of the 64 bits of some received word.

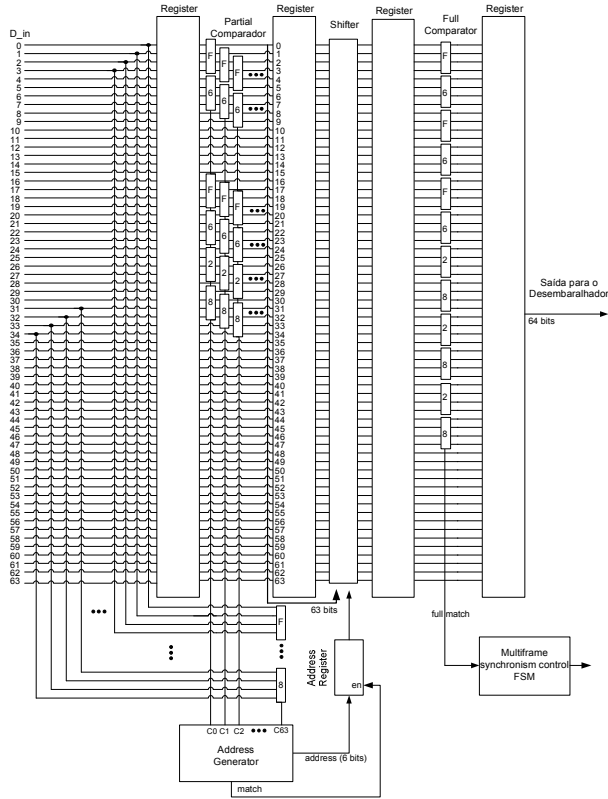


Figure 6 – The frame aligner circuit pipeline.

The aligner module is divided into sub-modules and organized as a pipeline structure, with a delay constraint of 5 ns. The first stage of the pipeline consists of various partial comparators and an address generator. Partial comparators can identify the alignment sequence. They are partial since this structure replicates itself 63 times. The

address generator identifies in which bit of the incoming 64-bit word the alignment sequence starts.

The second to seventh pipeline stages implement a logarithmic shifter, responsible to perform the word displacement in such way that the alignment sequence position itself starts in the beginning of the 64-bit word. The eighth pipeline stage has a complete comparator, used to prevent false positives of the alignment sequence and to feed the synchronization finite state machine. This finite state machine informs when this module finds the correct alignment of the incoming OTN signal, information that is used by the remainder of the circuit.

C. Scrambler

The scrambler module is responsible to scramble the data the framer transmits, using an LFSR pseudo-random data generation technique. This technique is used to avoid the transmission of long sequences of “0”s or “1”s. The scrambling process operates on the overall OTN G.709 multi-frame, with the exception of the FAS field. The unscrambling functionality is the same of the scrambling, because this process is symmetric.

As occurred for the aligner, the use of a straightforward LFSR is not possible to construct the scrambling system, due to its delay (eight clock cycles to treat a 64-bit word). To meet the delay constraints, a memory block with the contents of the generated LFSR values is used, replacing the traditional LFSR structure. In this new architecture, all possible polynomial scrambled sequences are stored in memory blocks (BRAMs), and since the OTN standard polynomial order is 16, there are 65535 ($2^{16}-1$) pseudo-random bits. In fact, only four 1024x16 BRAMs are required to store the complete sequence, organized as Figure 7 shows.

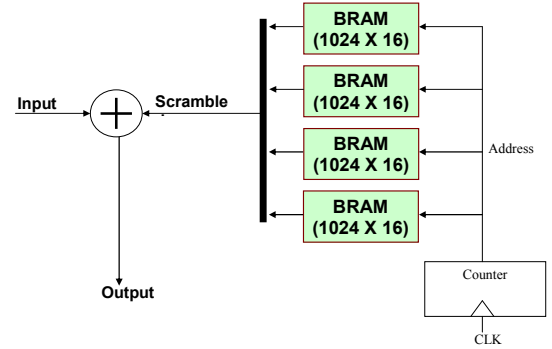


Figure 7 – Scrambler architecture using BRAMs.

D. FEC

The forward error correction (FEC) module uses the Reed-Solomon (RS) error correction method to introduce redundant information into the OTN frame. The receiver employs this additional information to search and correct errors, which may appear due to the transmission process [6]. The FEC encoder architecture is depicted in Figure 8.

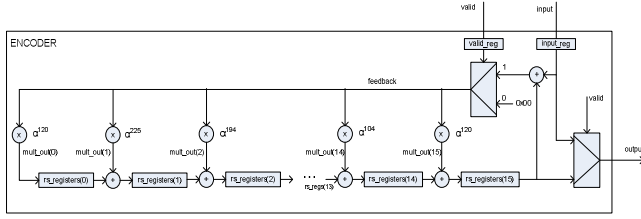


Figure 8 – The developed FEC encoder architecture.

Each line of the OTN frame generates one FEC parity information (Figure 4), but frame transmission takes place column-by-column (Figure 2). This transmission method presents a problem for the FEC error correction module, since in each 64-bit word received there are bytes from 8 different columns. It would be necessary to have a context retrieval capability for each byte of the FEC, to treat each byte separately. Again, due to the imposed design timing constraints, sixteen copies of the FEC encoder are present in the proposed OTN framer.

A more complex module is the FEC decoder, responsible to detect and correct errors. For the same reasons presented for the encoder, the decoder also appear replicated 16 times. Figure 10 depicts the FEC decoder architecture. The detailed description of the internal modules is out of the scope of this paper and is subject of extensive discussion in references [6], [7] and [8]. Because the FEC decoder is quite large, its area and timing optimizations were important to achieve FPGA area restrictions for the whole framer design.

E. 10 Gbps Ethernet client interface

The 10 Gbps Ethernet client interface is responsible to deliver and receive data to/from a 10 GbE client. The client interface interacts two major components as depicted in Figure 9, the *client drop interface* (corresponding to the Payload Extractor in Figure 5) and the *client add interface* (corresponding to the Payload Inserter in Figure 5).

The *client drop interface* is responsible to receive the processed OTN frame, retrieve its payload and deliver it to the 10 GbE interface. This module analyzes the justification fields of the OTN frame to determine if there is justification on the multiframe. If there is a justification, positive or negative, the client drop interface performs a displacement of the 64-bit word in order to align it with the

OTN payload when needed. The OTN payload feeds an asynchronous FIFO, which then transmits data to the 10 GbE client. This module is also responsible to generate a reference signal for external jitter control.

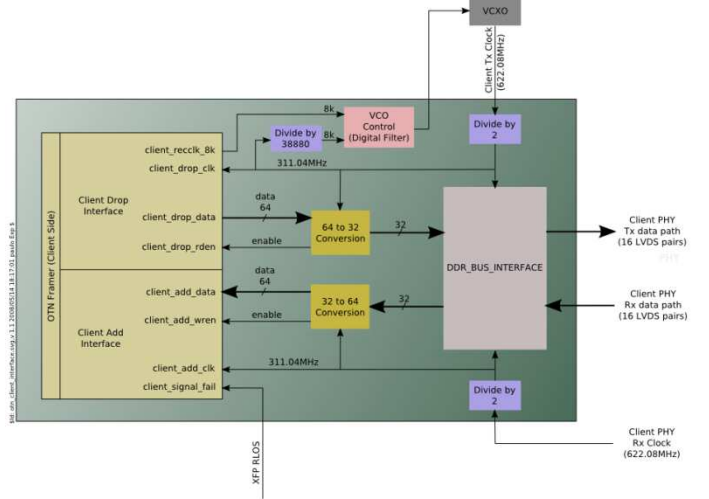


Figure 9 – Client interface block diagram.

The *client add interface* is responsible to deliver a constant dataflow stream to the optical interface at an operating frequency of 161.13 MHz. This module uses an internal FIFO, to avoid frequency variations due to the signal jitter inserted by the 10 GbE client side. The *client add interface* is also responsible to introduce positive or negative justification bytes, thus controlling jitter.

F. Control Interface Subsystem

The control interface subsystem is responsible to browse the OTN multiframe header and report any alarm contained on it to the system processor, the Xilinx firm core Microblaze. Also, it receives configuration commands from the processor. The control interface subsystem consists in 3 main modules: the Microblaze CPU, the header extractor and the control interface itself, containing the system register bank. The header is the area of the OTN multiframe carrying the OTN communication control bytes. Figure 11 shows the OTN header. The header extractor module processes OTN multiframe header bytes and writes information about them to the register bank.

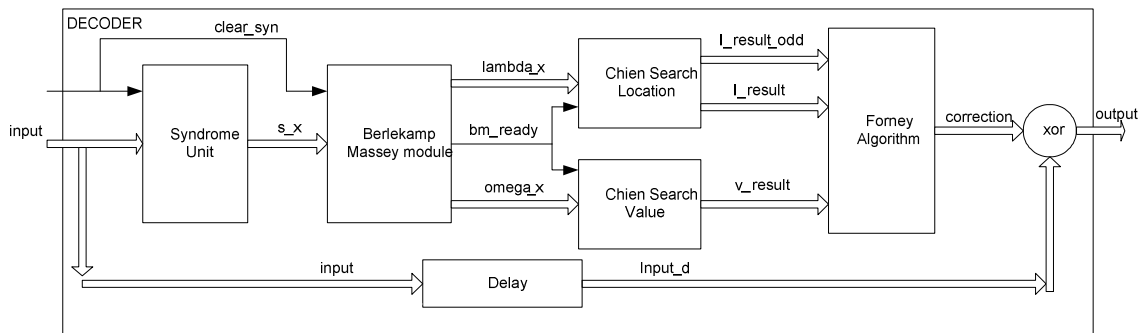


Figure 10 – Developed RS (255,239) decoder architecture.

The Microblaze CPU is the system control processor. It runs the OTN framer software, which has the responsibility to manage all system alarms and transmission operation, using an embedded Linux operating system. The system control processor manages the OTN framer through read/write operations into the system register bank.

The system register bank contains all OTN multiframe header bytes, the FEC module statistics, transmission control information and some system variables.

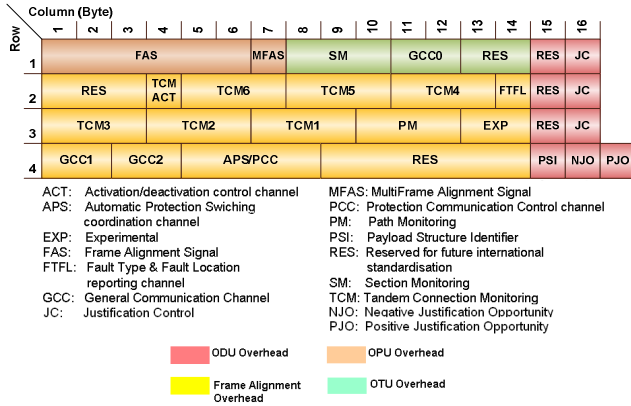


Figure 11 – The OTN multiframe header bytes [9].

IV. DESIGN VALIDATION

This Section presents the OTN framer design validation process and the structures employed for this process. The OTN framer was simulated and fully prototyped in FPGAs.

A. Simulation Environment

The framer simulation environment allows testing and evaluating all external system interfaces and the framer correct functionality. The framer simulation environment is depicted in Figure 12 and explained next. The simulation testbench instantiates the framer hardware, a module that simulates the behavior of the system control processor (CPU software simulator) and a loopback connection in place of the 10 GbE interface. The testbench is also responsible to send simulated OTN frames to the optical interface and to store frames transmitted by the framer hardware to the optical interface.

The simulated frames are generated by an tool called “geraframe”, and are available at the “input.x10” file. This tool can be configured to generate controlled frames for specific tests, building valid OTN multiframe header bytes and introducing frame or scrambling errors. The testbench stores any received frames into the “output.x10” files for later evaluation.

The CPU software simulator enables the emulation of read and write operations into the system register bank, as would be produced by Microblaze running Linux. The “instructions.x10” file defines these operations, and a log file generated by the testbench stores results of read

operations. Figure 13 presents the appearance of a log of sequential read and write operations in a set of registers.

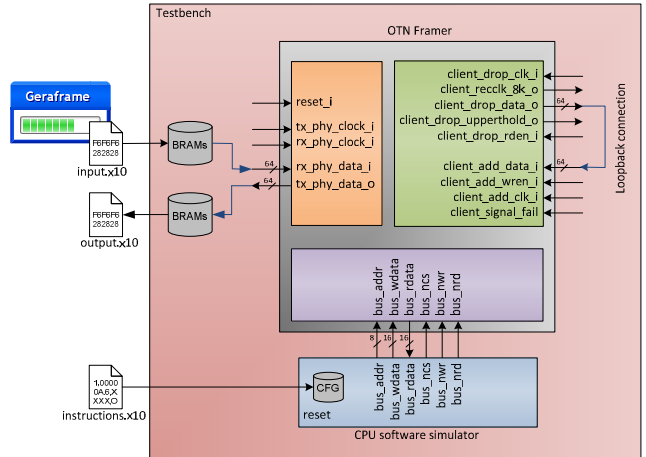


Figure 12 – OTN framer simulation environment.

```

56268.793 ns: READ TEST OK
Valor no registrador SM_BEI_CNT_ADDR      = 0000 0000 0000 1000 (0x8)
Valor esperado                             = 0000 0000 0000 1000 (0x8)

58630.793 ns: READ TEST OK
Valor no registrador PM_BEI_CNT_ADDR      = 0000 0000 0000 1000 (0x8)
Valor esperado                             = 0000 0000 0000 1000 (0x8)

212238.793 ns: READ TEST OK
Valor no registrador ODU_PM_STS_ADDR      = 0000 0000 1110 0000 (0xE0)
Valor esperado                             = 0000 0000 1110 0000 (0xE0)

215787.793 ns: READ TEST FAIL
Valor no registrador OTU_FRAME_CNT_ADDR   = 0000 0000 0000 1001 (0x9)
Valor esperado                             = 0000 0000 0000 1010 (0xA)

-----
Results
Performed Tests = 4

Tests Passed = 3
Tests Failed = 1
  
```

Figure 13 – Example of a CPU simulator log file, used to store test results performed by the simulation environment. Each entry contains the value read from the register, followed by the expected register value.

The generated simulation input frames contain one or more test conditions, e.g. header alarms, FEC error insertion or system register read/write operations. The set of test frames containing a number of pre-selected test conditions define a “simulation test scenario”. While performing OTN framer simulation, two main simulation scenarios were created: one that generates and evaluates all possible OTN header alarms and another that evaluates the framer internal modules functionality.

B. FPGA Prototyping

In order to verify the correct functionality of the developed hardware, all modules of the project were synthesized, prototyped and validated on FPGA. Evaluated metrics include FPGA area occupation (slices, memory, etc.) and timing constraints.

Two different Xilinx FPGA devices served as target during the synthesis process: a Virtex-4 FX100-11 and a Virtex-5 LX220T-2. The project was synthesized first in a Virtex-4 device to quickly evaluate design functionality in a prototyping board, the Dinigroup DN8000K10PCI board. The synthesis for the Virtex-5 device came later, when the specific board prototype of the envisaged commercial product became available.

The Virtex-4 synthesis results shows a minimum period of 4.736 ns (i.e. a maximum operating frequency of 211.149 MHz) and an area occupation of 78% of the FPGA available slices. The Virtex-5 synthesis results shows a minimum period of 3.413 ns (i.e. a maximum operating frequency of 292.985 MHz) and an area occupation of 44% of the FPGA available slices.

For the Virtex-4 evaluation, the framer prototype was validated using a communication structure called MainBus [10], which is a bus architecture developed by the Dini Group to facilitate data transfers between FPGA and a host PC through the PCI bus. The host PC sends and receives test frames to the framer hardware, comparing the results. As in the simulation environment, the external connections of the 10 GbE clients are connected via loopback link.

As for the Virtex-5 prototyping, the proposed OTN framer is part of a DATACOM product prototype board (Figure 14), which contains an embedded Virtex-5 LX220T-2 FPGA. The final OTN framer logic is currently under test using a JDSU optical network tester with an OTN testing module. Preliminary results show that the OTN framer implementation is correct, the client interface being currently under test.



Figure 14 – PUCRS/DATACOM OTN framer prototype board.

V. CONCLUSIONS AND ONGOING WORK

This work shows that FPGAs are effective in creating complex systems with tight performance requirements. Besides the intrinsic complexity of the design, and the strict rules defined in the OTN definition, timing closure was one of the major design challenges. Simple modules, such as the frame aligner, required a VHDL gate level description, with several pipeline stages. The FEC decoder design process evaluated several different Galois multiplier architectures before meeting the critical path constraint (5 ns). In addition, due to the high-speed incoming stream, several modules had to be replicated, increasing the final FPGA occupation.

This system is currently a functional prototype. The final PCB board with Virtex-5 FPGAs was fabricated, and tests in the final product are ongoing activities. As a future work, the system will be extended to include other client mappings, not only 10 Gbps Ethernet.

ACKNOWLEDGMENT

This work receives support from the Brazilian Funding Agency FINEP, under project number 5161/06.

REFERENCES

- [1] ITU-T “G.870: Terms and definitions for optical transport networks (OTN)”. Available at: <http://www.itu.int/rec/T-REC-G.870-200803-I/en>, Apr. 2009.
- [2] Xelic XCO2 OTN transponder, captured at: http://www.xelic.com/Networking_Cores/product_brief/xco2_product_brief.pdf, Apr. 2009.
- [3] Broadcom BCM8512 OTN transponder, captured at: <http://www.broadcom.com/collateral/pb/8512-PB05-R.pdf>, April 2009.
- [4] Kocialski, C. and Harwood, J., “A Primer on Digital Wrappers for Optical Transport Networks”. Vesta Corporation, 2000.
- [5] Virtex-5 FPGA User Guide, available at: http://www.xilinx.com/support/documentation/user_guides/ug190.pdf, Apr. 2009.
- [6] Silva, A. and Rodolfo, T. “Implementatin of a Reed-Solomon architecture for use in OTN 10.7 Gbps Networks”. End of term work, Computer Engineering, PUCRS, Dec. 2007. (In Portuguese)
- [7] Wilhelm, W. “A New Scalable VLSI Architecture for Reed-Solomon Decoders”. IEEE Journal of Solid State Circuit, Vol. 34, No 2. Mar. 1999, pp. 388-396.
- [8] Park, T. “Design of the (248,216) Reed-Solomon Decoder with Erasure Correction for Blu-ray Disc”. IEEE Transactions on Consumer Eletronics, Vol. 51, No 3. Aug. 2005, pp. 872-878.
- [9] Vissers, M. “Optical Transport Network & Optical Transport Module”. Captured at: <http://ties.itu.ch/ftp/public/itu-t/tsg15opticaltransport/OTN/g709-intro-v2.ppt>, Apr. 2009.
- [10] The Dini Group. “MainBus Specification”. Available at: http://www.dinigroup.com/product/common/mainbus_spec.pdf, Apr. 2009.