Felipe Calliari

Automatic High-Dynamic and High-Resolution Photon Counting OTDR for Optical Fiber Network Monitoring

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétricada PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

Advisor: Prof. Gustavo Castro do Amaral

Rio de Janeiro
July 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Calliari

Automatic High-Dynamic and High-Resolution Photon Counting OTDR for Optical Fiber Network Monitoring

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétricada PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica. Approved by the undersigned Examination Committee.

Rio de Janeiro, July 7th, 2017

Felipe Calliari

Felipe Calliari graduated from the Pontifícia Universidade Católica do Rio de Janeiro (Rio de Janeiro, Brasil) in Electrical Engineering with both Computer Electronics and Telecommunications emphases. He is a member of the Optoelectronics Laboratory of Center for Telecommunication Studies in PUC-Rio.

Bibliographic data

Calliari, Felipe

Automatic High-Dynamic and High-Resolution Photon Counting OTDR for Optical Fiber Network Monitoring / Felipe Calliari; advisor: Gustavo Castro do Amaral. – Rio de janeiro: PUC-Rio, Departamento de Engenharia Elétrica, 2017.

v., 105 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Comunicações Ópticas;. 3. Monitoramento de Fibras Ópticas;. 4. Reflectômetro Óptico no Domínio do Tempo;. 5. Optoeletrônica.. I. Amaral, Gustavo Castro do. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Acknowledgments

I would like to thank Beatriz Mynssen, my wife and the love of my life, for her dedication, patience and affection.

To my friend and advisor Gustavo Amaral for the support and teachings along all these years that we have met.

To my colleagues and friends of PUC–Rio, specially L. E. Y. Herrera who was always glad to help me out in this work.

To all my family, specially my grandmother, Neiva Maria Calliari.

I would also like to thank to the brazilian agency CNPq for the financial support, without which this work would not have been realized.

Abstract

Calliari, Felipe; Amaral, Gustavo Castro do (Advisor). **Automatic High-Dynamic and High-Resolution Photon Counting OTDR for Optical Fiber Network Monitoring**. Rio de Janeiro, 2017. 105p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

In this work the development of an automated structure for the monitoring of optical fibers is presented. This structure consists of two types of Photon Counting Optical Time Domain Reflectometers and a trend filter that is used to detect fiber faults in an automated way. The first Photon Counting OTDR has a 32 dB dynamic range with spatial resolution of 6 m, while the second OTDR has a 14 dB dynamic range and a resolution of 3 cm. Its ability to automatically detect faults in an optical fiber link and tunability for monitoring of optical WDM networks has been demonstrated.

Keywords

Optical Communications; Optical Fiber Monitoring; Optical Time Domain Reflectometry; Optoelectronics.

Resumo

Calliari, Felipe; Amaral, Gustavo Castro do (Orientador). **Sistema de Monitoramento Automático de Fibras Ópticas por OTDR de Contagem de Fótons e modos de Alta-Resolução/Alto-Alcance**. Rio de Janeiro, 2017. 105p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Neste trabalho é apresentado o desenvolvimento de uma estrutura automatizada para o monitoramento de fibras ópticas. Esta estrura consite em dois tipos de reflectômetros ópticos por contagem de fótons no domínio do tempo e um filtro de tendências que é utilizado para detectar as falhas em uma fibra óptica de forma automatizada. O primeiro OTDR por contagem de fótons apresenta uma faixa dinâmica de 32 dB com resolução espacial de 6 m, já o segundo OTDR apresenta uma faixa dinâmica de 14 dB e uma resolução de 3 cm. Foi demonstrada a sua capacidade de detectar falhas automaticamente em um enlace óptica e de sintonização no monitoramento de redes passivas WDM.

Palavras-chave

Comunicações Ópticas; Monitoramento de Fibras Ópticas; Reflectômetro Óptico no Domínio do Tempo; Optoeletrônica.

Table of contents

List of figures

List of tables

List of Abreviations

ADC	Analog-to-Digital Converter
APD	Avalanche Photodiode
API	Application Programming Interface
ASE	Amplified Spontaneous Emission
BPF	Band-Pass Filter
BWTF	Bandwidth and Wavelength Tunable Filter
CCLs	CMOS Configuration Latches
CLB	Configurable Logic Blocks
CMOS	Complementary Metal-Oxide Semiconductor
CRN	Coherent Rayleigh Noise
CW	Continuous Wavelength
DAC	Digital-to-Analog Converter
DDG	Digital Delay Generator
DLL	Dynamic-Link Library
DS	Dispersion-Shifted Fiber
EDFA	Erbium-Doped Fiber Amplifier
FPGA	Field Programmable Gate Array
FUT	Fiber Under Test
G-APD	Geiger Mode APD
GPIB	General Purpose Interface Bus
GUI	Graphical User Interface
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input and Output
ISP	Instruction Set Processor
IOB	Input and Output Blocks
JTAG	Joint Test Action Group
KARL	Kaiserslautern Register Transfer Language
LASER	Light Amplification by Stimulated Emission of Radiation
LUT	Look-Up Table
MMF	Multi-Mode Optical Fiber
NA	Numerical Aperture
OSA	Optical Spectrum Analyzer

OTDR	Optical Time Domain Reflectometer
PCI	Peripheral Component Interconnect
PXI	PCI eXtensions for Instrumentation
PON	Passive Optical Network
RAM	Random Access Memory
RTL	Register Transfer Level
SMF	Single-Mode Optical Fiber
SOA	Semiconductor Optical Amplifier
SPAD	Single Photon Avalanche Detector
STD	Standard Fiber
TAC	Time-to-Analog Converters
TDC	Time-to-Digital Converter
TLS	Tunable Laser Source
TTL	Transistor-Transistor Logic
UWS	Ultra Wideband Laser Source
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VISA	Virtual Instrument Software Architecture
VSLI	Very-Large-Scale Integration
WDM	Wavelength-Division Multiplex
ν-OTDR	Photon Counting OTDR

I want to know God's thoughts, the rest are details.

Albert Einstein.

1 Introduction

Ever since mankind began to live in society, the man began to feel need to communicate with each other, whether to warn about something or to express his culture or feeling. As Anu Gokhale states, in his book Introduction to Telecommunications [1]:

> "Communication has always been an integral part of our lives. Family relations, education, government, business, and other organizational activities are all totally dependent on communications. It is such a commonplace activity that we take it for granted. Yet, without communications most modern human activity would come to a stop and cease to exist. To a great extent, the success of almost every human activity is highly dependent on how the available communications methods and techniques are effectively utilized."

Initially, the man began to communicate by gestures and sounds, and with time the cave paintings appeared. Next, writing emerged as a development necessity of the society. The first writing systems developed were based on ideographic or mnemonic symbols, this type of writing is described as a proto-writing and cannot be classified as "proper writing".

Writing evolved from pictographic and ideographic forms into phonetic form. The pictographic writing used a drawing to represent a word, this type of writing was the basis for cuneiform writing (Sumerians) and hieroglyphs (Egyptians) and is used today, for example, in traffic signs, etc. Ideographic writing has probably emerged as an evolution of pictographic writing and consists of "ideograms", symbols or drawings representing objects, ideas or words, an example of ideographic writing is the Japanese writing system kanji. It is important to note that the Japanese use three writing systems hiragana, katakana and kanji, all of which are used at the same time.

Phonetic writing is a writing system that is based on the representation of speech sounds. It is important to remember that Sumerian writing, although rooted in pictographic writing, evolved to the point where it combined elements of phonetic and pictographic character.

1.1 What is Telecommunication?

The word telecommunication is defined as the science or technology of distance communication and has its origins in two words *tele* from Greek, meaning “far, distant”, and *communicare* from Latin, meaning “communicate, share or make common” [1, 2].

Communication is the process by which information generated at one point in space and time, called source, is transferred to another point in space and time, called destination. Therefore, it is implicit, when we speak in telecommunications, that there is a sender, a receiver, a medium and, most importantly, the message itself.

Currently, the term telecommunications is widely used to refer to the means of transmitting information over long distances, by means of electrical or electronic devices and radio waves or fiber optics.

1.2 History of Telecommunication

Since prehistory, man tries to communicate at a distance through smoke signals, drums, etc. Through the use of these primitive means of telecommunication, the men were able to contact neighboring clans and warn them of the presence of enemy clans, hazards, or send appeals for help. A timeline of the major developments in telecommunications is show in Tab. 1.1.

Year	Major Development
before 3000 BCE	smoke signals, communication drums, horn
3000 BCE	papyrus
2400 BCE	couriers, first postal systems
500 BCE	heliograph, pigeon post
400 BCE	hydraulic semaphores
105 CE	paper
1440 CE	first European printing presses
1672 CE	first acoustic string phone
1790 CE	semaphore lines
1838 CE	electrical telegraph
1876 CE	telephone
1896 CE	radio
1927 CE	television
1962 CE	commerical telecommunications satellite
1964 CE	fiber optical telecommunications
1969 CE	computer networking

continued from previous page

Year	Major Development
1983 CE	internet

Table 1.1: Timeline of Telecommunication [1,3].

Mail

In Egypt, the postal systems were used for the dissemination of Pharaoh's written documents and decrees. It is estimated that the mail system was used since 2400 BCE, although the earliest known correspondences dates back to 1350 BCE, the Amarna letters [4].

It is given credit to Cyrus the Great (550 BCE) to the invention of the first real postal system, although the beginning of mail remains unknown. Cyrus the Great ordered that each province of the Persian empire should arrange reception and delivery of post to each of its citizens. In China, the first credible postal system came up around 206 BCE and had relay stations every 15 km. Until today, the postal system has played an important role in the transport of documents and packages.

Pigeon Post

It is well known that the Persians and Romans wore post pigeons around the 5th century BC. Its use is based on the pigeons' natural ability to return home. Post pigeons were often used as military messengers because of their homing instinct and speed, they were even used during the first and second world wars. The pigeons were transported in cages and, when necessary, messages were tied in their paws. After World War II they fell into disuse.

Hydraulic Telegraph

The word telegraph has its roots in two Greek words *tele* and *graphein* which means "to write", so telegraph means "to write at distances". The hydraulic telegraph was developed in the 4th century BC, in Greece. This system was first described by Aeneas Tacticus and was based on fire signals (semaphoric fires) and hydraulic pressure, see Figure 1.1. The operation of a hydraulic telegraph is described below.

To begin with, the sender lifts a torch and waits for the receiver to lift a torch, meaning that he's is ready to receive the message. When the receiver

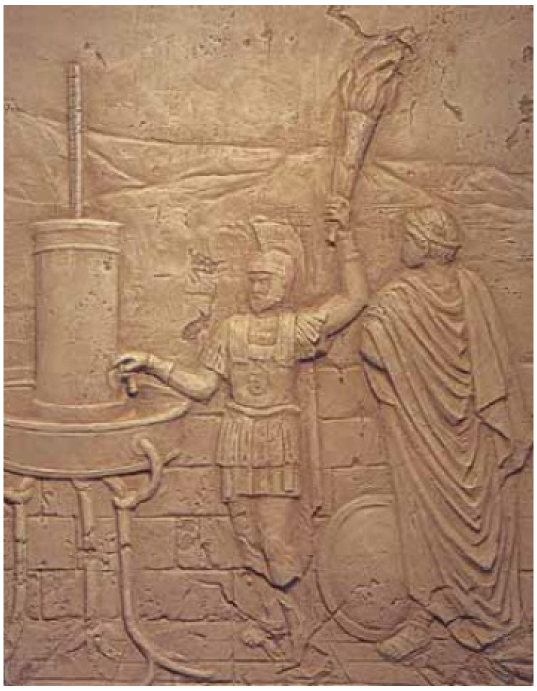


Figure 1.1: Ancient hydraulic telegraph design described by Aeneas Tacticus. 🄍 Demetre Valaris, public domain.

lowers his torch, both begin to drain their vessels through a hole in the bottom of equal size. Then the sender lifts his torch to signal that both should stop the flow of water. As a result, the water levels in both vessels are ideally the same, denoting a shared message. The water level inside the vessel was then associated with predetermined codes and messages.

One of the major disadvantages of this system was its limitation due to the speed with which a message was transmitted. It is believed that this system had been used during the First Punic War to send messages between Sicily and Carthage.

Acoustic string phone

A tin can telephone consists of two cans joined by a wire, which when stretched allows the transmission of sound waves through the wire's vibration. Nowadays, or even a few years ago, a tin can telephone was considered a children's toy.

The invention of the first acoustic string telephone is attributed to Robert Hooke due to his experiments carried out, from 1664 to 1665, in sound transmission through a distended wire. The first electromagnetic phones worked relatively poorly, which made the acoustic string telephones represented a viable alternative for relatively small distances for a few years. According to the text *Mechanical or String Telephones* of *Telefoniemuseum Rotterdam* [5]:

"(...) Truthfully though, very early telephones performed poorly; and during these years, the acoustic telephone represented a truly viable alternative for relatively short, private-line telephone systems. Since they contained no electrical transmitting or receiving devices, they did not infringe on the Bell patents. Thus they were able to enter the telephone industry during the protected years of the late 1870s and 1880s, carving out a small niche for themselves. (...) Although ordinary line wire could be used, it was common for manufacturers to recommend special types of wires. Typically it would possess high tensile strength, to minimize stretching and breaking. Many were galvanized to combat corrosion."

Optical Telegraph

The Optical Telegraph or Semaphore Line was invented in 1792 in France by Claude Chappe, who also coined the word telegraph. It is an information transmission system that is performed through towers with black movable wooden arms whose positions indicate letters of the alphabet. Once built, semaphore towers had low operating costs in the long run. By using several lines of relay towers it was possible to transmit information over long distances in a short period of time compared to post riders.

Different models of telegraphs have been built in several countries throughout history but their basic operating principle remained practically identical. Figure 1.2 shows a semaphore tower.

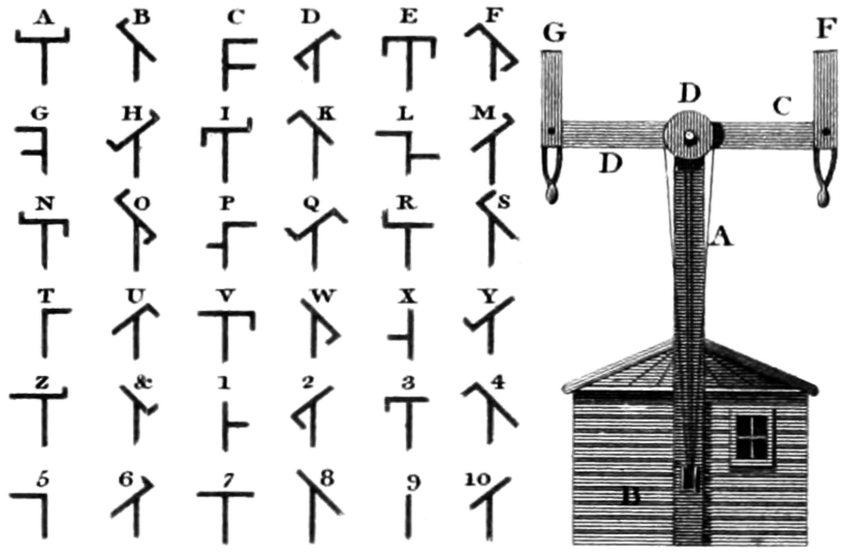


Figure 1.2: Diagram showing the Chappe's Optical Telegraph [6].

Morse Code and Electric Telegraph

The electric telegraph was the first form of transmission of messages through an electrical system. Several inventors attempted to create the electric telegraph, but the first practical system was developed and patented by Samuel Morse in 1837. Samuel Morse and Alfred Vail, his assistant, also developed the forerunner of the current Morse code. Morse code consists of a series of on-off tones representing the alphabet. The rules of the Morse code are: a dash is equal to three dots, the space between the signals forming the same letter is equal to one dot, the space between two letters is equal to three dots and the space between two works is equal to seven dots [7]. Table 1.2 shows the representation of the alphabet and numbers in Morse code.

Letter	Code	Letter	Code	Letter	Code	Number	Code
a	•-	k	-•-	u	••-	1	•----
b	-•••	l	•-••	v	•••-	2	••---
c	-•-•	m	--	w	•--	3	•••--
d	-••	n	-•	x	-••-	4	••••-
e	•	o	---	y	-•--	5	•••••
f	••-•	p	•--•	z	--••	6	-••••
g	--•	q	--•-			7	--•••
h	••••	r	•-•			8	---••
i	••	s	•••			9	----•
j	•---	t	-			0	-----

Table 1.2: International Morse Code [7].

The advent of the telegraph allowed the transmission of messages between continents almost instantaneously, which generated strong social and economic impacts in his time. The electrical telegraph superseded the optical telegraph, but with the dissemination of the telephone, the telegraph also fell into disuse.

Telephone

The invention of the telephone is generally attributed to Alexander Graham Bell and his assistant, Thomas A. Watson. However, in 2002 the United States Congress recognized that the telephone was invented by the Florentine inventor, Antonio Meucci. Its principle of functionally consisted in turning the speaker's voice into electrical signals, by means of a transducer, that could be sent over long distances. The invention of the telephone marked a significant development of telecommunications systems.

Radio

Guglielmo Marconi, an Italian inventor, is often credited as the inventor of the radio because of his pioneering work on radio transmission. The Marconi experimental device was capable of operating the transmitter to send short and long pulses (Morse code). Reginald Aubrey Fessenden, a Canadian inventor, perfected Marconi's radio and was the first to transmit the sound of the human voice through radio waves in 1900. It's important to note that the radio was the first wireless transmitting system utilizing electromagnetic waves.

Modern Era

In the twentieth century, a revolution in wireless communication began. After the invention of radio, several technologies emerged including television, the computer, satellite communications, optical fibers, etc.

1.3 Communication Systems

A communication system is formed by a set of elements that allows the transmission of information from one point to another [8]. The most basic communication system that exists is composed of a transmitter, a transmission medium, a receiver and a source of information. However, a communications system can be quite complex involving, see Fig. 1.3, in addition to the elements already mentioned, a step of encoding, modulation, demodulation and decoding.

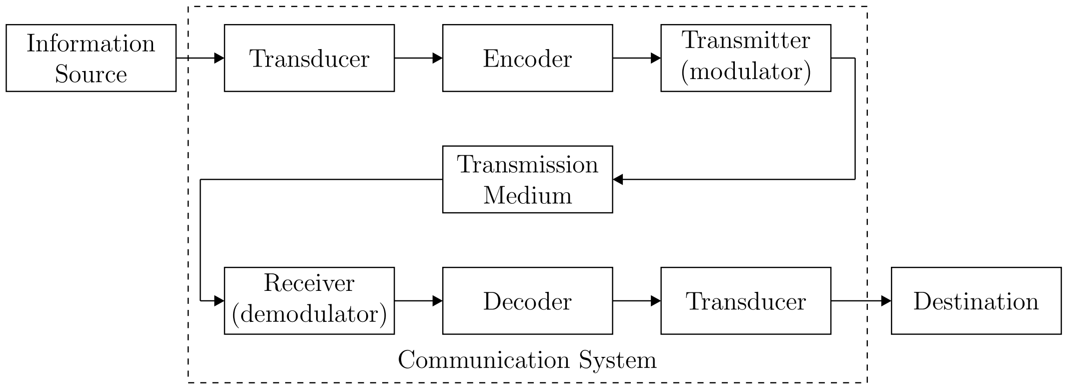


Figure 1.3: A generic communication system.

A transducer is a device which converts one form of energy into another form of energy [9]. Examples of transducers are loudspeakers, microphones, thermocouples, among others. A loudspeaker, for example, is a device that converts electrical energy into an acoustical signal energy. The encoder is a device that converts information from one format to another and its main function is to add redundancy to the transmitted signal to avoid errors caused by noise. The decoder, on the other hand, does the reverse process and, by removing redundancy, retrieves the transmitted signal. A transmitter is an electronic device that converts the input signal into another signal, by means of modulation, to make the signal suitable for the communication medium, then the signal is capable of propagating with the least distortion possible. As the transmitted signal propagates through the transmission medium, it is corrupted due to the added noise and distortions suffered as it propagates. The receiver tries to reverse the process by reconstructing the signal into a recognizable form [8, 10].

1.3.1 Optical Fiber Communication Systems

An optical fiber communication system is very similar to the communication system shown in Fig. 1.3. However, the transmitter block consists of two basic components: a light emitting circuit and a driver circuit. In optical fiber communications, the light emitting circuit is responsible for converting the electrical signal into an optical signal, while the driver circuit has the function of electric polarization control and optical power control. The transmission medium is composed of optical fibers, in which the light travels. The receiver block has the inverse function of the transmitter block, that is, it detects the optical signal and converts it to electric. The receiver consists of a photodetector, which performs optoelectric conversion, and an amplifier circuit.

Optical fiber communications systems began to be used only in the mid-1970s when K. C. Kao and G. A. Hockham proposed the use of a clad glass fiber as a dielectric waveguide [8]. Although the first optical fibers presented attenuation of the order of 1000 dB/km, Kao and Hockham predicted that the attenuation could be reduced below 20 dB/km. The transmission capacity of the coaxial cables reached its limit in the 80's, which caused their gradual replacement by the optical fibers. Among the main advantages of using fiber optics are its low attenuation (0.2 dB/km), its broad bandwidth and its immunity to electromagnetic interference. Nowadays, optical communication systems are the standard technology in many applications, including data transmission, voice, video and telemetry [11].

1.4 Motivation

One of the most important elements in telecommunication networks is the fiber optics. Optical fibers have many advantages over other transmission methods (cables, satellites, etc.). Some of the advantages of using fiber optics are: its ability to carry information, its immunity to electrical or magnetic interference, its weight in relation to metallic cables, its manufacturing raw material is silica, and therefore its low cost.

Although the optical fibers are very reliable, they can sometimes be damaged. The causes are the most varied: in the ocean, for example, ships can break optical fibers, sharks and other marine animals can chew the protection of the optical fibers, among others. On land, the optical fibers can be broken due to works, storms, etc.

One way of discovering where a failure occurred in communications systems using fiber optics is to use a Rayleigh scatter-based monitoring system. There are some types of techniques that make use of this phenomenon, but what stands out most certainly is the OTDR (Optical Time Domain Reflectometer). Through the use of an OTDR it is possible to extract information such as: bending losses, losses due to the use of connectors, losses due to improper welding, etc., by accessing only one end of the fiber. This means that a central transmission station can monitor all the fibers connected to it *in loco* without the need to install an apparatus for monitoring in each of the multiple nodes of the optical network. A good quality OTDR offers both good spatial resolution (less than 1 meter) and long range (greater than 100 kilometers).

The objective of this work is to develop a hybrid structure of Photon Counting OTDR with very high dynamic range and very high resolution. Using two independent monitoring structures and a technique of automatic identification of fiber optic faults in OTDR traces, it is intended to multiplex both allowing a rapid and accurate characterization of faults in optical fibers.

2
Basic Review

2.1
FPGA

The Field Programmable Gate Arrays (FPGA) is a programmable device composed of a large number of Configurable Logic Blocks (CLB), small cells that contain the basic structures of digital electronics, Input and Output Blocks (IOB), circuits responsible for implementing the input and output functions of an FPGA, and Switch Matrix, responsible for the connection between the CLBs internally in the FPGA and the connection between the CLBs and the IOBs. By programming the switch matrices that connects these blocks, it is possible to create any type of digital circuit.

To implement some logical structure in an FPGA we use some type of software that is able to generate a configuration file that contains the connections between the logical blocks. This configuration file is generated through a hardware description language (HDL) and contains all the information on how the blocks are connected internally in the FPGA.

The first FPGAs arose in the mid-1980s, the first commercially viable FPGA was created by Xilinx, Inc. in 1985, the XC2064 had only 64 Configurable Logic Blocks [12]. In the 1990s FPGAs began to evolve and become more sophisticated and soon began to be used in telecommunication and network projects. In the late 1990s, FPGAs were also used by the automotive industry and industrial applications.

At the turn of the century, educational entities began to integrate FPGAs into their courses and vendors began to develop development boards. These development boards came with several integrated components, LEDs, buttons, USB ports, serial, etc.

2.1.1
What is an HDL?

The development of modules in FPGAs is done through hardware description languages. A Hardware Description Language (HDL) is a software programming language used to describe the structure and operation of electronic circuits and digital circuits.

The hardware description language allows us to create a precise and formal description of the electronic circuit to be analyzed [13]. In addition, with the use of an HDL, it is possible to carry out automated analyzes to verify design errors, as well as to perform simulations of digital circuits.

The first HDLs (Hardware Description Languages) emerged in the 1960s. One of the first widely used HDLs was the Instruction Set Processor (ISP) language. By the late 1970s, the first PLDs (Programmable Logic Devices) became popular, although there was a limitation on the size of circuits that could be implemented.

Later, in the mid-1980s, as new technologies emerged, the VLSI[1] was being developed, the need arose for a language that could handle such large circuits, so Kaiserslautern Register Transfer Language (KARL) emerged. The KARL language was developed by the University of Kaiserslautern in 1979. The first modern HDL was Verilog, created by Gateway Design Automation in 1985.

In 1987 the VHDL language was developed at the request of the US Department of Defense (DARPA) [13], in order to document and simulate circuits already implemented. The VHDL was created based on the ADA language and the experience acquired with the development of ISPS (successor of the ISP language). Currently, the VHDL language is standardized by the IEEE.

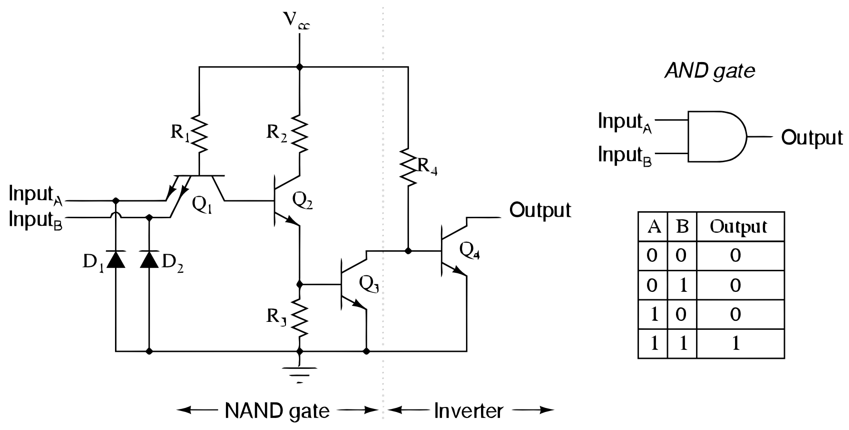
At this time the process of creating integrated circuits (VSLI) had already been developed and engineers were creating integrated circuits through schematic and they performed simulations with the help of the languages Verilog and VHDL.

The HDL languages gained, even more, popularity after the introduction of the logic synthesis step for HDL. That is, transforming the HDL code to Register Transfer Level (RTL), in order to refine the RTL code at the netlist level and to perform logical error checking steps, to then transform that RTL code into a design implementation in logical ports.

That is, the HDL language could be translated into a language of easy understanding for the designers through a schematic representation. Some of the most common and most used logic gates are AND, OR, NAND, NOR,

[1]VSLI — Very-large-scale integration technique is a process of creating integrated circuits by combining thousands of transistors into a single chip. Before the emergence of VSLI in the 1970s, integrated circuits had limited functions and could be implemented with few transistors, so it was necessary to use multiple ICs. With the emergence of the VLSI technique it was possible to integrate several ICs into a single chip.

XOR and NOT logic gates. The logic gates basically consist of transistors, hence the name TTL (Transistor-Transistor Logic). Figure 2.1 presents the schematic of a TTL AND port.



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Figure 2.1: TTL AND gates with open-collector output [14].

One of the problems encountered in the VHDL language is that writing synthesizable RTL codes requires practice and discipline on the part of the designer. Another important fact to mention is that if we were to compare the implementation of a large circuit by a qualified engineer using conventional methods (schematic circuit layout) and another using VHDL language, the circuit implemented by the traditional method would probably be faster.

However, we must take into account that there are already several optimizations built into the VHDL synthesizers and that, using programming "best practices", it is possible to create VHDL circuits with performance comparable to that of the circuits implemented by engineers using conventional methods.

Over the years, VHDL and Verilog have become the dominant HDLs in the electronics industry, but both languages have the following limitations: neither language is capable of describing analog or mixed circuits (with analog and digital signals).

The initial version of the standard VHDL was created in 1987 and defined by the IEEE standard IEEE 1076-1987. The standard VHDL version used in this project is defined by the IEEE document 1076-1993. In this release some of the modifications include a more consistent syntax, greater flexibility in naming, allowing the use of ISO-8859-1 printable characters, adding the IEEE 1164 standard, and more.

VHDL is used to describe logical circuits in the form of texts and, after passing through the stages of synthesis and simulation, where prototyping errors are verified and corrected, it is possible to use synthesis tools to translate the design into real hardware. Once a block is created, it can be reused in other projects without the need for modifications.

2.1.2 Configuring an FPGA

To program an FPGA it is necessary to load a file called *configuration bitstream* that defines the FPGA functionalities. Whenever there is any modification in HDL, it is necessary to create a totally new *configuration bitstream*. This need comes from the fact that during the synthesis process numerous optimizations are performed. Optimizations can be performed to achieve the highest clock frequency, decrease area usage, ie use the minimum FPGA resources, useful when the project is large, or reduce power consumption.

An FPGA is a highly flexible and reprogrammable device and can be reprogrammed, on demand, an unlimited number of times. Whenever we configure the FPGA, configuration bitstream is stored in highly robust CMOS configuration latches (CCLs) [12]. As stated in the Spartan-3 manual [15]:

> "Although CCLs are reprogrammable like SRAM memory, CCLs are designed primarily for data integrity, not for performance. The data stored in CCLs is written only during configuration and remains static unless changed by another configuration event."

The CCLs, as well as an SRAM memory, do not retain their data when power is removed. Therefore, the FPGA must be reconfigured every time it is turned off. The FPGA configuration process can be performed by an external smart source, such as a microprocessor, a PC or an external non-volatile memory device.

The *configuration bitstream* can be loaded into the FPGA through special configuration pins. The configuration methods can be:

- JTAG (Joint Test Action Group) interface
- Synchronous Serial interface

Being the JTAG interface, the most common way to program and debug an FPGA. The JTAG interface was developed by the Joint European Test Action Group and in 1990 was approved by the IEEE as a standard, IEEE Standard 1149.1-1990 [16].

2.1.3
Fundamental FPGA Blocks

The Spartan-3 family architecture, see Fig. 2.2, consists of five fundamental programmable functional elements [15]:

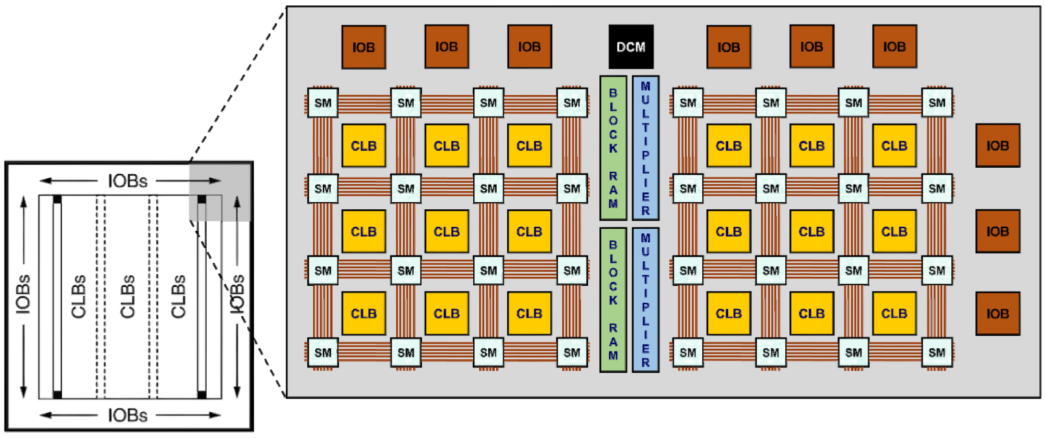


Figure 2.2: Spartan-3 Family Architecture

- Configurable Logic Blocks (CLBs) are used to implement various logical functions and also to store data.
- Input/Output Blocks (IOBs) manages the data that enters and leaves the device.
- Block RAM are 18-Kbit memories that can be used to store data.
- Multiplier blocks are highly specialized blocks whose inputs are 18-bit binary numbers.
- Digital Clock Manager (DCM) are fully digital blocks to delay, multiply, divide or apply a phase to a clock signal.

2.1.3.1
Configurable Logic Block

One of the main logic resources of an FPGA are the Configurable Logic Blocks (CLBs). The CLBs are used to implement synchronous or asynchronous combinational circuits. The internal structure of a CLB is composed of 4 slices, as shown in Fig. 2.3.

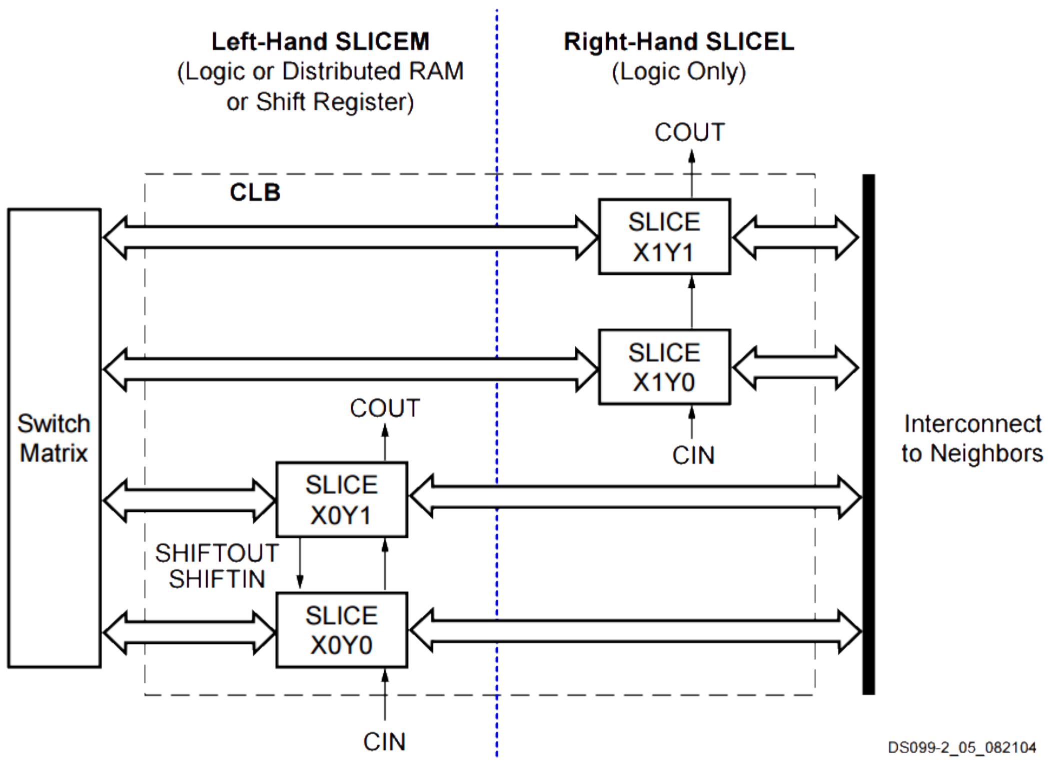


Figure 2.3: Internal arrangement of Slices within a Spartan-3 Configurable Logic Block [12].

These slices are grouped in pairs, the left pair is called SLICEM and supports both combinational logic and memory function, and the right pair is called SLICEL and supports only combinational logic. The SLICELs can have a performance advantage over the SLICEMs and also lower the cost of the device because they are simpler [12]. The table 2.1 shows the functions that each type of slice is able to provide.

	Slice type	
Supported functions	SLICEL	SLICEM
Two 4-input LUT function generators	✓	✓
Two storage elements (D flip-flop)	✓	✓
Two 2-to-1 multiplexers	✓	✓
Carry and arithmetic logic	✓	✓
Two 16x1 distributed RAM blocks	✗	✓
Two 16-bit shift registers	✗	✓

Table 2.1: Comparative table of supported functions by each type of slice.

If the Configurable Logic Blocks are the main logic resources of an FPGA, the Look-Up Tables (LUTs) are the primary features of a CLB. In the *Spartan-*

3 Generation FPGA User Guide [17], a Look-Up Table (LUT) is described by Xilinx, Inc. as follows:

> "The Look-Up Table or LUT is a RAM-based function generator and is the main resource for implementing logic functions. Furthermore, the LUTs in each SLICEM pair can be configured as Distributed RAM or a 16-bit shift register, (...). Each of the two LUTs (F and G) in a slice have four logic inputs (A1-A4) and a single output (D). Any four-variable Boolean logic operation can be implemented in one LUT. Functions with more inputs can be implemented by cascading LUTs or by using the wide function multiplexers (...). The output of the LUT can connect to the wide multiplexer logic, the carry and arithmetic logic, or directly to a CLB output or to the CLB storage element."

2.2 TDC

A Time-to-Digital Converter (TDC) can be used to measure time intervals in the range of 1 nanosecond down to the picosecond range. The very first TDCs were created as a combination of time-to-analog converters (TAC) and analog-to-digital converters (ADC). Time-to-Analog Converters (TAC) are devices capable of measuring time very accurately based on the time constant of an *a priori* known circuit. Usually, a simple, but highly precise, integrating RC circuit is fed with a constant voltage signal for the period between a *start* and *stop* input signals; after the *stop* signal, the voltage signal is withdrawn and a given voltage has been stored in the capacitor, which is proportional to its time constant $1/RC$ and the time between the *start* and *stop*. This way, one can measure the voltage across the capacitor, transform it in an Analog-to-Digital Converter (ADC) and determine the time as a binary number which goes from 0 to the maximum range of the ADC multiplied by its time resolution, hence, the name Time-to-Digital.

However, this technique is subject to many restrictions [18]. This work uses digital delay time TDCs which work without any analog components [18], ie, they use the delay of inverters gates for fine quantization of time intervals. Thus, resulting in integrated circuits which are efficient, power saving, space saving and also less expensive. Precision TDCs can achieve a time resolution down to 22 picoseconds. However, the TDC used in this work can achieve a time resolution down to 55 picoseconds, when using the "high resolution mode" which consists of internally combining two TDC channels [19,20]. Also, the threshold voltage for single ended inputs can be programmed via Digital-to-Analog Converters (DACs) between 0.8 V and 2.7 V and the measurement range in high resolution mode is $5\,\text{ns} - 3.9\,\mu\text{s}$. The maximum operating rate is approximately 20 million measurements per second.

2.3 Avalanche Photondiodes and G-APDs

Avalanche photodiodes (APDs) are devices that operate very close to the breakdown voltage of a semiconductor heterojunction in order to boost the discharge current whenever a photon is absorbed in the active region. The Linear-mode of operation consists of two elements: the APD and a transimpedance amplifier. When photons with sufficient energy excites electrons to the conductive band, a proportional current flows then the transimpedance amplifier converts the current flow delivered by the APD into a proportional voltage signal [21].

Avalanche photodiodes can also operate in the so-called Geiger-mode, where the applied voltage across the device's terminal is above the breakdown voltage. In this situation, the device becomes even more sensitive, and can respond to the arrival of a single photon (the energy necessary to generate an avalanche is equal or smaller to that of a single photon). However, great care must be taken in order to operate the APDs in the Geiger-mode, since the avalanche is self-sustained when the applied voltage is greater than the breakdown voltage and the device could never recover from a detection.

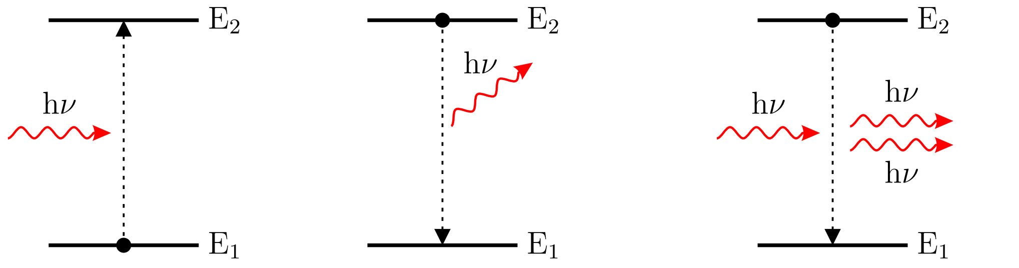
In Geiger-mode operation, the APDs are only active during a very small time window, which is usually called a "gate". The detector, therefore, checks if a photon (or more than one photon) is impinging on its active region during the gate window and then shuts off. In case of no detection, no care needs to be taken, but if the device is activated, it achieves saturation and a quenching circuit (which can be passive or active) is necessary to remove it from this state and send it to a relaxed state.

2.4 Lasers and Optical Amplifiers

Lasers are devices that emit electromagnetic radiation in the form of light with some special features: the light it emits is monochromatic, coherent and collimated. Monochromatic because it has a well-defined wavelength, coherent because all the photons emitted are in phase and collimated because the beams that propagate are electromagnetic waves almost parallels.

The acronym LASER was coined by Gordon Gould [22], an American scientist, and stands for Light Amplification by Stimulated Emission of Radiation. The theoretical foundations for the laser theory were established by Albert Einstein in his paper "On the Quantum Theory of Radiation" in 1917 [23].

In his paper, Einstein dealt with electromagnetic radiation from the mechanical-statistical point of view, and, through the re-derivation of Max Planck's law of radiation, he described the three kinds of interaction between photons and atoms under the probabilistic view: absorption, spontaneous emission and stimulated emission (Fig. 2.4).



(a) Absorption (b) Spontaneous emission (c) Stimulated emission

Figure 2.4: The three kinds of interaction between photons and atoms: (a) absorption (b) spontaneous emission (c) stimulated emission.

Optical Amplifiers are exactly as lasers, but without the optical resonator, i.e., they are merely composed of a gain medium. Different kinds of Optical Amplifiers make use of different media: the Erbium-Doped Fiber Amplifier (EDFA) is a very important low-cost fiber amplifier which makes use of energy levels created in a silica fiber doped with erbium and pumped by a lower wavelength laser; Semiconductor Optical Amplifiers (SOAs), on the other hand, make use of a semiconductor heterojunction to create the energy levels. The gain and bandwidth of different types of optical amplifiers are shown in Figure 2.5. A brief overview of EDFAs and SOAs is presented as follows.

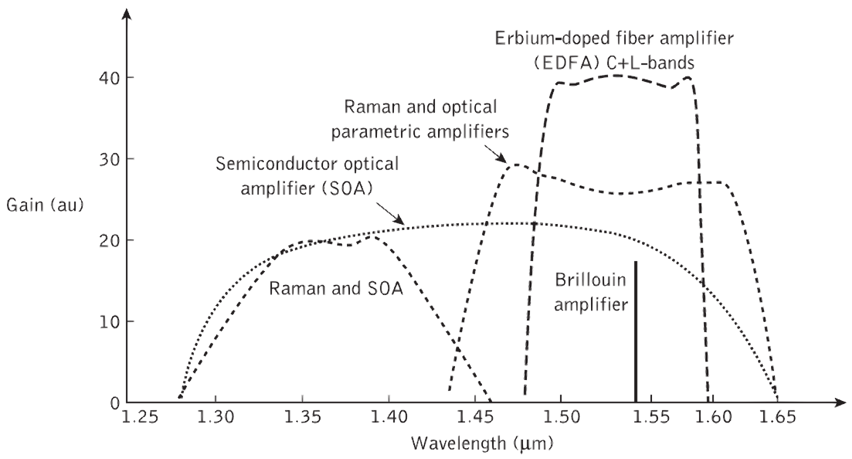


Figure 2.5: Gain-bandwidth characteristics of different optical amplifiers [10].

2.4.1 EDFA

An Erbium-Doped Fiber Amplifier is an amplifier based on a rare-earth-doped fiber which operation range is limited to the C-band (1530 to 1560 nm region, which is the window of lowest fiber loss). The length of the active medium is normally around 10 to 30 meters long and the most common host fiber materials are the standard silica, phosphate glasses and fluoride glasses [24]. In addition, EDFAs are by far the most important fiber amplifiers for long-distance telecommunication applications [10, 24] since they are phase and polarization independent and can provide high gains (around 10 and 40 dB) with a noise figure value lying between 4 and 5 dB [10, 25].

Its principle of operation is based on a pump laser emitting 1480 or 980 nm photons that are used to excite the erbium ions into metastable levels. Once the excited ions decay to the ground state via stimulated emission, the emitted photons have the same phase and direction as the signal being amplified. However, the electrons can also spontaneously decay to the ground state and when this happens, photons are emitted in all directions. The photons that fit the numerical aperture of the fiber will be guided by the fiber and will be amplified, hence the term Amplified Spontaneous Emission (ASE).

2.4.2 SOA

Semiconductor Optical Amplifiers (SOAs) are essentially like a laser diode in terms of physical structure but without the optical resonator [10]. To avoid the SOA from acting as a laser anti-reflective coatings are used at the end faces to ensure that the gain medium does not act as an optical resonator.

The physical structure of the SOA waveguide is shown in Figure 2.6. The gain wavelength can be selected by varying the composition of the active InGaAsP material [24]. An SOA is capable of amplifying signal over the whole spectral range from 1280 to 1650 nm and generating gains of up to 35 dB with low power consumption. In comparison to EDFAs, SOAs are more compact, require less operating power, have lower output power, have a typically larger noise figure ($\approx$ 8 dB [10]), and exhibit much stronger nonlinear characteristics at high input powers (such as self-phase modulation and four-wave mixing). A comparison of the performance parameters of these devices can be seen in Table 2.2.

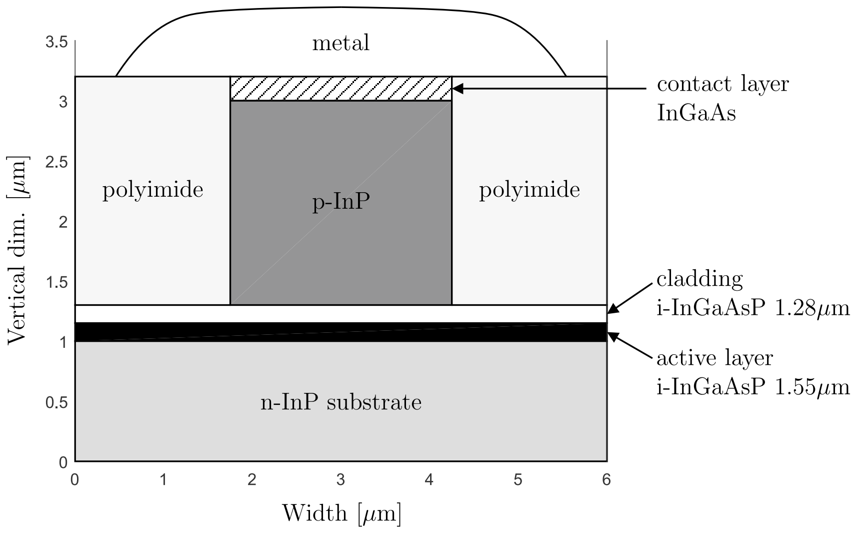


Figure 2.6: Cross section of the typical Semiconductor Optical Amplifier waveguide structure.

Parameter	EDFA	SOA	Unit
Output Power	> 15	> 13	dBm
Gain	> 24	> 25	dB
Noise Figure	< 6	< 8	dB
Polarization Mode Dispersion	< 0.3	< 0.5	dB

Table 2.2: Comparison of the main parameters of EDFAs and SOAs [10].

2.5 Optical Fibers

An optical fiber is a crystalline structure monofilament element with dielectric properties that carries electromagnetic energy in the range of optical ($\approx$ 10^{14} Hz) and infrared frequencies. Optical fibers are composed of a core made of transparent and flexible glass (silica) or plastic, into which light is transmitted, and a cladding, which confines light within the core. The light is guided inside a fiber due to the phenomenon of total internal reflection which is a consequence of a difference in refractive indexes. The difference in refractive index between the core, with refractive index $n_1 > n_2$, and the cladding ranges from 10^{-3} to 10^{-2}. By means of the Snell Law, Equation 2-1, it is possible to determine the critical angle (θ_c) that occurs when the refracted ray is parallel to the interface of the two media, i.e., $\theta_2 = \pi/2$, see Figure 2.7 (b). When the angle of incidence is greater than the critical angle there is the phenomenon of total reflection of light, see Figure 2.7 (c).

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{2-1}$$

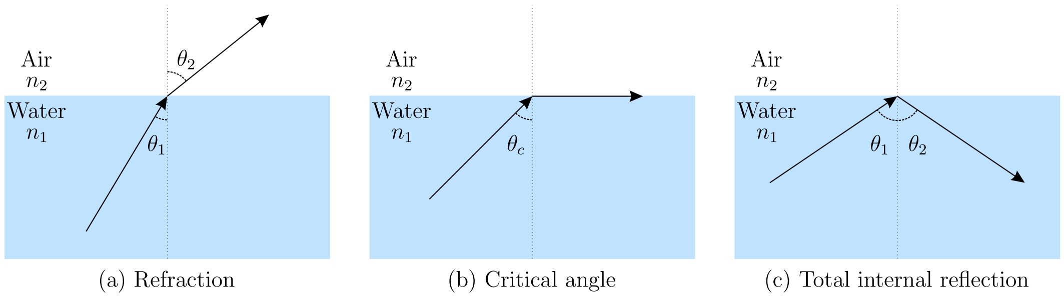


Figure 2.7: Refraction of light at the border between two media.

The acceptance angle of the fiber is a parameter that represents the largest coupling angle (θ_{max}) of a light ray, relative to the fiber axis, so that the total internal reflection occurs at the core-cladding frontier of that fiber, refer to Figure 2.8. The full acceptance angle is equal to $2\theta_{max}$, refer to Figure 2.8. The numerical aperture is the name given to the sine of the acceptance angle and is defined by Equation 2-2 where n_0, n_1 and n_2 are the refractive index of air, fiber core and fiber cladding, respectively.

$$\text{NA} = n_0 \sin\theta_{max} = \sqrt{n_1^2 - n_2^2} \tag{2-2}$$

In practice, numerical aperture knowledge is important in determining light coupling efficiency because the numerical aperture defines an accepting cone for the fiber. In addition, the optical fiber acts as a waveguide and there is a maximum amount of modes that can propagate within it. To find out how many modes can propagate within a fiber, we use Equation 2-3, where a is the radius of the fiber core and λ is the light wavelength. So, the V parameter determines the number of modes supported by a fiber [11]. A fiber will support only one propagation mode, i.e., it will be a Single-Mode Fiber, if $V < 2.405$,

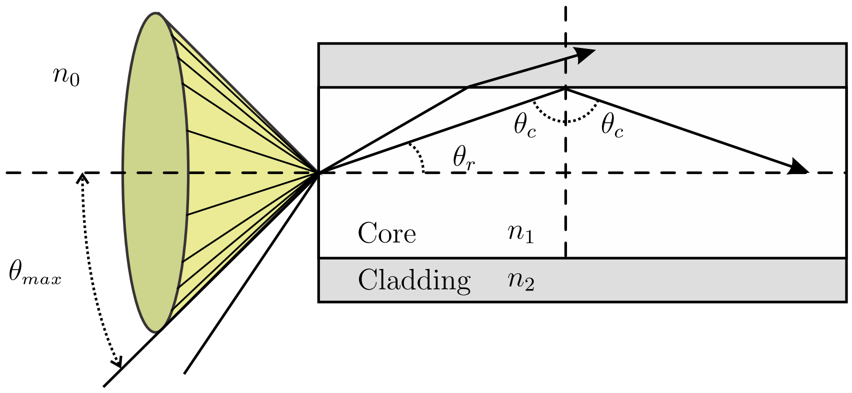


Figure 2.8: Concept of acceptance cone and numerical aperture for step-index fiber.

which is the first root of the Bessel function J_0.

$$\mathrm{V} = \frac{2\pi a}{\lambda} NA \tag{2-3}$$

Furthermore, the optic fibers have the capacity to guide more than one propagation mode, however, Single-Mode Fibers are preferable for communication since it's possible to attain higher transmission rates and greater distances when compared to Multi-Mode Fibers [11]. Among the main advantages of using fiber optics are its low attenuation (0.2 dB/km), its broad bandwidth, its reduced weight and its immunity to electromagnetic interference. With these characteristics, an optical fiber is able to transmit information with less noise and errors. However, its major disadvantages are the high cost of implementation and maintenance, but over time and with widespread use have made the optical fibers a less expensive alternative than the transmission in copper wires [26].

2.5.1 Rayleigh Scattering

Rayleigh scattering occurs due to particles in the fiber optic, such as atoms, molecules or fine dust, that are much smaller than the wavelength of the laser [25], typically less than $\frac{\lambda}{15}$ [27]. Rayleigh scattering can be perceived when light travels through gases or transparent solids and liquids. When a pulse is launched into an optical fiber, less than 0.0001% of the light is spread in the opposite direction of the pulse [28, 29]. Light scattered in the opposite direction of the light pulse is called backscattered light. Rayleigh scattering is the main loss factor in an optical fiber. It causes the incident light on the optical fiber to be scattered in all directions, as shown in Figure 2.9. The

Rayleigh coefficient in optical fibers can be calculated from Equation 2-4 [30],

$$\alpha_s = \frac{8\pi^3}{3\lambda^4} n^8 p^2 k T_f \beta \tag{2-4}$$

where: α_{scat} is the Rayleigh backscatter coefficient (unit-less), λ is the wavelength of light (m), n is the refraction index in the fiber core, p is the photo-elastic coefficient of the glass, k is the Boltzmann constant ($1.3806 m^2 kg s^{-2} K^{-1}$), β is the isothermal compressibility (fractional change in the volume of the fiber as pressure changes at a constant temperature), T_f is a fictive temperature, representing the temperature at which the density fluctuations are "frozen" in the material (K).

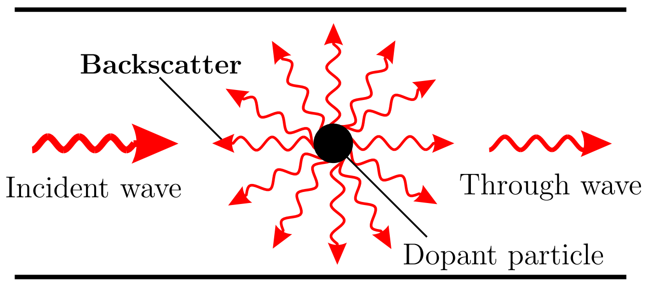


Figure 2.9: Rayleigh Scattering

2.5.2 Fresnel Reflection

When the light traveling on the optical fiber encounters a change in the density of the material, a part of the light is reflected back to the source. These changes in material density generally occur at the beginning or end of the fiber, at connectors, at breaks or at fiber junctions.

The Fresnel reflection at the beginning of the fiber can be calculated by the equation 2-5. Knowing the resulting power from the Fresnel reflection (P_{ref}) at the beginning of the fiber it is possible to discover the power of the incident light (P_0). If the interface on the connector is perfectly polished and cleaved perpendicular to the core axis, then about 4% of the incident power will be reflected [24].

$$P_{ref} = P_0 \left(\frac{n_{fiber} - n_{air}}{n_{fiber} + n_{air}} \right)^2 \tag{2-5}$$

2.6

Fiber Monitoring

Fiber monitoring is essential to enable long-distance optical telecommunication links since the high data rates can be jeopardized due to the mechanical fragility of optical fibers. In addition, the supervision of the physical layer of the network is fundamental because a break can cause the suspension of essential services, such as optical links from banks, telephone or internet services, etc.

One of the most used techniques for the characterization of optical links is optical reflectometry. There are two main types of optical reflectometry, namely: Optical Time Domain Reflectometry (OTDR) [31] and Optical Frequency Domain Reflectometry (OFDR) [32]. Reflectometry techniques provide link characterization without the need of end-to-end measurements relying on the phenomena of Rayleigh backscattering.

In OTDR applications, spatial resolution and dynamic range come as a trade-off since strengthening the pulse for enhanced reach affects the pulse width and diminishes 2-point resolution [11].

2.6.1 OTDR

The most widely used optical reflectometry technique is Optical Time Domain Reflectometry (OTDR). An OTDR is an optoelectronic instrument used to characterize an optical fiber. The characteristics that can be extracted from an OTDR trace are the fiber length estimation, fiber attenuation, fault location and/or loss due to connectors or splices. It is also possible to detect a gradual or sudden degradation of the fiber as a function of time through comparisons with previous fiber tests.

The basic principle of an OTDR is the detection of reflected and/or backscattered light, due to the small imperfections and impurities in an optical fiber. To perform a measurement, only access to one end of the fiber is required. An OTDR should have a compromise between good spatial resolution and dynamic range. It is important to note that when we want a good spatial resolution, the dynamic range will be compromised, and vice versa.

2.6.1.1 How does an OTDR work?

The OTDR operates as follows, pulses of light, which are emitted periodically, are coupled to the test fiber through a circulator, see Figure 2.10. The light pulses enter through the port 1 of the circulator and exit through port 2 which is connected to the optical fiber being tested. As these pulses propagate within the fiber a part of the light is reflected through the Rayleigh backscatter and Fresnel reflection phenomena. The backscattered light enters through port 2 of the circulator and is directed to port 3 which is connected to a detector. The electric signal generated by the detector is sent to a microprocessor which calculates the round trip time and the power of its reflections of points within the fiber and produces a OTDR trace, as shown in Figure 2.11.

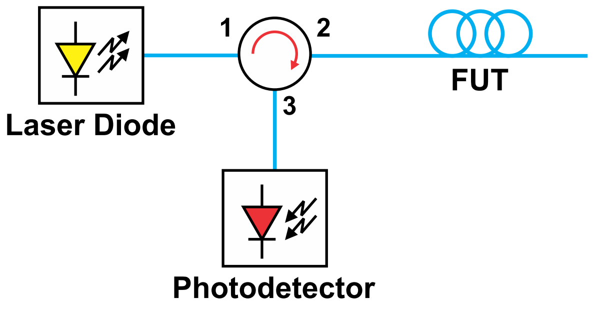


Figure 2.10: Block Diagram of a OTDR.

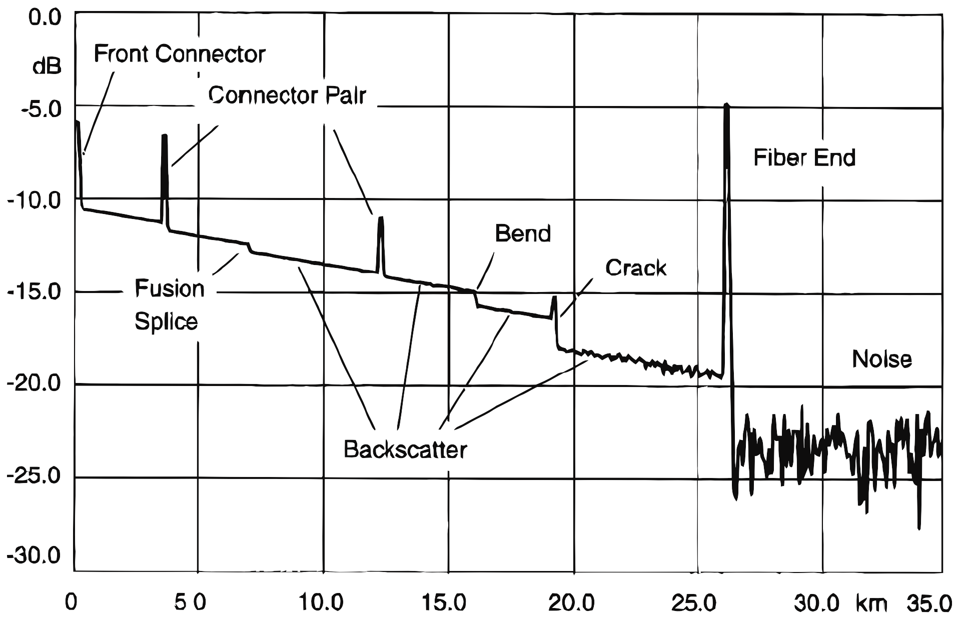


Figure 2.11: A typical OTDR trace [33].

From the time parameters in which the pulse enters the fiber, refractive index of the optical fiber and the time in which there is a detection, it is possible to calculate the distance at which this event occurred through the equation 2-6.

$$d = \frac{c}{2n} t \tag{2-6}$$

where c is the speed of light, n is the refractive index of the fiber and t is the round-trip travel time.

2.6.2 Characteristics of an OTDR

2.6.2.1 Backscatter Level

According to Anritsu's white-paper *Understanding OTDRs* it is possible to draw a relation between the backscattered power level and the incident power in the fiber, although the OTDR measures only the backscattered power level and not the power level of the transmitted light. The "backscatter coefficient" is known to be the ratio of backscattered light power to transmitted light [28]. The "Backscatter Coefficient" is given by equation 2-7.

$$k = S \cdot \alpha_s \tag{2-7}$$

where α_s is the scattering coefficient (Eq. 2-4) and S is backscattering capture coefficient which is the fraction of the light guided back to the OTDR. The backscattering capture coefficient is given by [33]:

$$S = \left(\frac{\mathrm{NA}}{n_0}\right)^2 \cdot \frac{1}{m} \tag{2-8}$$

where NA is the fiber's numerical aperture, n_0 is the fiber core's refractive index and m is 4.55 for single-mode fibers [34].

2.6.2.2 Dead zone

Dead zones occurrs after strong reflections, i.e., the receiver becomes blinded during a certain time interval due to a temporary saturation undergone by the photodetector [10, 28] and can are defined as the distance in which the Fresnel reflection is $\pm$ 0.5 dB from the backscattered power [25, 28]. Decreasing the pulse width, one can reduce the dead zone but it will also lower the dynamic range since the dynamic range is related to the pulse width. The width of the dead zone will be at least the width of the pulse plus the recovery time of the photodetector. The recovery time is the time the photodetector takes to readjust to its maximum sensitivity [28].

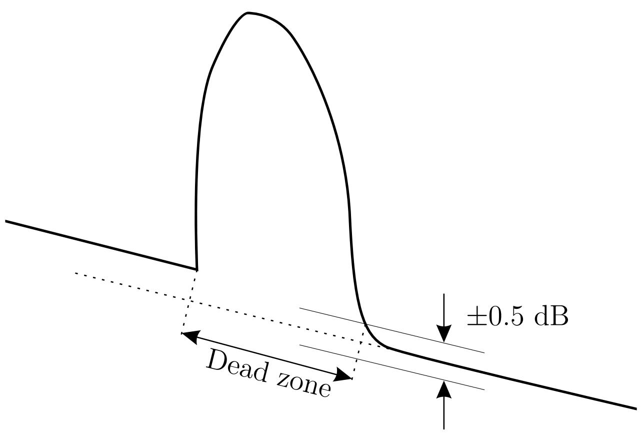


Figure 2.12: The representation of a dead zone in an OTDR profile.

2.6.2.3 Dynamic Range

One of the most important parameters of an OTDR is the dynamic range, which is expressed in dB, and refers to the maximum length of an optical link that can be measured or can also be understood as the maximum attenuation that can be measured in a fiber. The dynamic range is defined by the combination of detector sensitivity, the pulse width and the peak power. The dynamic range is determined by the difference between the initial backscatter level and the noise level for a signal-to-noise ratio equal to 1 [25,28,33].

2.6.2.4 Spatial Resolution

The spatial resolution of the OTDR indicates the ability to solve two adjacent events, i.e., it is the distance in which two reflections can be resolved. The spatial resolution depends mainly on the light pulse width entering the fiber, however, the pulse will widen as it travels within the fiber due to the chromatic dispersion. There is also a compromise relationship between dynamic range and spatial resolution, i.e., for high resolution narrow pulses are required, however, the narrower the pulse, the less energy it will have and therefore will result in a low dynamic range.

2.6.2.5 Distance Accuracy

The accuracy of the distance is another important factor of the OTDR and depends on the uncertainty of the refractive index, spacing between the points, jitter on the detector, the light pulse width and the dispersion.

2.6.3 Other types of reflectometers

Although OTDR has been widely adopted in the field for fiber monitoring there are various others techniques that have advantages and drawbacks over the plain OTDR. Assuming that an OTDR is a linear and time-invariant system, it is possible to obtain the output of the system from the convolution of the transfer function with any given input. If the input is composed of a number of identical pulses, the output of the OTDR to each pulse will be the same with different delays and the total output will be their superposition. In the early stage the coded OTDR employed pseudo-random bit sequences to scan the fiber and correlation techniques to recover the fiber trace [35,36]. One of the advantages of coded OTDR is that it can improve signal-to-noise (SNR) using a pulse sequence while the spatial resolution remains the same as that of the single-pulse response.

The Chaotic OTDR makes use of a continuous-wave chaotic laser light which generates a real random signal with bandwidth of several tens of GHz which, in turn, leads to higher spatial resolution. Though spatial resolutions much lower than 0.6m can be achieved [37], this technique cannot be used to measure the fiber attenuation. The Chaotic OTDR can only be used to detect reflection events by cross correlation method [38].

There are two types of OFDR: coherent-OFDR (C-OFDR) and incoherent-OFDR (I-OFDR). The OFDR technique uses continuous wave light source which is swept continuously in frequency. The sweep can be done by varying the optical carrier, coherent-OFDR, or its modulation envelope, incoherent-OFDR [32,39,40]. The coherent measurement consists of analyzing the heterodyne beat between a fixed reference reflection and the backscattered signal.

This technique is suited for the fiber characterization since it offers a superior dynamic range (70 dB [39]) and spatial resolution than tradicional OTDRs.

2.6.4 Photon Counting OTDR

With the advent of the Geiger-mode Single Photon Detector in the telecommunication wavelength based on InGaAs/InP Avalanche Photodiodes, the Photon-Counting OTDR (ν-OTDR) was proposed. Such devices offer higher sensitivity due to extremely low power detection in the single-photon regime [21].

The G-APD can be used in gated mode or in free running mode. The free running mode is usually used when the photon arrival time is unknown, the gated mode, on the other hand, is used when the photon arrival time is known or when a synchronized signal is being detected. Also, the gated mode can attain high photon detection efficiencies and extremely low dark count rates [21, 41]. The gated operation requires, however, an intelligent management system to reduce monitoring periods and enhance the acquisition of statistically relevant data.

Recently, a tunable ν-OTDR with 5 meters spatial resolution and 32 dB dynamic range has been proposed in a WDM-PON context [42], the Tunable Photon Counting OTDR. In [43], the ν-OTDR is employed in a setup for ultra-high resolution measurements up to 2 centimeters with a 10 dB dynamic range, the Ultra-High Resolution Photon Counting OTDR.

The ν-OTDR can offer some advantages when compared to a conventional state-of-the-art OTDR, some of which include: higher spatial resolution, better dynamic range, better 2-point resolution, lower timing jitter and superior behavior concerning afterpulsing [21,41,44]. Of course, there are disadvantages too, some of which are: dead zones after large loss events (charge persistence effect) [41] and trace speed, depending on the application [45].

2.6.5 High Dynamic OTDR (FPGA)

In [44] is presented a methodology that aims to ally the high sensitivity of a ν-OTDR with the parallel processing capacity of an FPGA. Through the use of an FPGA, it is possible to manage several devices while collecting data from the Single Photon Avalanche Detector (SPAD).

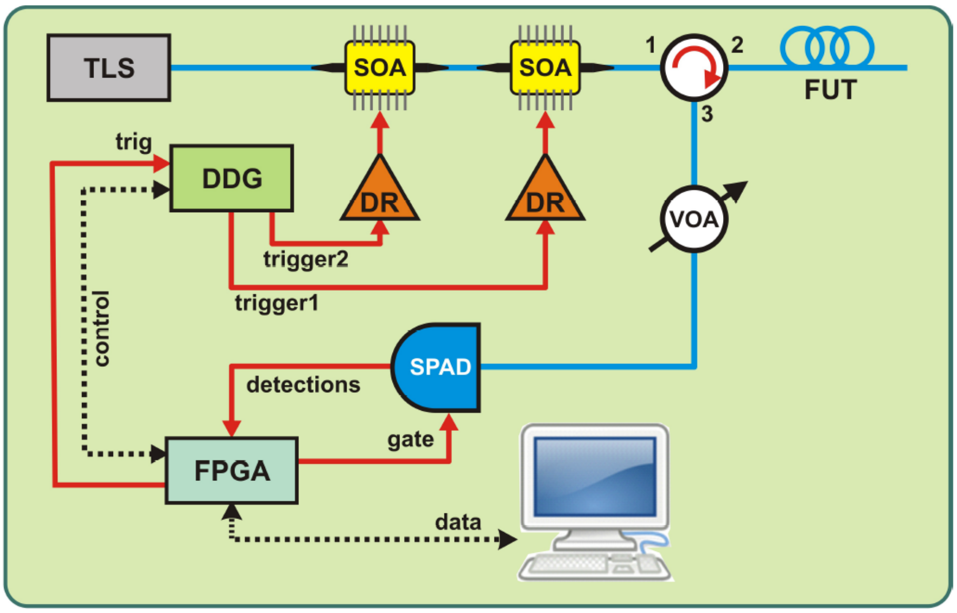
Among the features of Tunable ν-OTDR is its high dynamic range, its high-speed traceability, its ability to zoom in on a designated portion of the fiber, among others.

Furthermore, a Variable Optical Attenuator (VOA) is coupled to the SPAD's input. The VOA is used to reduce the effects of a temporary saturation undergone by the SPAD due to a strong reflection[2]. It should be borne in mind that the longer the optical link the smaller the power returned to the detector. In this sense, "the possibility of selecting the portion of the fiber to be monitored allied with an attenuation control capable of keeping a high detection rate is a relevant tool that can be applied in many different areas" [44].

[2]Refer to Section 2.6.2.2 for additional explanation about dead zone.

2.6.5.1 Setup

In Fig. 2.13, the setup of the High Dynamic ν-OTDR is presented. The first SOA, right after the Tunable Laser Source (TLS), boosts the optical signal inside a narrow window of 60 ns with a 2 A peak current driver creating high power narrow light pulses. The second SOA is driven by a 600 mA current pulse with 60 ns width. Even though the driving pulse is not enough to reach full power operation of the second SOA alone, the first boost guarantees optimal power usage while the second acted as a gainless optical switch, shaping the width of the optical pulse down to 60 ns. The tandem configuration of the SOAs guarantees a high extinction rate, contributing to the overall dynamic range of the detections.



TLS: Tunable Laser Source, **SOA**: Semiconductor Optical Amplifier, **D**: Driver, **DDG**: Digital Delay Generator, **SPAD**: Single Photon Avalanche Detector.

Figure 2.13: High Dynamic OTDR experimental setup [43].

The FPGA send trigger pulses to the Digital Delay Generator (DDG) which then drives both SOAs. The DDG drives each SOA respecting the optical delay between both to ensure perfect synchronization.

The gate pulse creates a 10 ns detection window. Therefore, an FPGA running at 100 MHz is employed as a data acquisition unit. The saved data is transmitted through an USB cable to a computer interface. The attenuation imposed by the VOA is set based on the results of Digital Signal Processing from a ℓ_1 Trend Filter [46]. This system could run in a loop and after each batch of data processed by the ℓ_1 Trend Filter, the FPGA could perform a zoom over the area near each break point. Tunability is guaranteed by the TLS.

An example of measurement using this Tunable ν-OTDR is presented in Figure 2.14 [47]. A one-hour measurement was performed on a test fiber to measure the dynamic range of the Tunable ν-OTDR. A test fiber simulates a PON network with a 3.5 km feeder fiber and a 4 km fiber, approximately, as the channel fiber. At the end of the feeder fiber, a 16 dB bend loss was induced to simulate a 1x32 power splitter with an insertion loss of 1 dB. The obtained results were a 6 m, approximately, spatial resolution and a 32 dB of dynamic range. The spatial resolution is defined as the full-width at half-maximum (FWHM) of the reflection peak above the Rayleigh backscattering level [48].

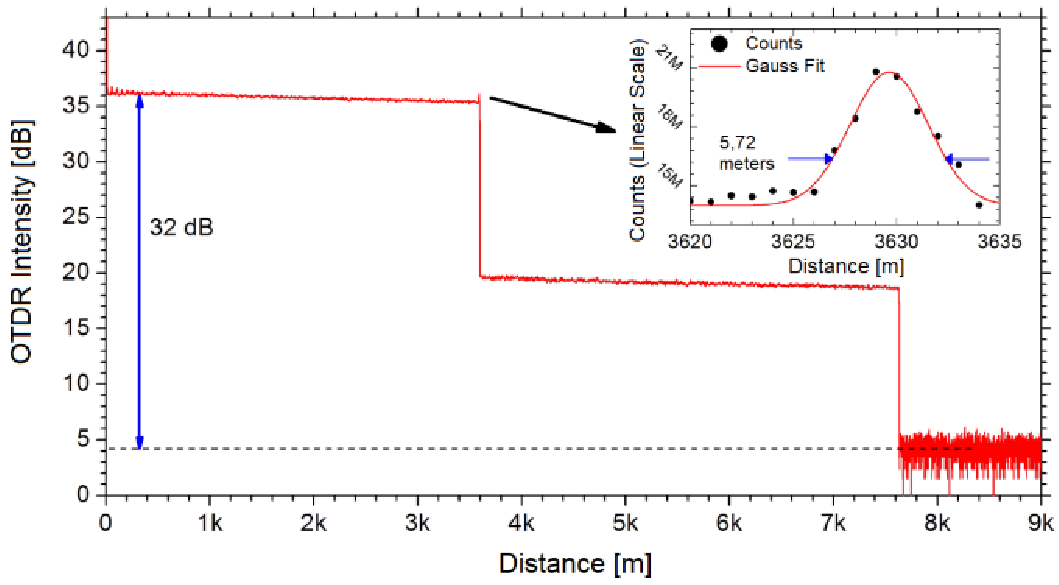


Figure 2.14: Achievable dynamic range of 32 dB with the proposed ν-OTDR setup. The inset shows the gaussian fit of the reflection peak in linear scale [43].

2.6.5.2 FPGA Logic

For a classic OTDR to work it is necessary that there is only one pulse of light inside the fiber, however, this fact substantially increases the monitoring period. An approach that seeks to maximize the number of light pulse detections is presented in [44, 49]. The main idea of this approach is to create a train of gates spaced apart by the dead time of the SPAD, and, with every light pulse that is sent into the optical fiber, a delay is added to the train of gates to ensure that the entire fiber is analyzed. The FPGA board is responsible for managing the train of gates and the delays between pulses as well as enabling the light pulses through the (SOA) Driver which works as a light switch [44].

This process is graphically shown in Figure 2.15 [47]. Each detection is associated with a 16 bit word composed by r and s, the Gate and Pulse number respectively. The parameters a, b and c represent the gate window, the dead time and the maximum delay between pulses.

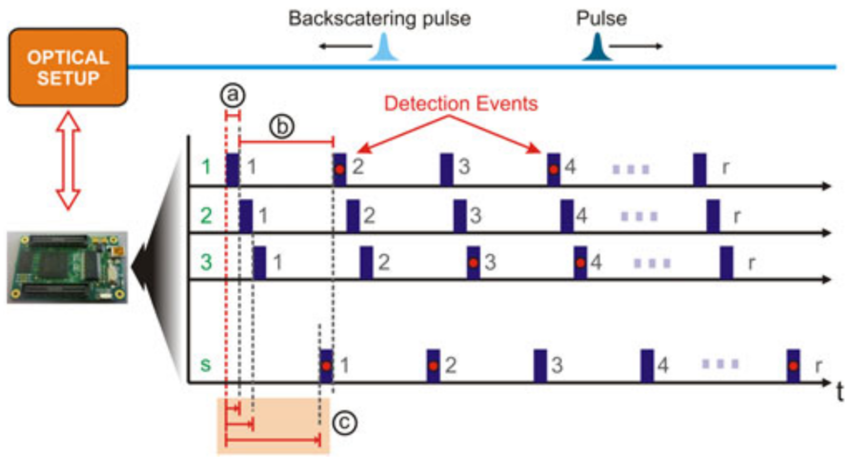


Figure 2.15: Method for detection indexation employed by the FPGA board [47].

2.6.5.3 User interface

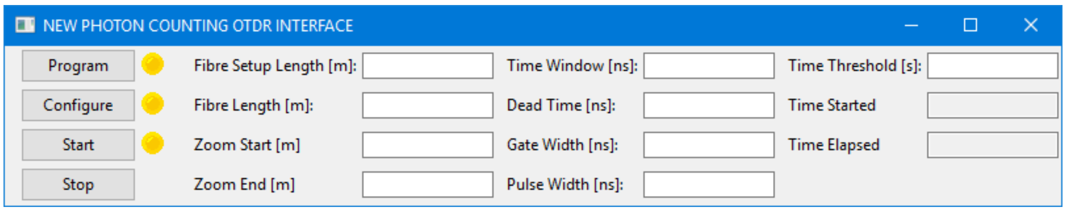
The Graphical User Interface (GUI) of the High Dynamic ν-OTDR is based in Python^{TM}. In this interface, the acquisition parameters are entered and then sent to the FPGA via a USB interface. The FPGA is responsible for generating the train of gates that are separated by the dead time of the SPAD and controlling the SOAs so that only one light pulse is inside the optical fiber, as explained earlier. The GUI is presented in Figure 2.16 and the description of the parameters is shown below:

- *Fiber Setup Length*: determines the start of the optical fiber, takes into account the length of the launch fiber.
- *Fiber Length*: determines the maximum length of optical fiber to be measured.
- *Pulse Width*: determines the light pulse width.
- *Time Window*: determines the acquisition granularity, must be greater than 10 ns.
- *Dead Time*: determines the dead time of the SPAD.
- *Gate Width*: determines the acquisition window of the SPAD.

– *Zoom Start*: if set, determines the initial position of the fiber that will be characterized.

– *Zoom End*: if set, determines the final position of the fiber that will be characterized.

– *Time Threshold*: determines the acquisition time.



NEW PHOTON COUNTING OTDR INTERFACE

Program	●	Fibre Setup Length [m]:		Time Window [ns]:		Time Threshold [s]:
Configure	●	Fibre Length [m]:		Dead Time [ns]:		Time Started
Start	●	Zoom Start [m]		Gate Width [ns]:		Time Elapsed
Stop		Zoom End [m]		Pulse Width [ns]:		

Figure 2.16: GUI of the High Dynamic ν-OTDR.

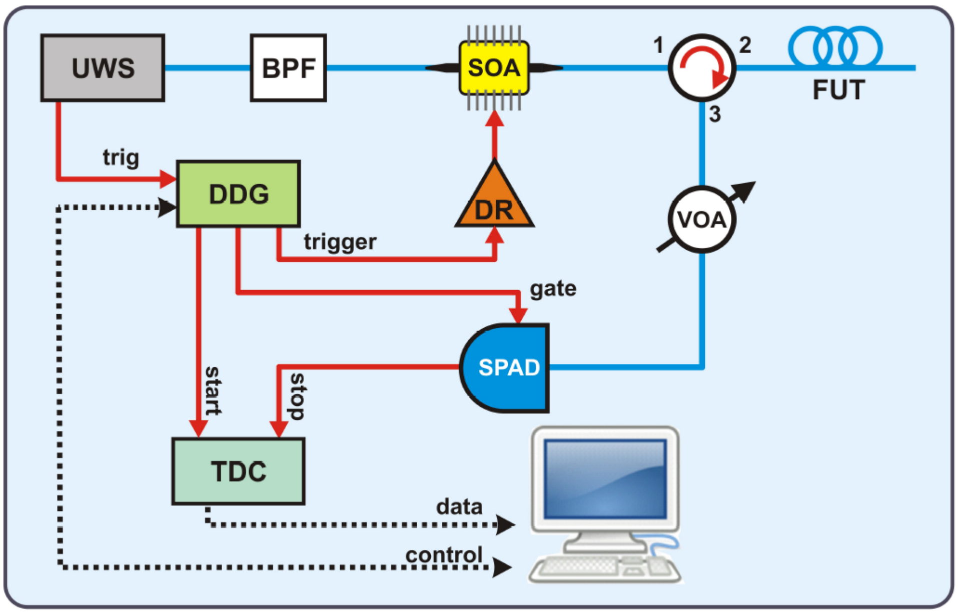
2.6.6 High Resolution OTDR

In [43] is presented a methodology that aims to ally the high sensitivity of a ν-OTDR with 115 femtoseconds wide pulses from an Ultra-Wideband Source (UWS) and the time resolution of a Time-to-Digital Converter (TDC) to create the Ultra-High-Resolution ν-OTDR. This setup presents an up to 2 centimeters spatial resolution with a 14 dB dynamic range.

2.6.6.1 Setup

In Fig. 2.17, the experimental setup of the High Resolution ν-OTDR is presented. The 115 fs wide pulses from an Ultra Wideband Laser Source (UWS) are modulated by a Semiconductor Optical Amplifier (SOA) in order to reduce the pulse repetition rate and satisfy the condition of one light pulse traversing the fiber at a time.

The enabling pulses of the SOA are generated by a Digital Delay Generator (DDG), which is triggered by the UWS. These enabling pulses are reshaped by the SOA driver which is capable of driving the SOA with 4 ns pulses. Since the light pulse passing through the SOA is 115 fs wide which is much narrower than 4 ns, the transmitted light pulse degrades due to the presence of amplified spontaneous emission (ASE) generated by the SOA. As stated in [50], "the ASE floor is, however, 10 dB below the pulse peak and has little effect on the achievable spatial resolution of the technique". The same DDG also triggers a Single Photon Avalanche Detector (SPAD) with varying delays such as to cover the whole fiber length. Detections are



UWS: Ultra Wideband Laser Source, **BPF**: Band Pass Filter, **D**: Driver, **SOA**: Semiconductor Optical Amplifier, **TDC**: Time to Digital Converter, **SPAD**: Single Photon Avalanche Detector, **DDG**: Digital Delay Generator.

Figure 2.17: High Resolution OTDR experimental setup.

processed by a Time-to-Digital Converter and sent to a computer. A Variable Optical Attenuator (VOA) at the SPAD's input guarantees that the power does not exceed the saturation limit so the detector is kept at linear regime [21]. Tunability is achieved by the use of a narrow bandpass filter at the source's output (BPF) [51].

Since the UWS generates ultranarrow pulses at 5.8 MHz, the DDG must trigger the SOA driver every x pulses from the UWS in order to satisfy the condition of one pulse inside a fiber at a time. This can be done by configuring the DDG Prescaler. To determine x, the Eq. 2-9 must be used.

$$x = \frac{1}{T_s} \cdot \frac{2 f_L \, n}{c} \tag{2-9}$$

Where: c is the speed of light, n is the refractive index of the fiber, f_L is the length of the fiber and T_S is the UWS's period.

In addition, the acquisition is carried out in windows that correspond to the width of the gate pulse sent to the SPAD [45]. The DDG sends a start pulse to the TDC, and, at a predetermined time, sends a gate pulse to the SPAD. If there is any detection, a stop pulse is sent to the TDC. Knowing the time when the light pulse enters the fiber, and the start and stop times of the TDC it is possible to compose the OTDR trace. Each window is acquired over a period of time, then a new window is acquired until it covers the entire length of the fiber. Since the SPAD is highly dependent on the above-threshold voltage applied, the transient region of the gate pulse should be discarded during the acquisition process. The gate pulse response is presented in Fig. 2.18.

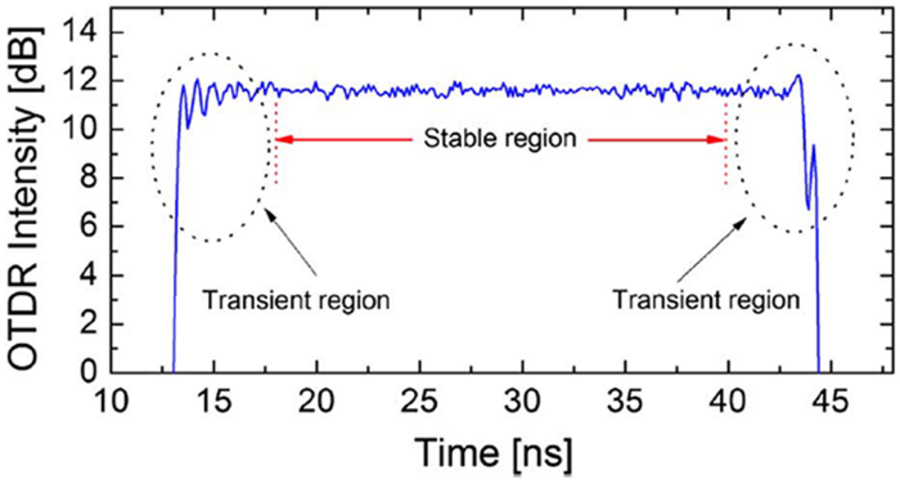


Figure 2.18: Response of the SPAD to a enable-pulse [50].

2.6.6.2 User interface

The Graphical User Interface (GUI) of the High Resolution ν-OTDR is based in LabVIEW^{TM}. The configuration of the DDG is done through General Purpose Interface Bus (GPIB), which is an 8-bit parallel bus standardized under IEEE-488, and the configuration of the TDC is done via an acquisition board connected to a Peripheral Component Interconnect (PCI) bus. The GUI is presented in Figure 2.19 and the description of the parameters is shown below [45]:

– *DDG Prescaler*: determines the SOA trigger rate in order to satisfy the condition of one pulse inside a fiber at a time, see Eq. 2-9.

– *Fiber Length Setup*: determines the start of the optical fiber, takes into account the length of the launch fiber.

– *Acquisition Time*: determines the acquisition time.

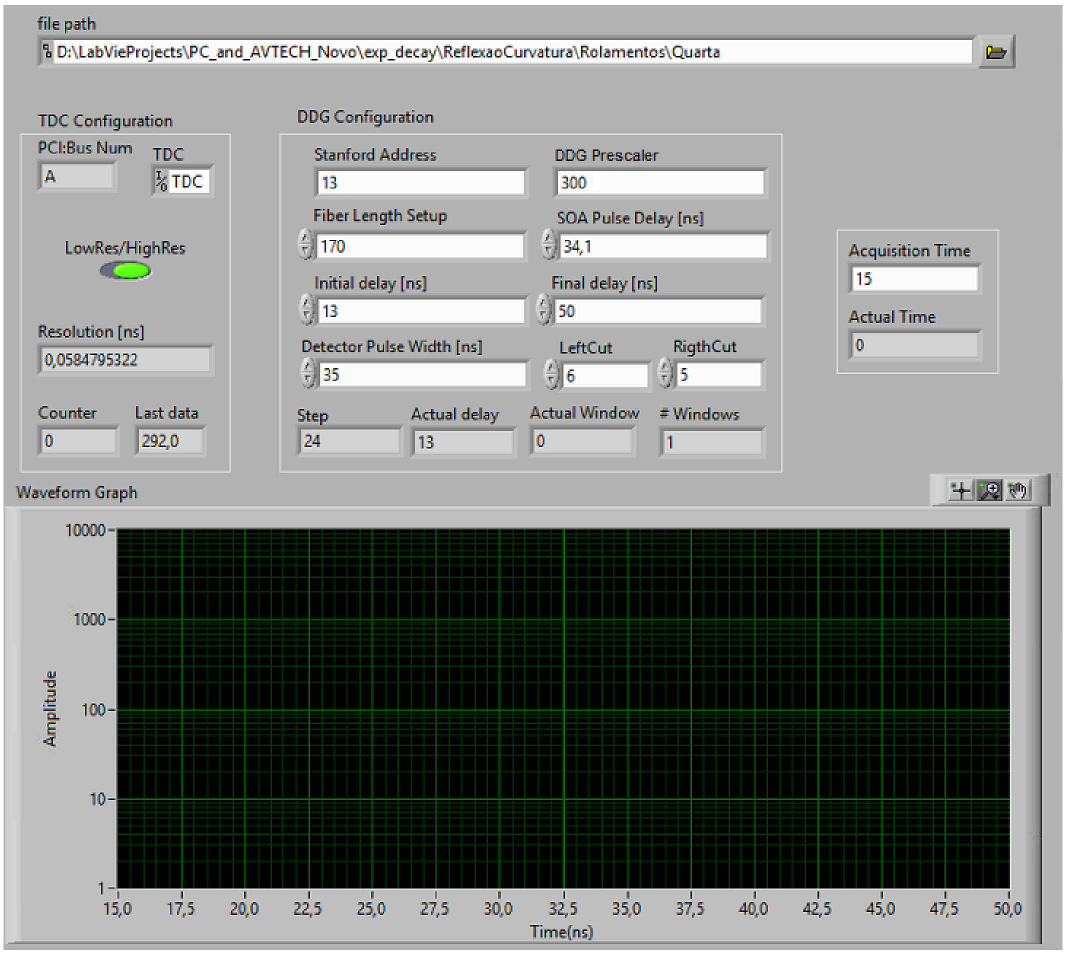
– *SOA Pulse Delay*: synchronizes the arrival time of the light pulse (UWS) with the activation of the SOA.

– *Initial delay*: determines the initial position of the fiber that will be characterized.

– *Final delay*: determines the final position of the fiber that will be characterized.

– *Detector Pulse Width*: is the electrical pulse width send to the gate of the SPAD.

– *LeftCut* and *RightCut*: determines the transient region inside a detection window of the SPAD, every detection inside this transient region will be discarded, see Fig. 2.18.



file path: D:\LabVieProjects\PC_and_AVTECH_Novo\exp_decay\ReflexaoCurvatura\Rolamentos\Quarta

TDC Configuration
PCI:Bus Num: A
TDC: TDC
LowRes/HighRes
Resolution [ns]: 0,0584795322
Counter: 0
Last data: 292,0

DDG Configuration
Stanford Address: 13
DDG Prescaler: 300
Fiber Length Setup: 170
SOA Pulse Delay [ns]: 34,1
Initial delay [ns]: 13
Final delay [ns]: 50
Detector Pulse Width [ns]: 35
LeftCut: 6
RigthCut: 5
Step: 24
Actual delay: 13
Actual Window: 0
Windows: 1

Acquisition Time: 15
Actual Time: 0

Figure 2.19: GUI of the High Resolution ν-OTDR.

2.6.7 Results from previous assemblies

The HR-ν-OTDR depicted in Fig. 2.17 was first presented in [43, 45] where Herrera *et al.* cited some applications like the use of this experimental setup for WDM-PON monitoring. The 2.8 cm spatial resolution can be maintained for distances up to 500 m, and 11.4 cm resolution at distances above 20 km as shown in Fig. 2.20. The pulse broadening is due to chromatic dispersion but by using a Bragg filter before the circulator it's possible to maintain the spatial resolution shown in Fig. 2.20 [43].

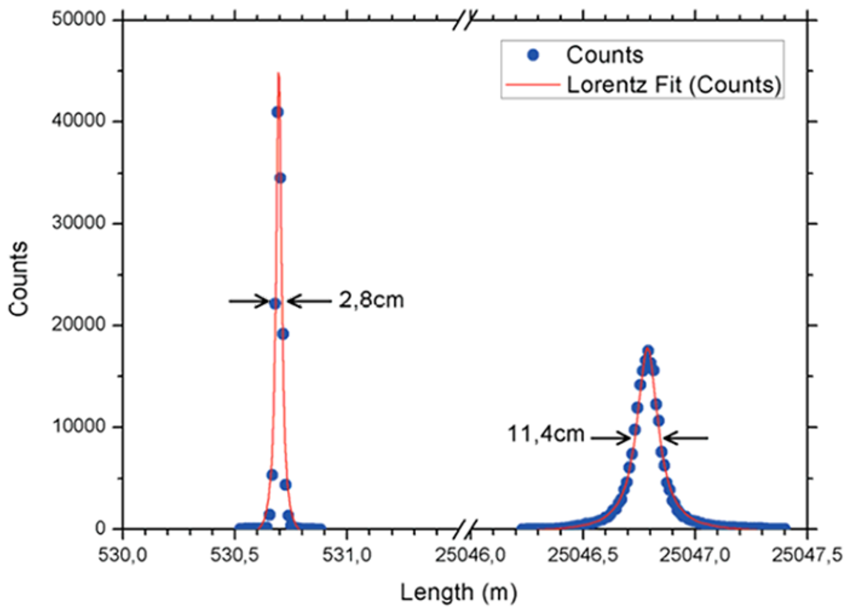


Figure 2.20: Spatial resolution of the HR-ν-OTDR for different distances [43].

Its use also would diminish the maintenance costs due to its high resolution feature capable of offering precise localization of a fault in a fiber. However, the 10 dB dynamic range imposed due to the low extinction rate provided by the SOA is still a great concern. A later study about the loss due to fiber bending was also conducted and the results were presented in [50].

Since the emergence of optical communications systems, passive optical networks (PON) have been seen as a solution for point-to-multipoint optical networks for simplicity, flexibility, scalability and lower power consumption. Telecommunications companies are increasingly looking for alternatives to lower operating costs, including in-service passive optical network (PON) monitoring costs. Possible monitoring solutions include the use of optical time domain reflectometry (OTDR) techniques. Although the monitoring of a TDM-PON [52] or TWDM-PON represents a very challenging problem, the monitoring of a WDM-PON can, in theory, be performed by an OTDR whose wavelength is tunable [53]. In [21, 54] "high dynamic range ν-OTDR" were used for PON monitoring, although tunability was not considered.

Amaral *et al.* used the HD-ν-OTDR depicted in Figure 2.13 in [42] to evaluate the feasibility of using an OTDR For WDM-PON monitoring. It is worth remembering that the tunability of HD-ν-OTDR is guaranteed by the TLS. A 32-channel cyclic AWG with 0.8 nm width covering the C, S and L bands was used. The bands C, S and L are used for downstream, upstream and monitoring, respectively. Two adjacent channels of the C band were used to simulate two subscribers, $\lambda_1 = 1434.4$nm e $\lambda_2 = 1435.2$nm. In Fig. 2.21 the

OTDR trace for the two channels is shown, which shows the existence of a 3.3 km feeder fiber connected to the AWG which induces a 5 dB attenuation for channel 1 and 5.5 dB for channel 2. Two fibers with a length of 12.32 km and 4.29 km were connected to channel 1 and a fiber of 6.25 km to channel 2. It is interesting to note that the spatial resolution reached was 5 m with 17 dB of dynamic range, considering a 15 dB attenuation imposed by the VOA which guarantees to the setup a dynamic range reachable of 32 dB.

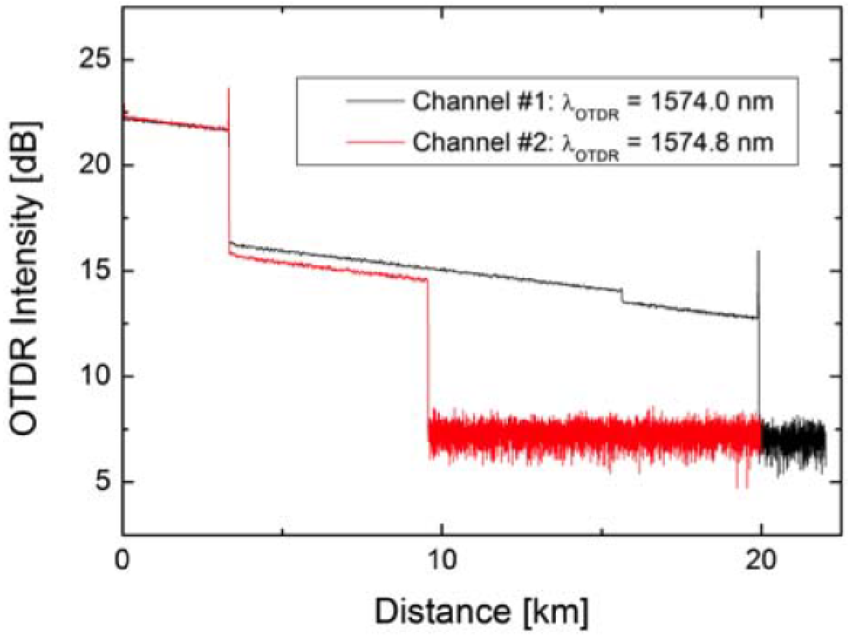


Figure 2.21: OTDR traces for two AWG channels λ_1 = 1434.4nm e $\lambda_2 = 1435.2$nm [42].

In [47] it is proposed to use a trend filter ℓ_1 for the automatic identification of faults in a fiber. The methodology used consisted of using HD-ν-OTDR to acquire the fiber profile that would then be processed by a modified ℓ_1 trend filter that looks for level changes that could indicate fiber faults. Subsequently, with the filter data new measurements could be made only in the region around the points of interest.

3 Experiments

3.1 High Resolution Automatic Fault Detection in a Fiber Optic Link via Photon Counting OTDR

Initially, tests were performed individually using the two OTDRs shown above, the High Dynamic ν-OTDR (see Fig. 2.13) and the High Resolution ν-OTDR (see Fig. 2.17), using a modified trend filter, the "ada ℓ_1". The results show a 2.25 m spatial resolution and a 29 dB dynamic range for High Dynamic ν-OTDR, and a 3.5 cm spatial resolution and a 12 dB dynamic range for High-Resolution ν-OTDR.

In Fig. 3.1, the monitoring result of a 1x32 PON architecture using the High Dynamic ν-OTDR is presented. The 3.5 km feeder fiber is connected to a 4.5 km user line by a 1x32 splitter, which induces a 15 dB loss. Both reflective and non-reflective events are present in the user line as an effect of defective connectors and induced bend losses, respectively. In red, is presented a detailed detection of the ℓ_1 Trend Filter.

In Fig. 3.2, we present the analysis of a reflective peak at the 1x32 splitter input. The data in linear scale is fitted by a Gaussian curve and the Full Width at Half Maximum criterion is used to determine the achievable spatial resolution of 2.25 meters.

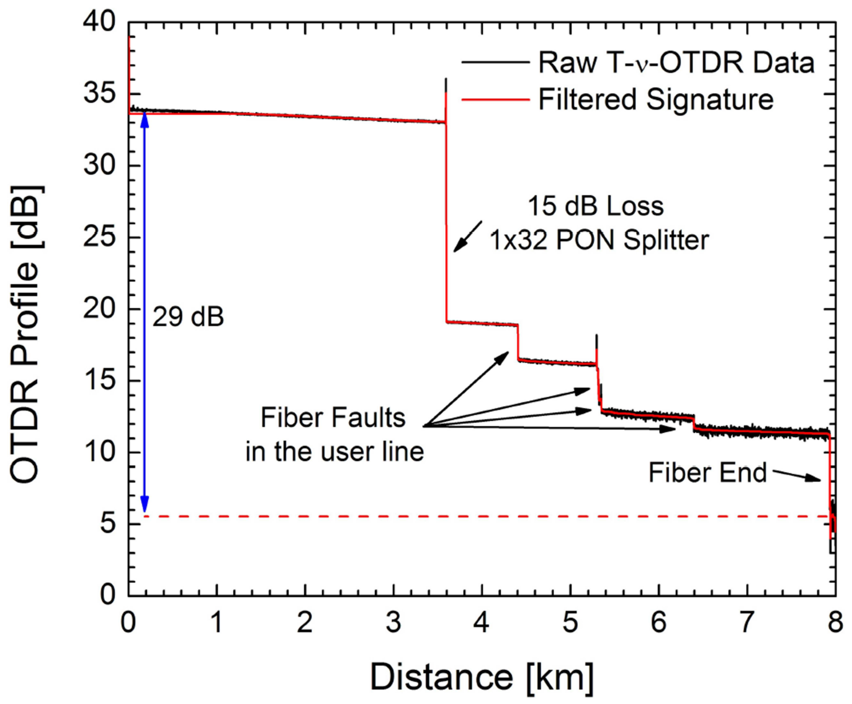


Figure 3.1: 1x32 PON architecture. 3.5 km feeder fiber + 4.5 km user line. Losses are inducted by bending (non-reflective) and by adding defective connectors (reflective).

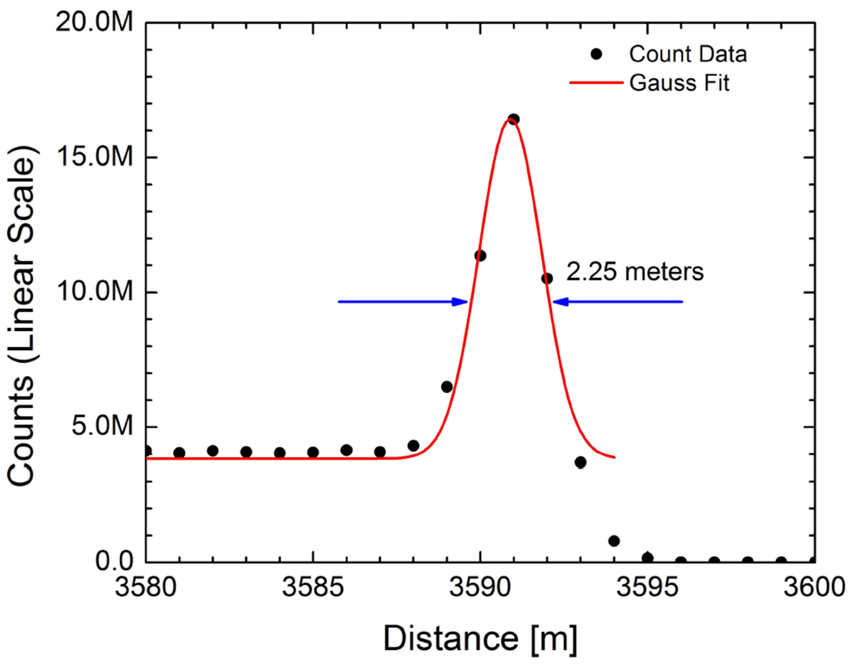


Figure 3.2: High Dynamic ν-OTDR spatial resolution determination via the reflective peak analysis. The fitted curve is a Gaussian fit of the data in linear scale.

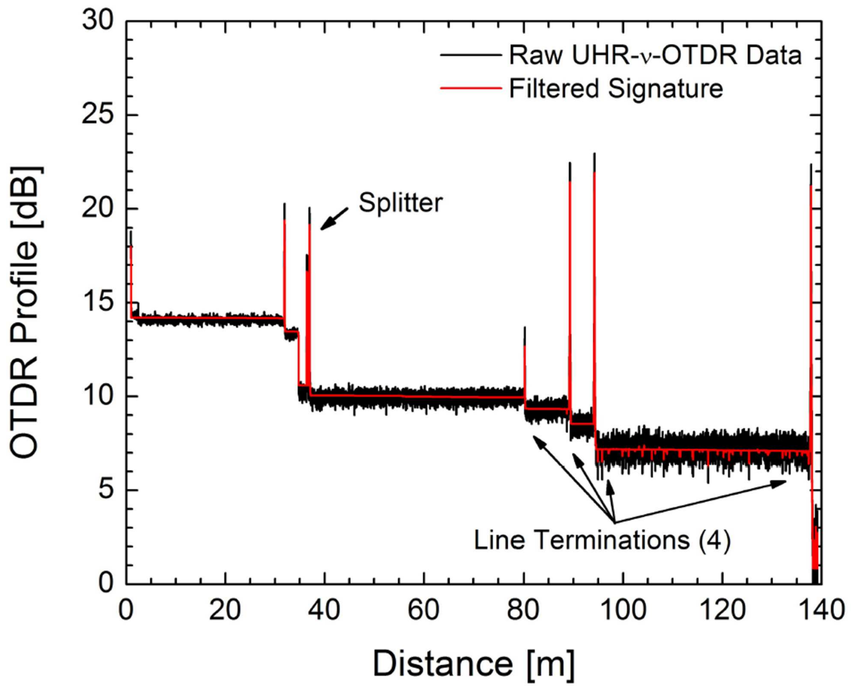


Figure 3.3: 1x4 PON architecture. 30.99 meters feeder fiber + (101.45, 43.88, 52.6, 57.23) meters user lines. Losses and reflections are related to physical connectors (FC-APC) of the optical devices.

Fig. 3.3 presents the monitoring result of a 1x4 Short-PON using the High Resolution ν-OTDR. Here, the 30.99 m feeder fiber is connected to four user lines by a 1x4 splitter, which induces a 3 dB apparent loss in the OTDR profile (6 dB to each link). The measured lengths of the user fibers are 101.45, 43.68, 52.60 and 57.23 m. Both reflective and non-reflective events are present in the user lines as line termination connectors and a splitter loss, respectively. A detailed detection of the ℓ_1 Trend Filter is also presented, the filter detected all reflections associated to connectors.

In Fig. 3.4, we present the analysis of a reflective peak at the longest line termination. The data in linear scale is fitted by a Gaussian curve and the Full Width at Half Maximum criterion is used to determine the achievable spatial resolution of 3.5 cm.

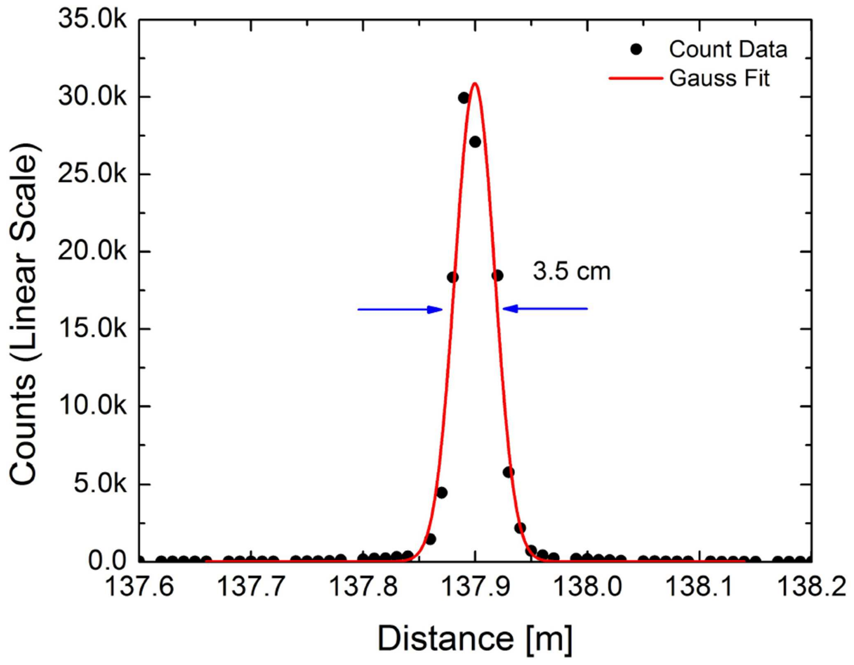


Figure 3.4: HR-ν-OTDR spatial resolution determination via the reflective peak analysis. The fitted curve is a Gaussian fit of the data in linear scale.

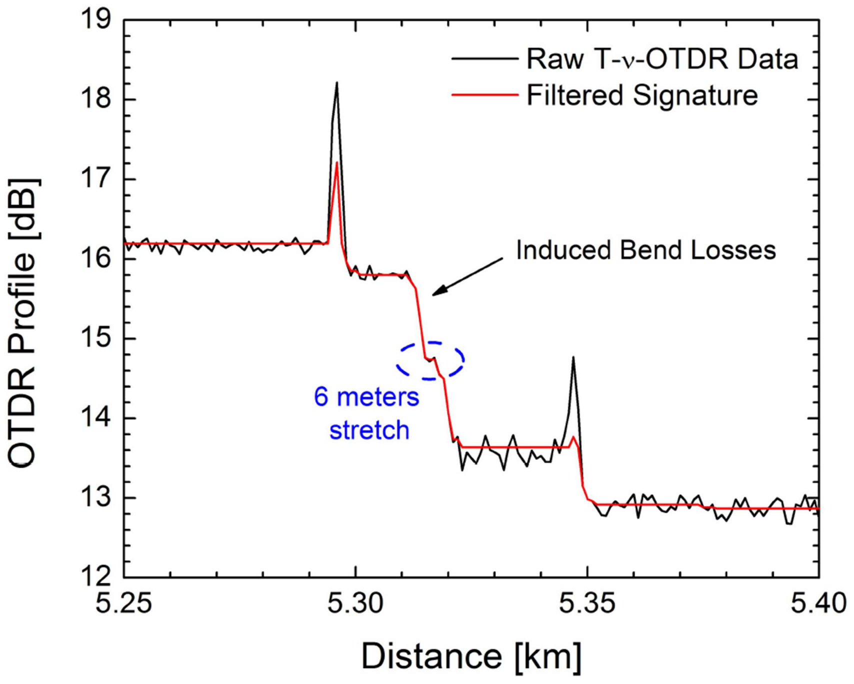


Figure 3.5: Detailed image of the filter's sensibility of two induced fiber faults separated by a 6 meters fiber stretch using the HR-ν-OTDR.

In Fig. 3.5, the detailed detection of two consecutive induced bend losses separated by a 6 meters fiber in the HD-ν- OTDR profile is depicted. The result shows the high sensibility of the digital signal processing filter even after a great loss such as the 15 dB one corresponding to the 1x32 optical splitter. Fig. 3.6 also shows a detailed detection of the ℓ_1 Trend Filter but in the HR-ν-OTDR. The filter detected all reflections associated to connectors. These results confirm the applicability of the proposed techniques along with the ℓ_1 Trend Filter.

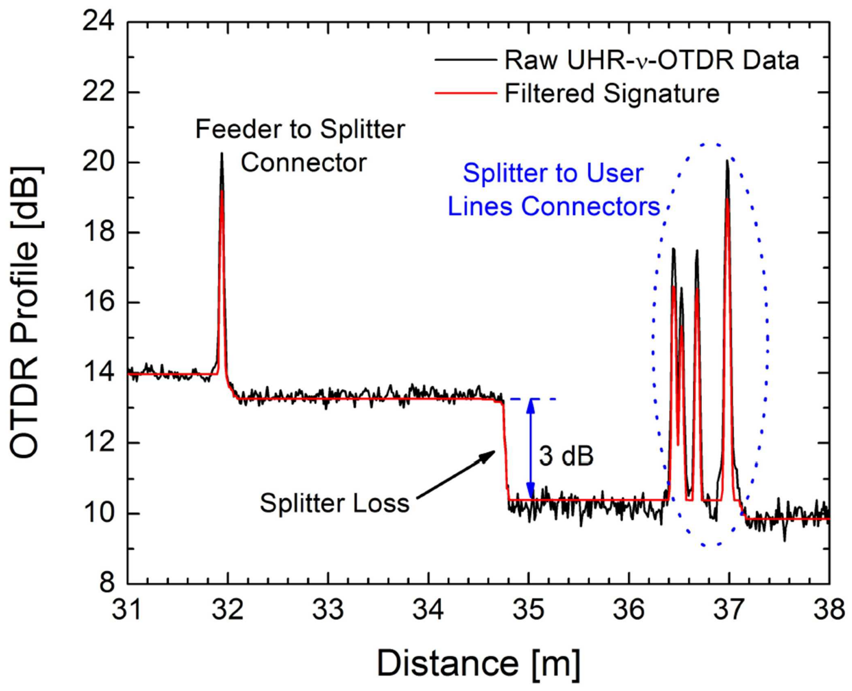


Figure 3.6: HR-ν-OTDR spatial resolution determination via the reflective peak analysis. The fitted curve is a Gaussian fit of the data in linear scale.

3.2 Proposal

In this work the development of a structure that aims to unite the high dynamic range of the HD-ν-OTDR to the high resolution provided by the HR-ν-OTDR with the objective of accurately and quickly finding faults in a fiber is presented. In addition, through the use of a trend filter, it was possible to create an automatic system. The schematic shown in Fig. 3.7 depicts the automatic steps of the system to inspect an optical link. First, the system uses the HD-ν-OTDR to obtain a fiber profile whose resolution is ≈ 5 m. Despite the limited resolution, the acquisition rate is quite high in a 20km link, for example, the system allows the determination of the last point of the profile with a signal-to-noise ratio of 10 dB in just under two minutes [44]. Then the trend filter estimates the positions that may present faults and sends them as events list to the HR-ν-OTDR. The signal processing step is also quite accelerated, with extremely accurate results in just under a minute [47]. The HR-ν-OTDR, in turn, analyzes only the regions around the points of interest. Although the acquisition is time-consuming, the extremely significant reduction of the total fiber length to be inspected makes the results be acquired with palatable times.

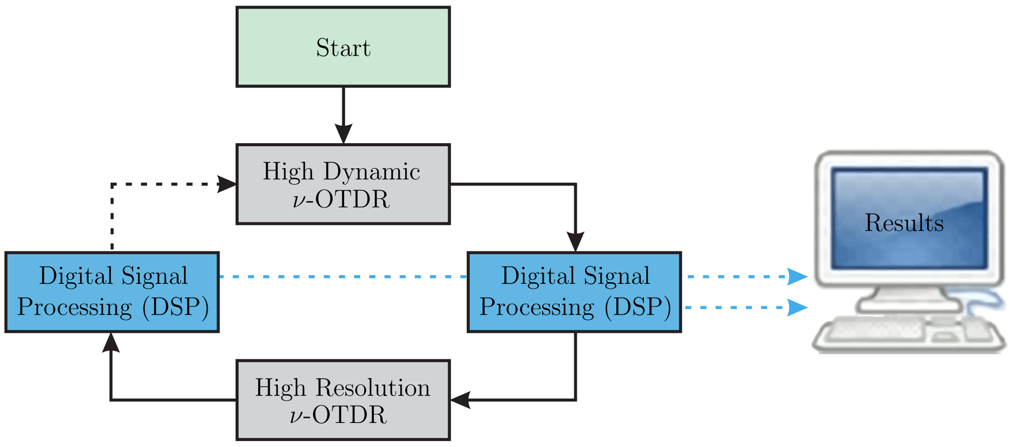

Start
High Dynamic ν-OTDR
Digital Signal Processing (DSP)
Digital Signal Processing (DSP)
Results
High Resolution ν-OTDR

Figure 3.7: Flowchart of the Automatic ν-OTDR.

It is evident from the schematic of Fig. 3.7 that the management system of the previously described methods is essential for the functionality of the proposal. This system must be able to interface with: the HD-ν-OTDR structures, the HR-ν-OTDR structures, the fiber profile analysis algorithm. In addition, in order to reduce redundant costs and re-utilizations with equipment, it is interesting that structures that can be adapted by both OTDR techniques are reused. In Fig. 3.8, the basic schematic of the proposal is presented, where it is already notable that one of the most important equipment, the SPAD, is used by both techniques. Note also that the sources of pulses are independent, since they have completely different characteristics. In order for both sources to operate within the same system, an optical switch is connected to its outputs and, through a system command, the pulse of the HD-ν-OTDR or the HR-ν-OTDR is Injected into the fiber. It is important to note that the use of an optical switch (JDSU SR12) generates an insertion loss of $\approx 2dB$, so a new approach in which no optical switches are used is proposed as a future work.

Although used by both techniques, the SPAD does not operate in the same way in each one. The SPAD operates in the free-gating mode for the HR-ν-OTDR, and, in external-gating for the HD-ν-OTDR. In the free-gating mode, the enable pulse passes an input block that checks for a Dead Time state. If not, a detection window with the same duration as the enable pulse is generated. If a detection occurs during this window, a detection pulse is generated and the quenching process starts. During the quenching process the G-APD stops responding to enable pulses, this time period in which the

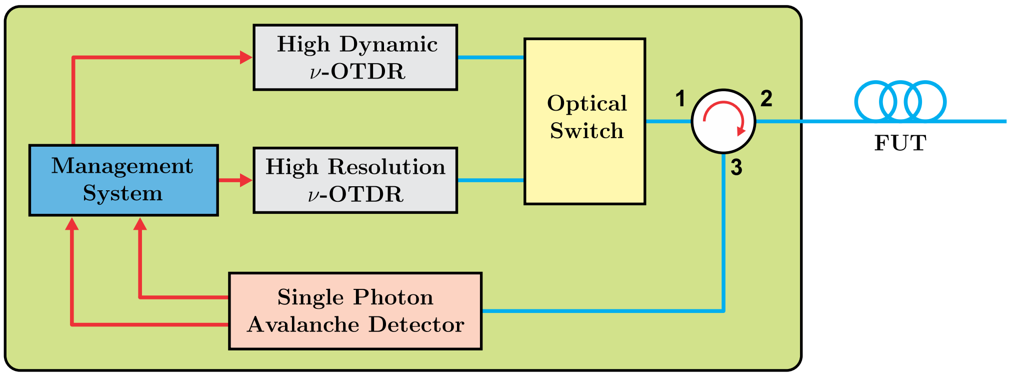


Figure 3.8: Block diagram showing the simplified circuit of Automatic ν-OTDR.

photodetector becomes dead is known as Dead Time [55]. On the other hand, in the external gating operation mode, the detector responds to the trigger port pulses that generate a user-determined width detection window.

It is, therefore, necessary to adapt the device to operate in each of the regions when necessary. Unfortunately, the SPAD model used ("ID210 NIR 100MHz Gated Detector" from the company ID Quantique [56]) does not allow it to be remotely controlled. Because of this, and inspired by the use of the optical switch, a radio frequency switch has been included in the assembly in order to force the detector to work in the correct mode. Its mode of operation has been set to free-gating and, in the case of HR-ν-OTDR, the original pulse is sent to the detector's enable port, i.e., there is no change. For the HD-ν-OTDR, a 4 ns pulse, is generated by a high-voltage pulse generator (AVM-1) from AVTech Electrosystems [57], is sent to the "Enable" by simulating the external gating mode detection window. The management system is responsible for acting on the radio frequency switch and selecting the time at which each of the pulses will be sent to the detector.

The experimental setup of the Automatic ν-OTDR is shown in Figure 3.9. It can be seen that all the components that are part of the HD-ν-OTDR are above the SPAD's line and that the components below the SPAD's line are part of the HR-ν-OTDR. The basic principle of operation is the same one presented previously, taking some modifications. The main modifications are the inclusion of an optical switch, the inclusion of a radio frequency switch, an EDFA and a tunable filter. The control of the optical and radiofrequency switches is performed by the FPGA which selects which ν-OTDR will be active.

The principle of operation of the High Dynamic ν-OTDR is presented below. The Tunable Laser Source (TLS) is a tunable continuous wavelength laser source which wavelength can be swept from 1500 to 1640 nm. The first SOA boosts the optical signal inside a narrow window of 60 ns with a 2 A

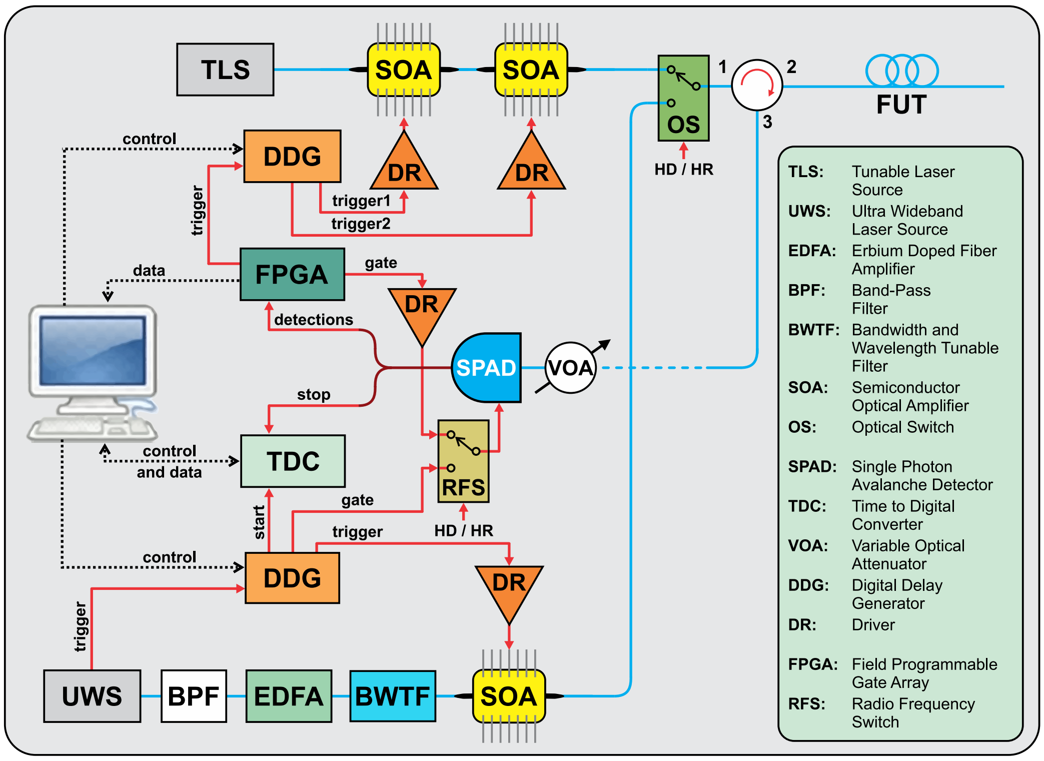

TLS
SOA
SOA
OS
1
2
3
FUT
control
DDG
DR
DR
trigger
trigger1
trigger2
HD / HR
data
FPGA
gate
DR
detections
SPAD
VOA
stop
control and data
TDC
RFS
gate
start
trigger
HD / HR
control
DDG
DR
trigger
UWS
BPF
EDFA
BWTF
SOA
TLS: Tunable Laser Source
UWS: Ultra Wideband Laser Source
EDFA: Erbium Doped Fiber Amplifier
BPF: Band-Pass Filter
BWTF: Bandwidth and Wavelength Tunable Filter
SOA: Semiconductor Optical Amplifier
OS: Optical Switch
SPAD: Single Photon Avalanche Detector
TDC: Time to Digital Converter
VOA: Variable Optical Attenuator
DDG: Digital Delay Generator
DR: Driver
FPGA: Field Programmable Gate Array
RFS: Radio Frequency Switch

Figure 3.9: Automatic ν-OTDR experimental setup.

peak current driver creating high power narrow light pulses. The second SOA is driven by a 600 mA current pulse with 10 ns width and works as a gainless optical switch shaping the width of the optical pulse down to 10 ns. The tandem configuration of the SOAs guarantees a high extinction rate, contributing to the overall dynamic range of the detections.

The FPGA send trigger pulses to the Digital Delay Generator (DDG) which then drives both SOAs. The DDG drives each SOA respecting the optical delay between both to ensure perfect synchronization. The light pulse is sent down the fiber and the backscattered light is detected during the 4 ns gate window generated by pulse shaping the 10 ns pulse from the FPGA. The Variable Optical Attenuator (VOA) guarantees that the SPAD is kept at linear regime.

The principle of operation of the High Resolution ν-OTDR is quite similar to that of the High Resolution ν-OTDR with the difference in the pulse generation method and in the acquisition process. The 115 fs wide pulses from an Ultra Wideband Laser Source (UWS) passes through a narrow bandpass filter and are then amplified by an Erbium Doped Fibre Amplifier (EDFA). The bandpass filter ensures that the EDFA will amplify only the selected wavelength and thus not wasting optical power. The tunability is guaranteed by the Tunable Filter which exhibits an ultra-sharp roll-off ideal for DWDM monitoring. The pulses are then modulated by a SOA in order to reduce the pulse repetition rate and satisfy the condition of one light pulse traversing the fiber at a time. The enabling pulses of the SOA are generated by a Digital Delay Generator (DDG), which is triggered by the UWS. The same DDG also triggers the start of the Time-to-Digital Converter (TDC) and the Single Photon Avalanche Detector (SPAD) with varying delays such as to cover the whole fiber length. The detection pulses are sent to the stop of the TDC. As before, the VOA guarantees the linear regime of operation of the SPAD.

3.2.1 User interface

The Graphical User Interface (GUI) of the Automatic ν-OTDR is based in Python^{TM}. This interface is divided into 3 subinterfaces: High Dynamic ν-OTDR, High Resolution ν-OTDR and Automatic ν-OTDR

Automatic OTDR

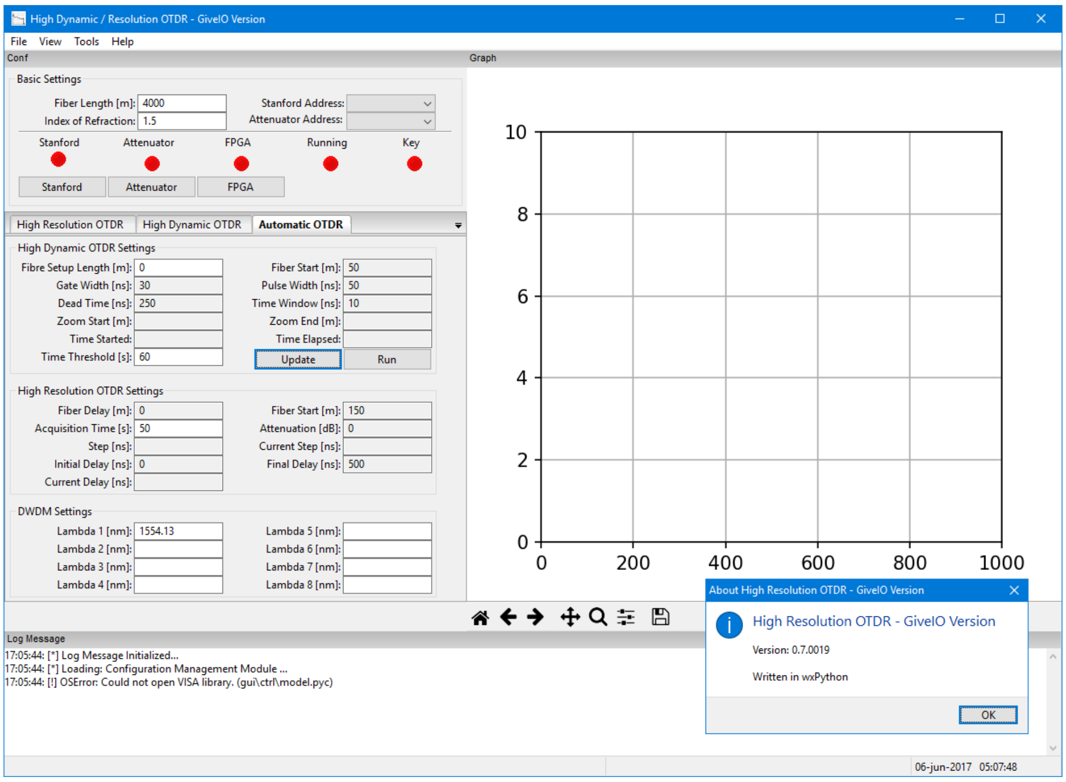
In this interface, the acquisition parameters are entered and then sent to the FPGA via a USB interface. The FPGA is responsible for generating the train of gates that are separated by the dead time of the SPAD and controlling the SOAs so that only one light pulse is inside the optical fiber. After the acquisition time, the digital signal processing takes place and in the end a event list with the fiber fault's position is generated. The analysis of these points is performed sequentially through the High Resolution ν-OTDR and the results are displayed. There is also the possibility of defining several wavelengths to characterize a DWDM network. The GUI is presented in Figure 3.10 and the description of the parameters is shown below:

Common Settings

- *Fiber Length*: determines the maximum length of optical fiber to be measured.
- *Index of Refraction*: determines the fiber's index of refraction.

High Dynamic ν-OTDR Settings

- *Fiber Setup Length*: determines the start of the optical fiber.
- *Fiber Start*: takes into account the length of the launch fiber.
- *Gate Width*: determines the acquisition window of the SPAD.
- *Pulse Width*: determines the light pulse width.
- *Dead Time*: determines the dead time of the SPAD.



Basic Settings

Fiber Length [m]: 4000
Index of Refraction: 1.5
Stanford Address:
Attenuator Address:
Stanford ● Attenuator ● FPGA ● Running ● Key ●

High Dynamic OTDR Settings

Fibre Setup Length [m]: 0
Gate Width [ns]: 30
Dead Time [ns]: 250
Zoom Start [m]:
Time Started:
Time Threshold [s]: 60
Fiber Start [m]: 50
Pulse Width [ns]: 50
Time Window [ns]: 10
Zoom End [m]:
Time Elapsed:

High Resolution OTDR Settings

Fiber Delay [m]: 0
Acquisition Time [s]: 50
Step [ns]:
Initial Delay [ns]: 0
Current Delay [ns]:
Fiber Start [m]: 150
Attenuation [dB]: 0
Current Step [ns]:
Final Delay [ns]: 500

DWDM Settings

Lambda 1 [nm]: 1554.13
Lambda 2 [nm]:
Lambda 3 [nm]:
Lambda 4 [nm]:
Lambda 5 [nm]:
Lambda 6 [nm]:
Lambda 7 [nm]:
Lambda 8 [nm]:

Log Message

```
17:05:44: [*] Log Message Initialized...
17:05:44: [*] Loading: Configuration Management Module ...
17:05:44: [!] OSError: Could not open VISA library. (gui\ctrl\model.pyc)
```

About High Resolution OTDR - GiveIO Version

High Resolution OTDR - GiveIO Version

Version: 0.7.0019

Written in wxPython

06-jun-2017 05:07:48

Figure 3.10: GUI of the Automatic ν-OTDR.

- *Time Window*: determines the acquisition granularity, must be greater than 10 ns.
- *Zoom Start*: if set, determines the initial position of the fiber that will be characterized.
- *Zoom End*: if set, determines the final position of the fiber that will be characterized.
- *Time Threshold*: determines the acquisition time.

High Resolution ν-OTDR Settings

- *Fiber Delay*: determines the start of the optical fiber.
- *Fiber Start*: takes into account the length of the launch fiber.
- *Acquisition Time*: determines the acquisition time.
- *Attenuation*: the attenuation imposed by VOA is calculated on the fly but can also be manually set.
- *Initial delay*: determines the initial position of the fiber that will be characterized.
- *Final delay*: determines the final position of the fiber that will be characterized.

– *Detector Pulse Width*: is the electrical pulse width send to the gate of the SPAD.

– *LeftCut* and *RightCut*: determines the transient region inside a detection window of the SPAD, every detection inside this transient region will be discarded, see Fig. 2.18.

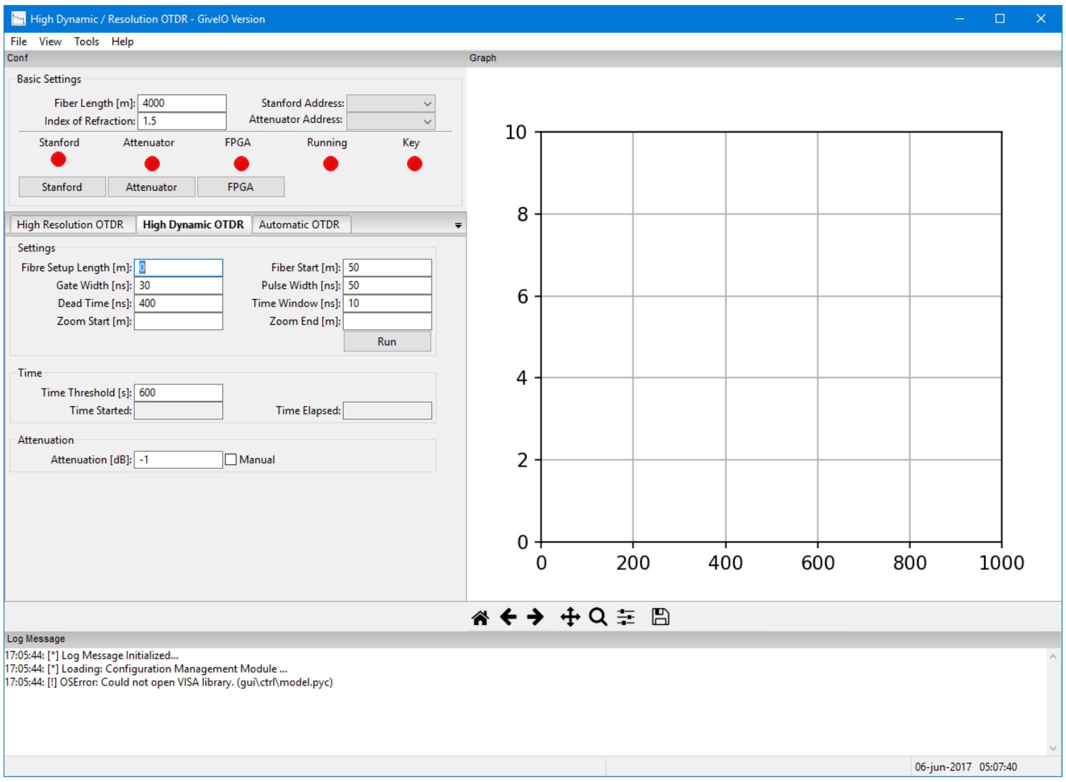
– *Step*: determines the stable region inside a detection window of the SPAD, each acquisition window has the width of this setting.

DWDM Settings

– *Lambda*$_i$: are the DWDM wavelengths, also the channel spacing can be changed through the Tools menu.

High Dynamic OTDR

In this interface, see Fig. 3.11, the acquisition parameters are entered and then sent to the FPGA via a USB interface. The FPGA is responsible for generating the train of gates that are separated by the dead time of the SPAD and controlling the SOAs so that only one light pulse is inside the optical fiber, as explained earlier. The description of the High Dynamic ν-OTDR parameters has already been shown previously.



Fiber Length [m]: 4000
Index of Refraction: 1.5
Stanford Address:
Attenuator Address:
Fibre Setup Length [m]: 0
Gate Width [ns]: 30
Dead Time [ns]: 400
Zoom Start [m]:
Fiber Start [m]: 50
Pulse Width [ns]: 50
Time Window [ns]: 10
Zoom End [m]:
Time Threshold [s]: 600
Time Started:
Time Elapsed:
Attenuation [dB]: -1

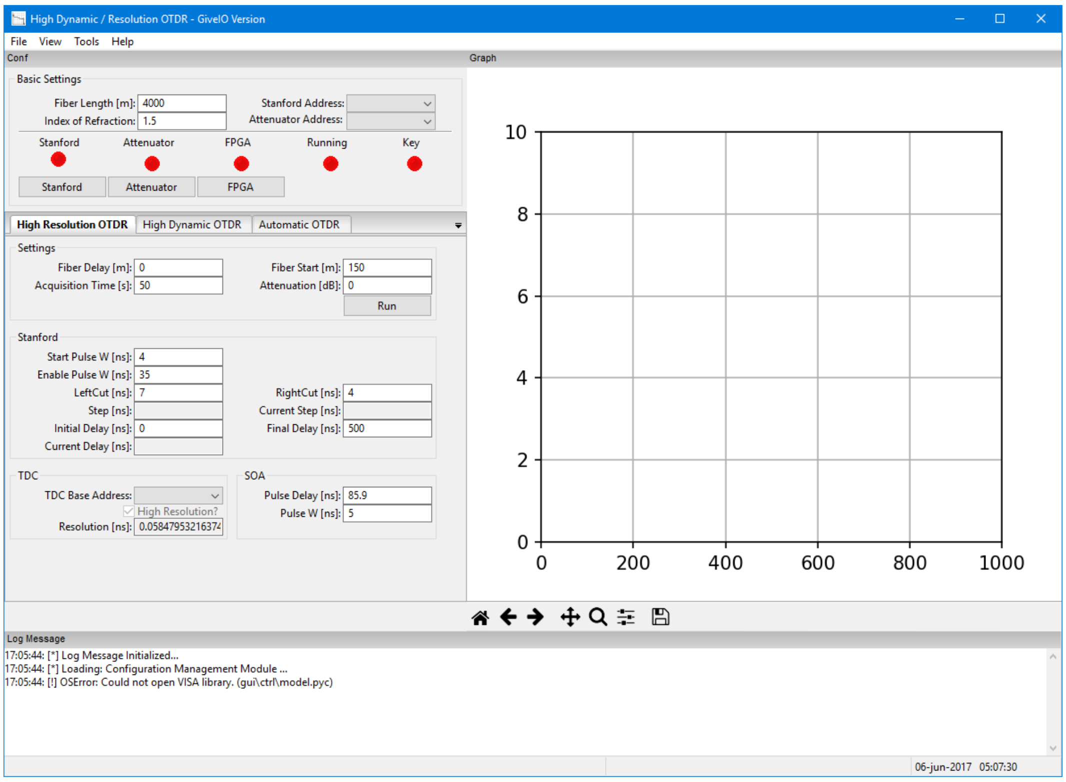
```
17:05:44: [*] Log Message Initialized...
17:05:44: [*] Loading: Configuration Management Module ...
17:05:44: [!] OSError: Could not open VISA library. (gui\ctrl\model.pyc)
```

06-jun-2017 05:07:40

Figure 3.11: GUI of the High Dynamic ν-OTDR.

High Resolution OTDR

In this interface, see Fig. 3.12, the acquisition parameters are entered and then the DDG Prescaler is set so that the condition of only light pulse inside the fiber is met. The DDG is responsible of starting the TDC and sending varying enable pulses to the SPAD in order to cover the desired fiber length. When a detection occurs, a stop pulse is sent to the TDC and the OTDR trace is drawn.



Basic Settings

Fiber Length [m]: 4000
Index of Refraction: 1.5
Stanford Address:
Attenuator Address:

Settings

Fiber Delay [m]: 0
Acquisition Time [s]: 50
Fiber Start [m]: 150
Attenuation [dB]: 0

Stanford

Start Pulse W [ns]: 4
Enable Pulse W [ns]: 35
LeftCut [ns]: 7
RightCut [ns]: 4
Step [ns]:
Current Step [ns]:
Initial Delay [ns]: 0
Final Delay [ns]: 500
Current Delay [ns]:

TDC

TDC Base Address:
High Resolution? ✓
Resolution [ns]: 0.05847953216374

SOA

Pulse Delay [ns]: 85.9
Pulse W [ns]: 5

```
17:05:44: [*] Log Message Initialized...
17:05:44: [*] Loading: Configuration Management Module ...
17:05:44: [!] OSError: Could not open VISA library. (gui\ctrl\model.pyc)
```

06-jun-2017 05:07:30

Figure 3.12: GUI of the High Resolution ν-OTDR.

3.3 Equipment description

3.3.1 Tunable Laser Source (TLS)

The Yenista TUNICS Plus CL is a tunable laser source with high output power, up to 20 mW, and broad spectral coverage, with an optical spectrum width of 1500 to 1620 nm [58]. Among its features, the 1 pm resolution, 5 pm wavelength stability and the 150 kHz linewidth are highly desirable. Also, one can remote control the TLS through GPIB and RS232. The specifications of the TLS are presented in Table 3.1.

Parameter	Min.	Max.	Unit
Optical Spectrum Range	1500	1620	nm
Absolute Wavelength Accuracy	$\pm$ 40		pm
Wavelength Stability	$\pm$ 5		pm
Wavelength Setting Resolution	1		pm
Tuning Speed	-	1	s
Spectral Width (FWHM)	150		kHz
	$>$ 100		MHz
Power Stability	$\pm$0.01		dB/h

continued from previous page

Parameter	Min.	Max.	Unit
Output Isolation	35		dB
Return Loss	60		dB
Optical Conector		FC/APC	
Optical Fiber		SMF	

Table 3.1: Specifications of the Yenista TUNICS Plus CL [58].

3.3.2 Ultra-Wideband Source (UWS)

The Santec UWS-1000H is a high power pulsed laser source that emits what is known as Supercontinuum Light [59]. This pulsed laser source emits pulses at a rate of approximately 6 MHz with an optical spectrum width of 1200 to 1700 nm, see Fig. 3.13. Although the pulse width is one of the determining factors in OTDR temporal resolution, the UWS pulses with approximately 115 femtoseconds [43, 50] must pass a narrow bandpass filter in order to achieve tunability. The specifications of the UWS are presented in Table 3.2.

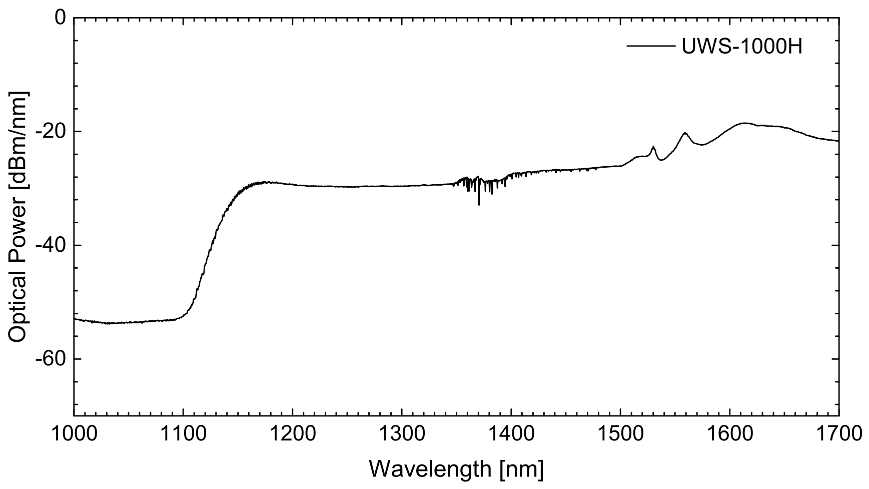


Figure 3.13: The UWS-1000H optical spectrum measured using an Optical Spectrum Analyzer.

Parameter	Min.	Max.	Unit
Optical Spectrum Range	1200	1700	nm
Spectral Power Density	-20	-	dBm/nm
Optical Spectral Stability	-	± 0.1	dB
Repetition Rate	6 ±0.5		MHz
Optical Conector		FC/APC	

continued from previous page

Parameter	Min.	Max.	Unit
Optical Fiber		SMF	

Table 3.2: Specifications of the Santec UWS-1000H [59].

3.3.3 Bandwidth and Wavelength Tunable Filter (BWTF)

The Alnair Labs CVF-300CL is a bandwidth and wavelength tunable filter with bandwidth ranging from 0.03 to 3 nm and wavelength ranging from 1525 to 1610 nm. The filter exhibits an ultra-sharp roll-off of 12 dB/GHz making it ideal for ideal for DWDM channel selection and removing ASE noise [60]. The specifications of the CVF-300CL are presented in Table 3.3.

Parameter	Min.	Max.	Unit
3dB Bandwidth Tunability	0.03	3	nm
Filter-Edge Roll-Off	1000		dB/nm
Center Wavelength Tunability	1525	1610	nm
Wavelength Accuracy	$< \pm 0.05$		nm
Wavelength Repeatability	$< \pm 0.01$		nm
Insertion Loss		6.5	dB
Return Loss	40		dB
Out-of-Band Suppression	40	50	dB
Maximum Input Power	≈ 27		dBm

Table 3.3: Specifications of the Alnair Labs CVF-300CL Tunable Filter [60].

3.3.4 Single Photon Avalanche Detector (SPAD)

The ID Quantique id210 is a single-photon avalanche photodetector whose key component is an InGaAs/InP avalanche photodiode [56]. The id210 can be internally or externally triggered with frequencies up to 100 MHz. One of its key features is its ability to keep a low dark count rate, it is also possible to detect photons with a probability of up to 25% at 1550 nm. Also, timing resolution lower than 200 ps can be achieved [56]. The specifications of the id210 are presented in Table 3.4.

Parameter	Min.	Max.	Unit
Wavelength Range	900	1700	nm
Deadtime Range	0.1	100	μs
Deadtime Step	100		ns
Timing resolution		200	ps
External Trigger Frequency		100	MHz
Internal Trigger Frequency	0.001	100	MHz
Efficiency Range	5	25	%
Operating Temperature	10	30	$°C$
Optical Fiber Type	SMF or MMF		

Table 3.4: Specifications of the ID Quantique id210 Infrared Single-Photon Avalanche Detector [56].

3.3.5 Semiconductor Optical Amplifier (SOA)

The Thorlabs SOA1013SXS is a polarization-independent semiconductor optical amplifier with extinction ratio greater than 60 dB in the 1528 to 1562 nm range. This device operates as a high-speed optical isolation switch and is capable of supporting gain levels up to 15 dBm. Its encapsulation is the industry-standard 14-pin butterfly package with single mode optical fiber pigtails [61]. The specifications of the SOA1013SXS are presented in Table 3.5.

Parameter	Min.	Max.	Unit
Operating Current	-	600	mA
Operating Wavelength	1528	1562	nm
Optical Isolation (@ 0 mA and 1550 nm)	45	-	dB
Extinction Ratio (@ $P_{IN} = -20$dBm and 1550 nm)	60		dB
Switching Speed	1		ns
Max Output Power for CW Input Signal	17		dBm
Max Output Power for Modulated Input Signal	9		dBm
Saturation Output Power (@ -3 dB)	12	14	dBm
Small Signal Gain Across BW (@ $P_{IN} = -20$dB)	10	13	dB
Noise Figure	8.0	9.5	dB

Table 3.5: Specifications of the Thorlabs SOA1013SXS [61].

This SOA is used in the High-Resolution ν-OTDR and activated by an AVTech that generates 4 ns wide pulses with currents close to the limit of 600 mA. Even though the maximum current suggested by the manufacturer is 600 mA, the duty cycle is very small and the repetition rate is less than 30 kHz.

3.3.6 Digital Delay Generator (DDG)

The Stanford DG645m is a 5 output channels digital delay generator with 5 ps resolution and 12 ps jitter [62]. Four of the five delay outputs can be user-defined through the time events: A, B, C, D, E, F, G and H, which represent the rise and fall times, respectively, of each channel. In addition, the fifth output channel (To) produces a pulse starting at t = 0 and remaining high up to 25 ns after the longest delay. Also, the rising and falling edges are lower than 2 ns, as presented in Fig. 3.14.

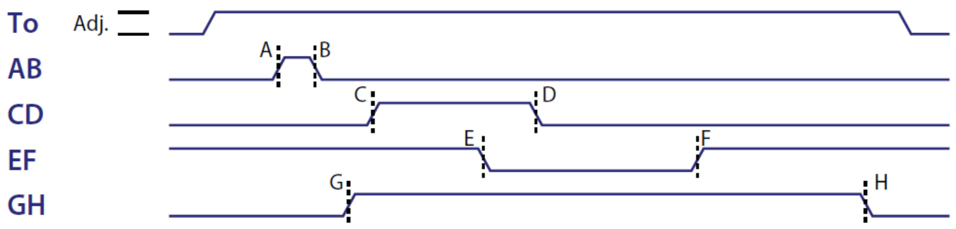


Figure 3.14: The five outputs and its time events: A, B, C, D, E, F, G and H, rising and falling edges. The amplitude of the five output channels can be individually adjusted in steps of 0.01 V from 0.5 to 5.0 V [62].

The amplitude and the offset of the five output channels can be individually adjusted. The amplitude range is 0.5 to 5.0 V with 0.01 V resolution and the offset range is ± 2 V. Moreover, a prescaler can be set to each channel, this permits to use a sub-multiple of the trigger input frequency. The internal rate generator is capable of generating 100 μHz to 10 MHz trigger rates with low jitter. Finally, one can remote control the DG645 through GPIB, Ethernet and RS-232 interfaces. The specifications of the DG645 are presented in Table 3.6.

Parameter	Min.		Max.	Unit
Delay Range	0		2000	s
Resolution		5		ps
Jitter (rms) for				
delay $< 100\,\mu$s		< 20		ps
delay $> 100\,\mu$s		$< 30 +$ (timebase jitter $\times$ delay)		ps
Timebase jitter		10^{-8}		s/s
Rising and falling edges		< 2		ns
Offset		± 2		V
Amplitude	0.5		5.0^{1}	V
Amplitude Resolution		0.01		V

[1] The amplitude of the pulse must be lower than 6 V, that is: amplitude level + offset level < 6.0 V.

Table 3.6: Specifications of the SRS DG645 [62].

3.4 Python

Python^{TM} is a high-level programming language created by Guido van Rossum in 1989 and first released in 1991 [63, 64]. Since the first version (Python v0.9.0), Python already had functions, classes, module system, inheritance, exception handling and its native data types. Python is an interpreted language and multi-paradigm, that is, Python is object-oriented, imperative, functional, procedural, reflective. As stated by Guido, “Python is a simple, yet powerful programming language that bridges the gap between C and shell programming, and is thus ideally suited for ‘throw-away programming’ and rapid prototyping” [65].

Its simple and intuitive syntax allows a development speed without comparison, moreover even who never programmed will be able to make simple programs with Python. As always, there are pros and cons to everything and with Python, it’s not different. Python’s performance is often lower than that of static languages (like C, C++, Fortran, among others) and even than some dynamic languages (Java, Javascript, PHP7, etc.). However, it is possible to write C or C++ code for the most critical parts of the code and compile it as modules for Python. These compiled modules will have performance comparable to C. In addition, there are modules that can dramatically improve Python performance in certain applications, such as Numpy, Numba, Cython, and others.

3.4.1 Why use Python?

There are several reasons for using Python, some of which have already been mentioned earlier, such as being object-oriented, being ideal for rapid prototyping which leads to faster development times and smaller and concise codes, among others.

– *Interpreted language*: after writing a code you just need to run it. Instead of having to compile the code then link the compiled file to turn it into an executable and then run the executable. However, as the file is interpreted line by line, some errors can only be found during runtime [66]. On the other hand, it is easier to debug since you can stop the program and examine the variables or write into the interpreter and check out if it works as desired.

– *Embeddable*: The Python interpreter can be embedded in third-party applications while maintaining the security and speed of a compiled application. The interpreter is rather small occupying less than 7 MB [64, 66, 67].

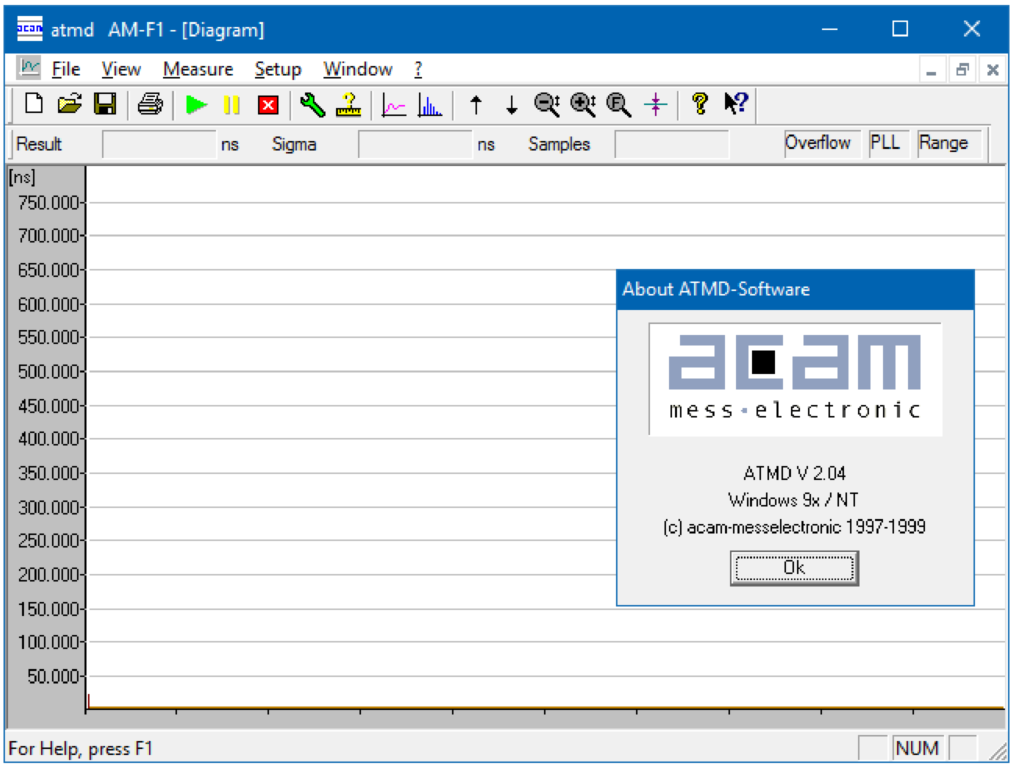
– *Cross Platform*: Unlike other languages, Python is truly cross-platform. Once you write a code, it can run anywhere [64, 66, 67].

– *Support*: In addition to the standard library, there are several tools available and a broad user base. The Python community is quite large and there are many books and sites devoted to Python programming.

In addition, Python is capable of interfacing the computer with the FPGA and all required equipment. The High Resolution ν-OTDR was fully written in LabVIEW [45] and when migrating to Python the acquisition performance of the TDC was much lower then it was necessary to create a C extension to Python that was able to access the TDC directly. On the other hand, the High Dynamic ν-OTDR was already written in Python so few modifications were required.

3.4.2 Python x LabVIEW

The Time-to-Digital Converter (TDC) used in this project was the acam TDC-F1. The description of its operating principle and its characteristics can be seen in Section 2.2. Figure 3.15 shows the Graphical User Interface (GUI) of the software that comes with the TDC. This software allows you to change some basic TDC settings such as the active channels, the threshold voltage of each channel, the edge sensitivity can be adjusted to rising or falling edge, the PLL configuration can be adjusted (resolution of the TDC) and also is able to display the data obtained by the TDC to a graph that can be: a histogram or a line graph.



About ATMD-Software

acam mess•electronic

ATMD V 2.04
Windows 9x / NT
(c) acam-messelectronic 1997-1999

Ok

Figure 3.15: Graphical User Interface tha comes with the Time-to-Digital Converter (acam TDC-F1).

The High Resolution ν-OTDR was fully written in G (LabVIEW's graphical language) [45]. Therefore, the Virtual Instrument Software Architecture (VISA) standard was used to control the TDC. The VISA is an industry standard used to configure, programming, testing, measurement and troubleshooting instrumentation systems. The VISA is an I/O API used to control instruments with Serial, GPIB, PXI, Ethernet or USB interfaces [68].

The communication between the LabVIEW and the TDC is done by the NI-VISA which is the National Instruments$^{\text{TM}}$ implementation of VISA. The NI-VISA is a driver package that controls the PXI interface of the TDC through the use of NI-PAL. National Instruments-Platform Abstraction Layer (NI-PAL) operates as an API to abstract the internal communication bus and operating system [69] by managing the data transmission process. The process by which NI-VISA and NI-PAL interact with the application, operating system, and hardware layers are depicted in Fig. 3.16.

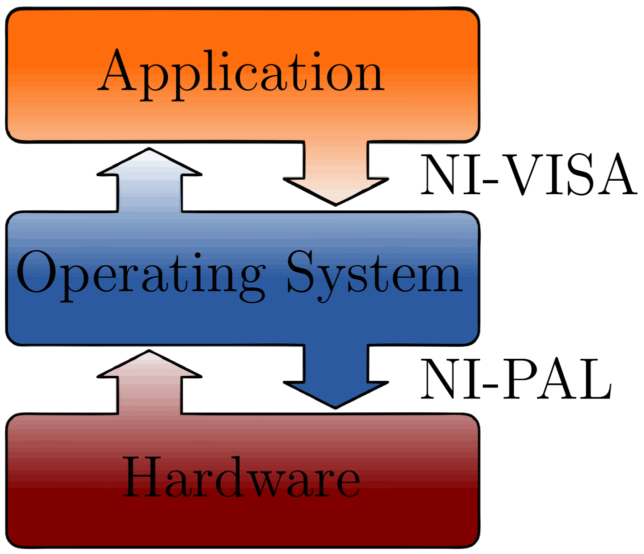


Figure 3.16: This diagram shows how an application interacts with the operating system, which then interacts with the hardware layer.

Although LabView uses VISA to communicate with the TDC, the speed at which this occurs is comparable to that of a compiled code. In Python, on the other hand, it is necessary to use a package called PyVISA. PyVISA is a Python package that acts as a wrapper for the National Instruments VISA library and permits you to control the measurement devices regardless of the interface [70]. This package uses the *ctypes* of the Python's standard library which allows calling functions in DLLs or shared libraries. The *ctypes* is used to call functions within the "visa32.dll" which then communicates with the device. However, the use of *ctypes* has an overhead of at least 3-10 times in relation to an C extension to Python.

Therefore, creating a C extension for Python is an essential step in achieving decent performance. To control the TDC, it is necessary to write and read the ATMD-PCI registers. The C functions: *_inp*, *_inpw*, *_inpd*, and *_outp*, *_outpw*, *_outpd* are used for this purpose. However, these functions require low-level access to an I/O port and the port address. Also, the base address of the ATMD-PCI card can be obtained through the ATMD API. The ATMD API is pretty basic by exposing only 3 functions: GetATMDPCIBoardCount, GetATMDPCIBaseAddr, and EnablePortAccess. In order to provide a practical view of the mentioned devices, in Fig. 3.17 the connection between the AM-F1 and the PCI board through the SCSI-type cable is depicted.

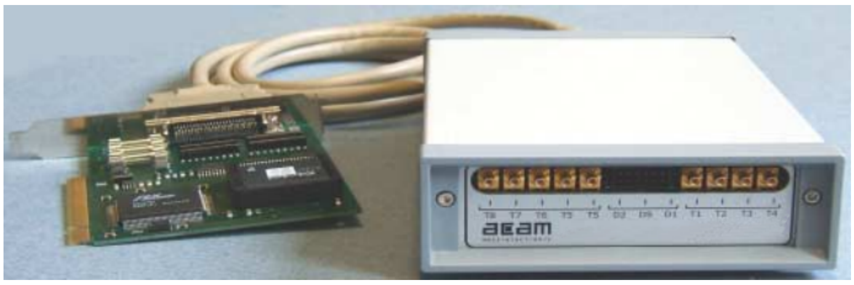


Figure 3.17: The ATMD-PCI, the SCSI-type cable and the TDC, from left to right respectively [20].

In Windows versions that are not part of the NT family (3.x, 9x, and 2000), low-level access to I/O ports was allowed. However, low-level I/O functions in user-mode are prohibited in the Windows NT family. The main reason for this restriction is that unrestricted access to the I/O ports would allow someone to turn off all the system interrupts, to lead a program to unexpected behavior, to damage the system, and other unwanted effects. As explained by Dale Roberts, “The Windows NT architecture requires that all hardware be accessed via kernel-mode device drivers” [71].

So, the “portio” extension to Python was created to configure and acquire TDC data through Python. Through the use of the kernel-mode device driver “giveio.dll”, it is possible to full user-mode access to all I/O ports. Both files, “portio.pyd” and “giveio.dll” [71], are used in the Automatic ν-OTDR. The source code of this C extension to Python can be found in Appendix A.

3.4.3 Python Server

In the context of Internet of Things (IoT), it is possible to replace the computer in Fig. 3.9 with a Raspberry Pi or another single-board computer that is capable of controlling the equipment and sending the data to the network operator over the Internet or to a centralized server that will do the digital signal processing (DSP) of the data and warn the network operator if a fiber fault is found. It is interesting to note that the use of a single-board computer aid in the creation of a low-cost embedded system.

However, when using an single-board computer the need for a centralized server becomes evident because memory usage can grow rapidly depending on the fiber length to be analyzed. A Python server was created to perform digital signal processing and return a list with the fault locations. To do this, it was necessary to install two Python modules: PyJulia, a Python wrapper for the library libjulia that performs the Julia-Python interface, and MATLAB Engine

API for Python, a Python package for calling MATLAB, since the DSP is done in MATLAB or Julia.

The server was created using the sockets, json, base64, and threads modules, so the server is able to talk to multiple clients at the same time. The use of json and base64 modules allows the exchange of messages between the server and the client and the possibility of sending binary files, such as images, among others. The digital signal processing algorithms used are described below.

3.5 Digital Signal Processing / L1 or LB filter?

Automatically identifying the position of a fault in a OTDR profile is of great interest since it relieves the necessity of an operator who constantly examines the resulting fibre profile in order to provide the network manager with the link's conditions [47]. In [72], the authors provide a thorough comparison between signal processing techniques which can be employed in the mentioned task. The comparison points are precision, accuracy and processing time, since the goal is to identify the technique which delivers the best estimate of the faults positions and magnitudes within the shortest time period.

Furthermore the authors of [72] propose a new methodology to tackle the problem and show that it is the best candidate given the comparison points mentioned before. Therefore, the ℓ_1 Adaptive Filter, is considered as a signal processing tool to be employed in the flowchart, see Fig. 3.7. The principle of operation of the ℓ_1 Adaptive Filter is to provide an over-complete set of candidates which are known to explain the signal of interest, even though the majority of them will not be used. The algorithm attempts to elect the minimum amount of candidates with the best approximation of the profile. Since the candidates are directly associated with the fault's positions, the result is an event list containing all positions of faults [72].

Even though its success, the memory usage of the ℓ_1 Adaptive Filter is very high since the over-complete dictionary must be loaded into the computer's memory. It is easy to see that, with a high spatial discretization necessary for reliable results and a long fiber link, the memory usage escalates very fast. At the same time, a new approach to the problem has been developed, which makes use of the Linearized Bregman Iterations [73]. This algorithm solves the exact same problem, but does not make use of the over-complete dictionary; the only stored input is the fiber profile. Therefore, embedding such algorithm in an FPGA is straightforward.

3.6 Results

3.6.1 Final Setups

3.6.1.1 High Dynamic OTDR

In order to analyze the High-Dynamic ν-OTDR performance, some measurements were made changing the laser source. The first measure was made using only the Semiconductor Optical Amplifiers (SOAs), then a second measure was made using the SOAs and the Tunable Laser Source (TLS) which has a ultra-narrow linewidth of a few kilohertz (150 kHz, typically). The result are shown in Figure 3.18, where a 36 km fiber link was measured. The 36 km long fiber link was composed of three almost identical fibers with 12 km each.

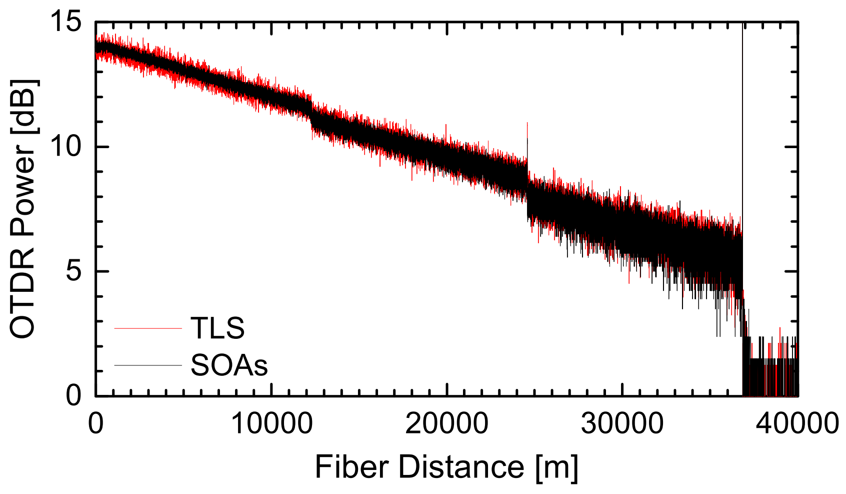


Figure 3.18: OTDR traces using the High Dynamic ν-OTDR. The trace in which the Tunable Laser Source is powered on shows the phenomenon of Coherent Rayleigh Noise.

One can say that the black measure is less noisy than the red measure where the Coherent Rayleigh Noise (CRN) is more evident since it can severely degrade the performance of OTDRs. The Coherent Rayleigh Noise or Coherent Fading Noise is a major problem for OTDRs that use narrow linewidth laser sources and is common in coherent detection but it can appear even when intensity detection is used (incoherent) [74]. The CRN are the fluctuations in backscatter intensity [75, 76] that appear when the coherence length is greater than the pulse length [76]. This occurs due to the interference caused by the superposition of several light waves arriving at the detector with random phase

[75,76]. The coherence length is given by

$$L_c = \left|\frac{c}{n \cdot \Delta\nu}\right| = \left|\frac{\lambda^2}{n \cdot \Delta\lambda}\right| \tag{3-1}$$

where λ is the wavelength, $\Delta\nu$ is the bandwidth, $\Delta\lambda$ is the linewidth, c is the speed of light and n is the refractive index. However, Villafani *et al.* showed that this phenomenon can be mitigated by the use of a broadened optical source or sweeping its optical frequency [74]. Figure 3.19 shows that sweeping the optical frequency from 1554.13 to 1554.93 nm mitigated the Coherent Rayleigh Noise phenomena.

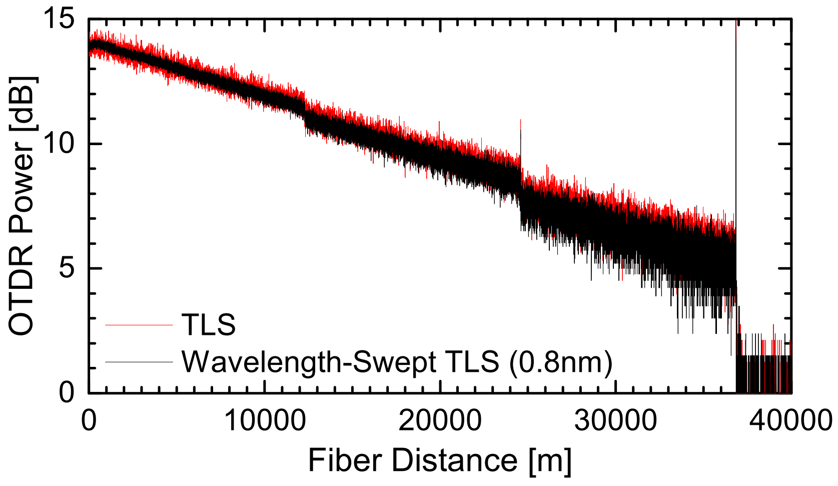
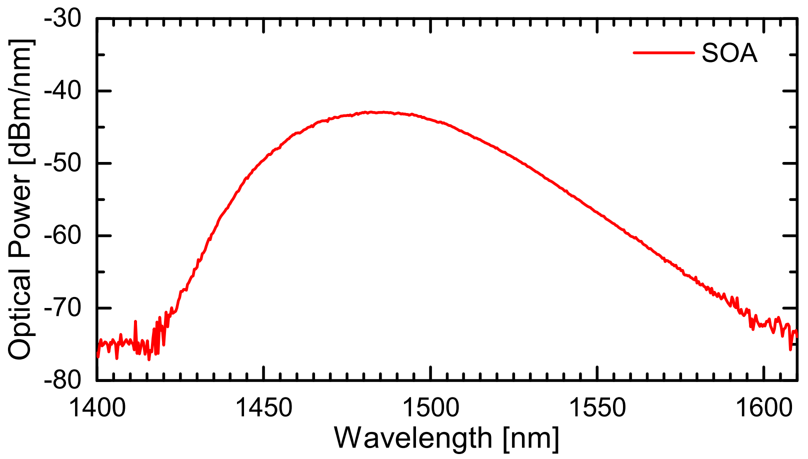
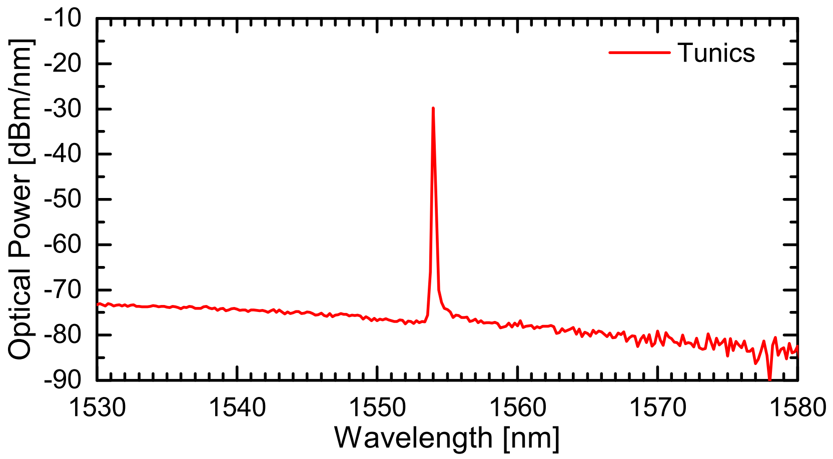


Figure 3.19: OTDR traces using the High Dynamic ν-OTDR. The TLS was swept from 1554.13 to 1554.93 nm.

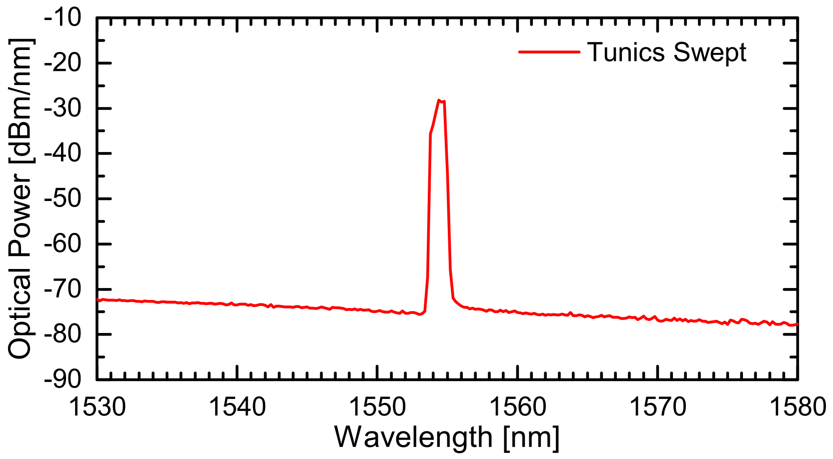
As shown in Figure 3.9, the first SOA, right after the Tunable Laser Source, boosts the optical signal inside a narrow window of 60 ns this guarantees optimal power usage while the second SOA acts as a gainless optical switch. The second SOA is driven by a pulse with 10 ns width. Once the fiber length is set, the FPGA sends trigger pulses to the Digital Delay Generator (DDG) which then drives both SOAs respecting the optical delay between both to ensure perfect synchronization. Subsequently, the spectral power density of each configuration was analyzed through an Optical Spectrum Analyzer (OSA). It is important to note that all of the spectral power density measurements shown below were performed on the pulsed regime, i.e., the Duty Cycle of the SOAs was quite low. Considering a 40 km fiber, the round trip time would be 400 us, and the width of the trigger pulse of the second SOA, we would have a Duty Cycle of approximately 0.000025. Therefore, we should add 46 dB to the spectral power densities shown below.



(a): Spectral power density of the Semiconductor Optical Amplifier when no input is applied (ASE spectrum).



(b): Spectral power density measured after the SOAs when the Tunable Laser Source is powered on.



(c): Spectral power density measured after the SOAs when the Tunable Laser Source is swept from 1554.13 to 1554.93 nm.

Figure 3.20: Spectral power density measurements performed using an Optical Spectrum Analyzer (Anritsu MS9710C) [77].

In Figure 3.20 (a), the SOA's ASE spectrum is shown with a peak power of -43 dBm which adding up to 46 dB gives a total of 3 dBm. In Figure 3.20 (b), the Tunics spectrum is shown with a peak power of 17 dBm (-29 dBm + 46 dB). Finally, in Figure 3.20 (c), the Tunics spectrum is shown while the optical frequency is swept from 1554.13 nm to 1554.93 nm using the "Max Hold" function of the OSA, the peak power is 18 dBm (-28 dBm + 46 dB). It worth remember that the maximum output power of this SOAs is around 18 dBm.

3.6.1.2
High Resolution OTDR

Similarly to the tests performed in High-Dynamic ν-OTDR, some measurements were made using the High-Resolution ν-OTDR to analyze its performance. The measurements carried out involve the characterization of the: pulsed laser source (the UWS emits light pulses every ≈ 174 ns), Band-Pass Filter, EDFA, and of the Bandwidth and Wavelength Tunable Filter, as well as the way they are connected to each other. The optical spectrum of the Ultra-Wideband Source is presented in Figure 3.21.

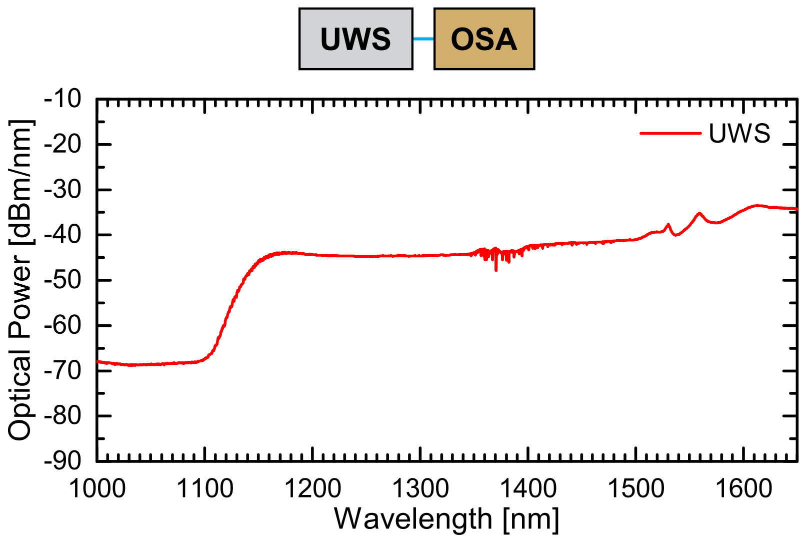
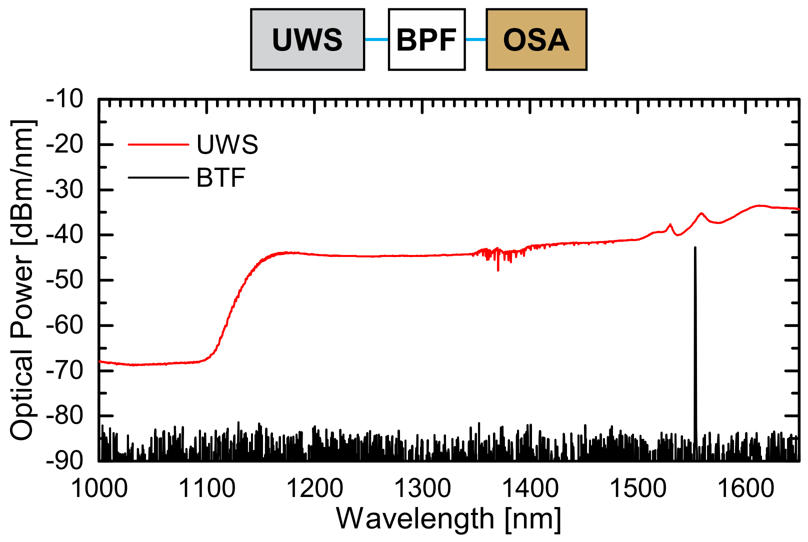
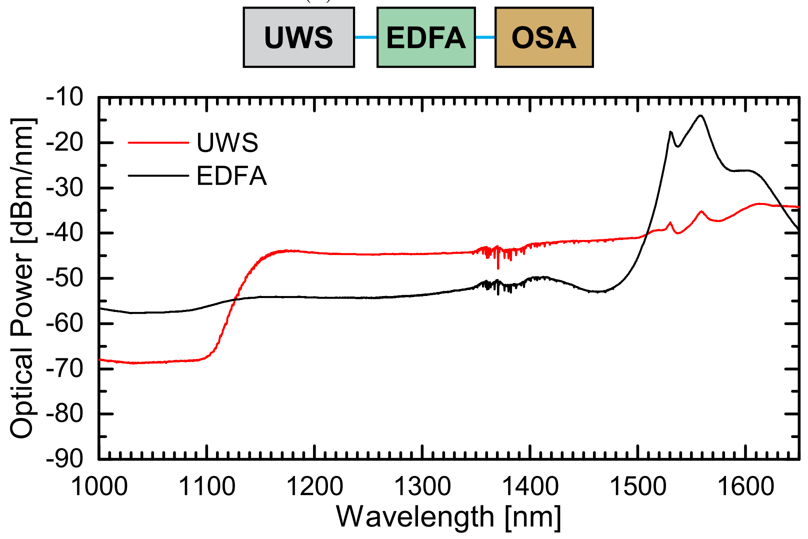


Figure 3.21: Spectral power density of the UWS-1000H.

One can see from Figure 3.22 (a) that the Bandwidth and Wavelength Tunable Filter exhibits an ultra-sharp roll-off which allows only the configured wavelength to pass, however, it is observed that the peak power is lower than the average power of the UWS, as expected. In Figure 3.22 (b), on the other hand, we can see the EDFA's medium gain response with a power peak at -14 dBm, which represents a gain of approximately 20 dB. In both cases, the UWS is used as input.



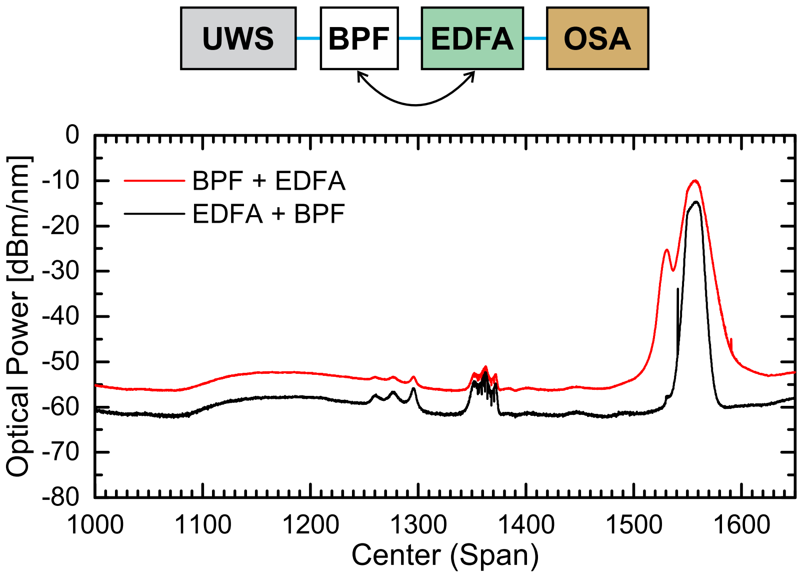
(a): UWS x BWTF.



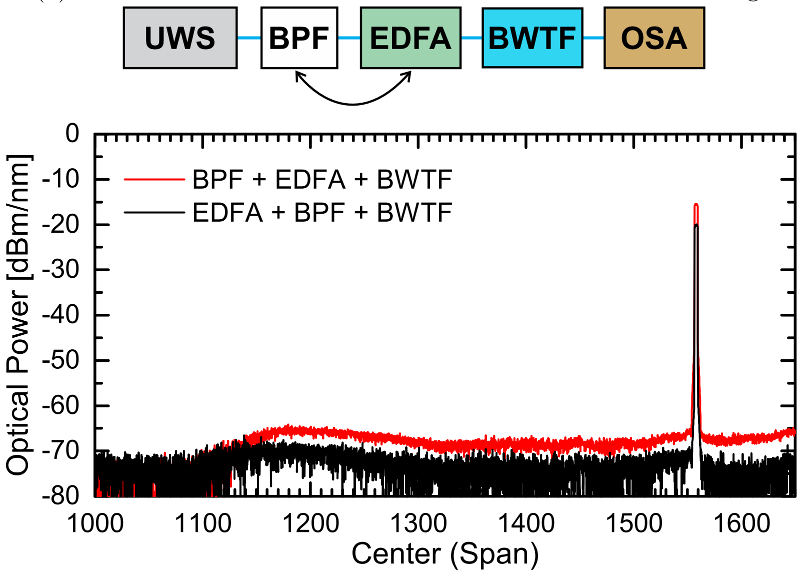
(b): UWS x EDFA.

Figure 3.22: Spectral power density measurements performed using an Optical Spectrum Analyzer (Anritsu MS9710C) [77].

In order to determine in which order the equipments should be connected, two measures were carried out in which the BPF and EDFA had their positions changed. In Figure 3.23 (a) the measurements of the spectral power densities of the BPF and the EDFA are depicted. It's clear that the red line performance is better than the black one. The reason that the spectral power density of the first assembly (red line) is greater than that of the second assembly is that the BPF allows only a certain wavelength range, which is narrower than the EDFA's amplification wavelength range, to be transmitted which is then amplified by the EDFA. In the second assembly, the EDFA will distribute its power over its entire amplification spectrum and will then be filtered by BPF. Also, in Figure 3.23 (b) we can compare the peak power after the BWTF which is approx 6 dB lower than that shown in Figure 3.23 (a). Thereby, we have verified that the BWTF insertion loss is around 6 dB which is in accordance with its specifications, refer to Section 3.3.3.



(a): The order in which EDFA and BPF are mounted is changed.



(b): Spectral power density filtered by the BWTF.

Figure 3.23: Spectral power density measurements of the EDFA and the BWTF were performed using an Optical Spectrum Analyzer (Anritsu MS9710C) [77].

The goal here is to determine the best compromise between dynamic range and spatial resolution for the HR-ν-OTDR. It is clear that, with a higher optical amplification (inclusion of EDFAs), the dynamic range can be increased. However, the EDFA operates inside a limited spectral region and this may induce the broadening of the optical pulse in time since we are considering transform-limited signals. The BWTF may also contribute to the time-broadening of the optical pulse and its impact should also be considered. Another striking factor of high-resolution measurements in long-distance optical fibers is the chromatic dispersion, which can severely limit the spatial resolution. The narrower the spectrum of the optical pulse is, the less impact the chromatic dispersion will have over its spatial resolution in long-distance measurements, which constitutes a compromise relationship: if the BPTF is set to a narrow bandwidth value, the optical pulse may be enlarged in time and render poor spatial resolution in both short- and long-distance measurements, however, if the BPTF is set to a broad bandwidth value, the optical pulse will exhibit a good spatial resolution for short-distance measurements and, due to chromatic dispersion, a poor spatial resolution for long-distance measurements. This compromise relationship is translated in Fig. 3.24, where the achievable spatial resolution versus the bandwidth of the pulse is plotted for different fiber lengths. It is worth noting that, for each distance measurement, there is an optimal bandwidth value that corresponds to the best achievable spatial resolution. Also, detector's jitter may, sometimes, limit the spatial resolution instead of either the chromatic dispersion on the spectral width.

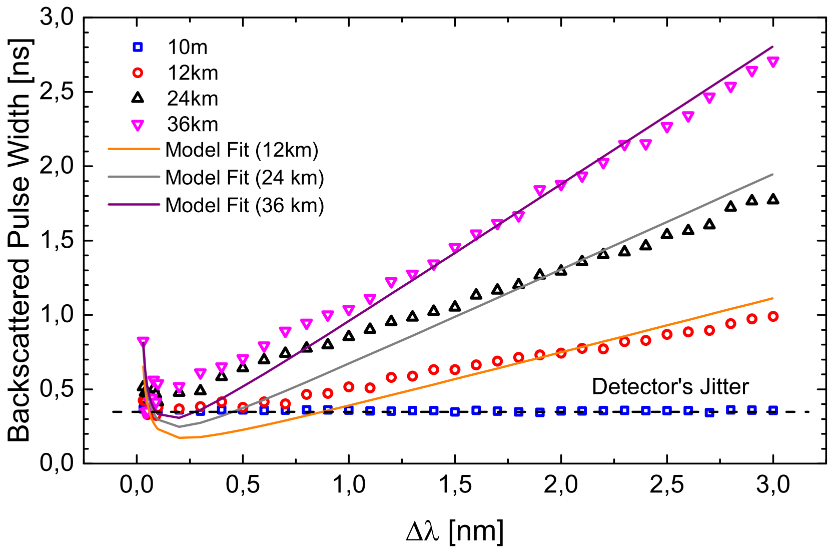


Figure 3.24: The compromise relationship between the spatial resolution and the spectral width for different fiber lengths.

The model used to fit the experimental data was the following:

$$W_p = \Delta\lambda \cdot a + \frac{b}{\Delta\lambda}, \tag{3-2}$$

where a corresponds to $2 \cdot \text{fiber length} \cdot D$ and b corresponds to pulse width of the UWS-1000H at full bandwidth, ie, $b \approx 115\,\text{fs} \cdot 800\,\text{nm}$. Also, we have used an approximation for the dispersion factor D and assumed that it remained constant within $\Delta\lambda$. From the results, it is clear that slope for $\Delta\lambda \geq 1$ nm is well fitted, but the experimental results below this value show some inconsistencies with the fit. We conjecture that this behaviour may arise from the fact that the pulse peak power for reduced $\Delta\lambda$ is very low and the measurement results may be distorted. The back-reflected power also diminishes as the fiber length grows, which is observed in a higher contrast between experimental and fitted data for the longer fibers of 24.6 km and 32.8 km. The pulse width enlargement for low values of $\Delta\lambda$ is due to the transform-limited pulse, i.e., it is as short as its spectral bandwidth permits.

3.6.2 Characterization of a fiber link using the Automatic OTDR

In order to test the performance of Automatic OTDR in searching for faults in a fiber optic link in an automated way, it was necessary to configure the acquisition parameters: fiber length, acquisition time and wavelength. These parameters are sent to the FPGA via a USB interface and then the FPGA starts generating the trigger pulses and the train of gates. The trigger pulses are sent to the DDG which drives both SOAs respecting the optical delay between both to ensure perfect synchronization and the condition of only one light pulse inside the optical fiber. The train of gates are separated by the dead time of the SPAD, each gate creates a detection window of 4 ns. When the acquisition time ends, the data is sent to the computer that then starts the trend filter.

3.6.2.1 Link 1 - 36km (12 + 12 + 12)

A 36 km long fiber link composed of three almost identical fibers with 12 km each was used to generate the OTDR trace and the Filtered Signature shown in Figure 3.25.

The trend filter can take from a few minutes to a few hours depending on the size of the optical fiber. Since the actual spatial resolution of the HR-ν-OTDR is 6 meters, but the acquisition resolution is 1 meter, we may reduce the sampling rate to one sixth of the original one while maintaining the spatial resolution. This way, the processing time is severely reduced from tens of minutes to a couple of minutes. In the measurements shown in Figure 3.25, for example, the trend filter took about two and a half minutes to complete. A list of events containing the location of fiber faults was generated by the filter and the results were as follows: 0, 1011, 12303, 24603 and 36830 meters. The

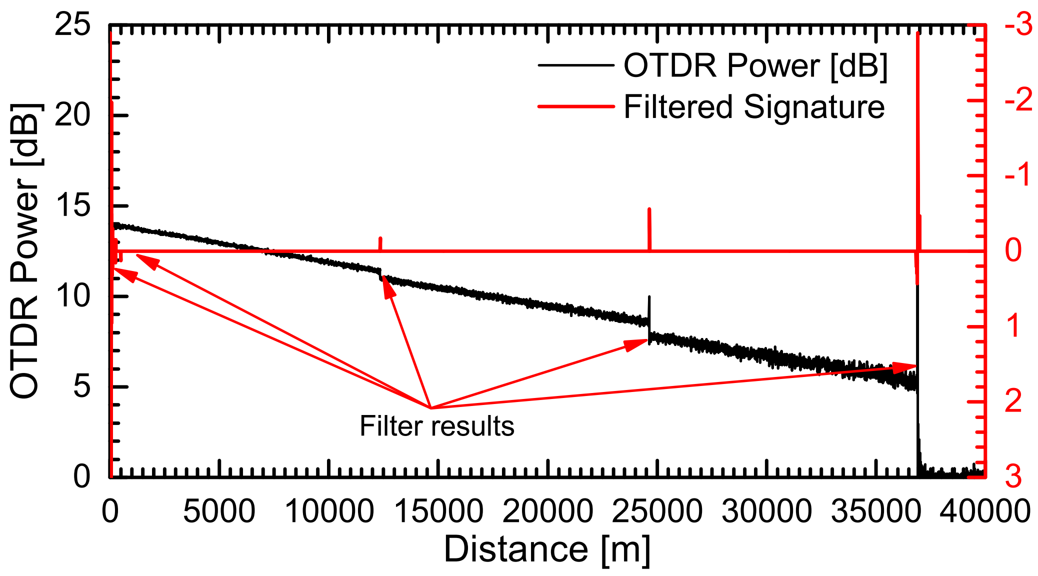
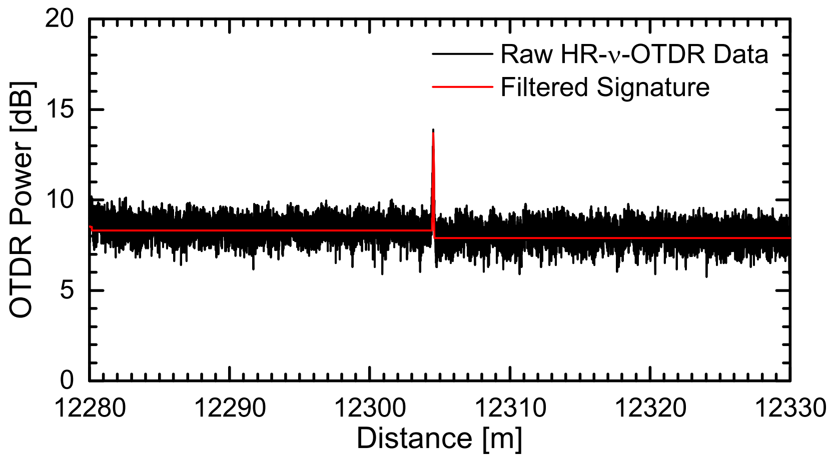


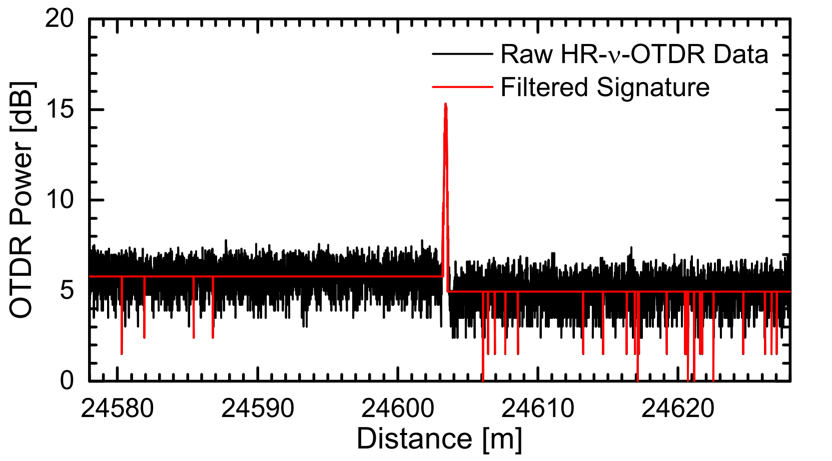
Figure 3.25: High-Dynamic ν-OTDR trace and the filtered signature (Linearized Bregman Adaptive Filter [73]).

beginning of the fiber is the first value (0 meters), the second value was a trend filter's spurious detection, and the third and fourth are the connectors and the last point is the end of the fiber. This list of events is then processed by the High Resolution ν-OTDR that will measure around the points returned by the trend filter to see if it really is a fiber fault. For example, for the second value of the event list (1011 m) the High-Resolution ν-OTDR will measure from 911 to 1111 meters if it is configured to check 200 meters around a fault.

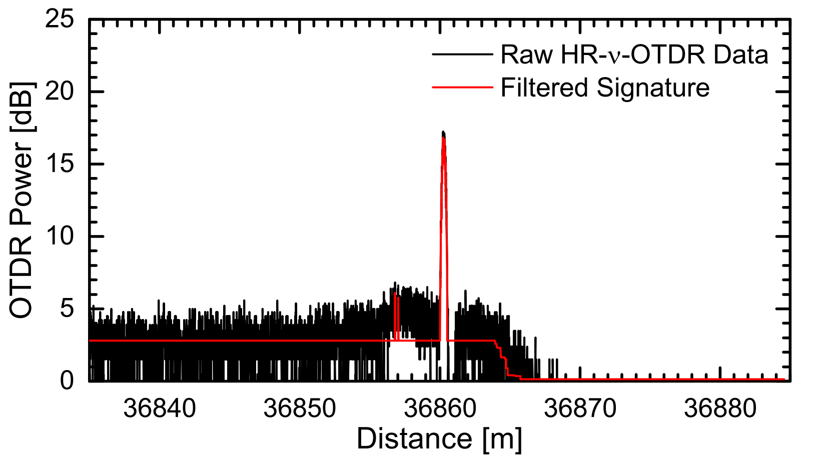
The results of the High-Resolution ν-OTDR are shown in Figure 3.26. Figure 3.26 (a) presents the monitoring result using the HR-ν-OTDR around a fiber fault located at 12304.49 m and a detailed detection of the ℓ_1 Trend Filter. The data in linear scale is fitted by a Gaussian curve and the FWHM criterion is used to determine the achievable spatial resolution of 7.9 cm. Figure 3.26 (b) presents the monitoring result using the HR-ν-OTDR around a fiber fault located at 24603.39 m with an achievable spatial resolution of 13.0 cm and a detailed detection of the ℓ_1 Trend Filter in which there are spurious detections. Finally, Figure 3.26 (c) presents the monitoring result using the HR-ν-OTDR around a fiber fault located at 36860.24 m with an achievable spatial resolution of 20.1 cm and a detailed detection of the ℓ_1 Trend Filter.



(a): Fiber fault located around 12.3 km.



(b): Fiber fault located around 24.6 km.

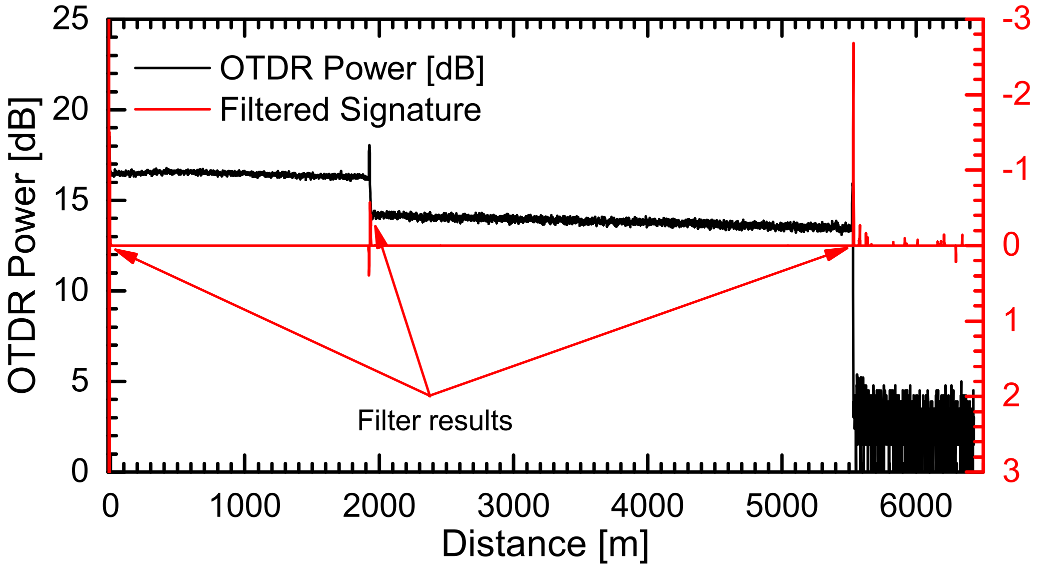


(c): Fiber fault located around 36.8 km.

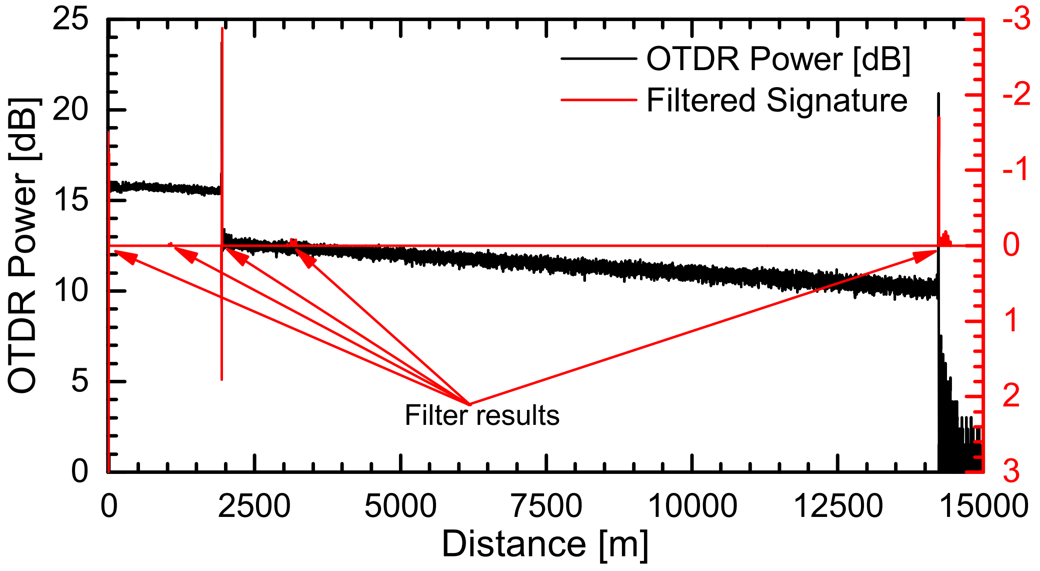
Figure 3.26: HR-ν-OTDR spatial resolution determination via the reflective peak analysis and the filtered signature (ℓ_1 Trend Filter [72]).

3.6.2.2 Link 2 - WDM Link

In order to evaluate the possibility of inspecting a wavelength-multiplexed link, link 2 was assembled containing a ∼2 km feeder fiber and two wavelength-dedicated fibers of ∼3.6 km and ∼12 km connected by a passive wavelength division multiplexer (WDM) at channels 37 and 40 of the DWDM grid. The results of the HD-ν-OTDR are presented in Fig. 3.27.



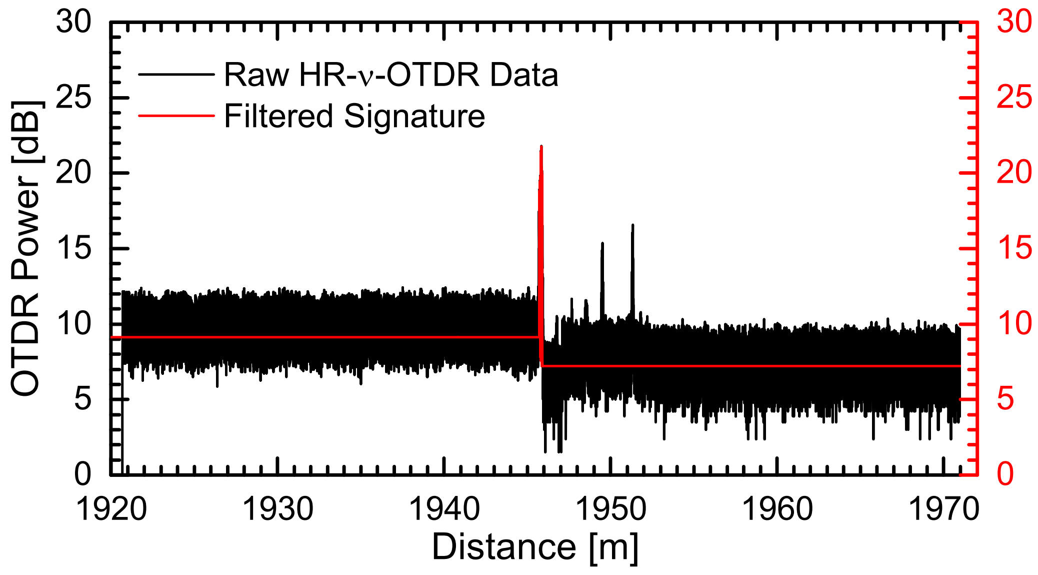
(a): DWDM Channel 37.



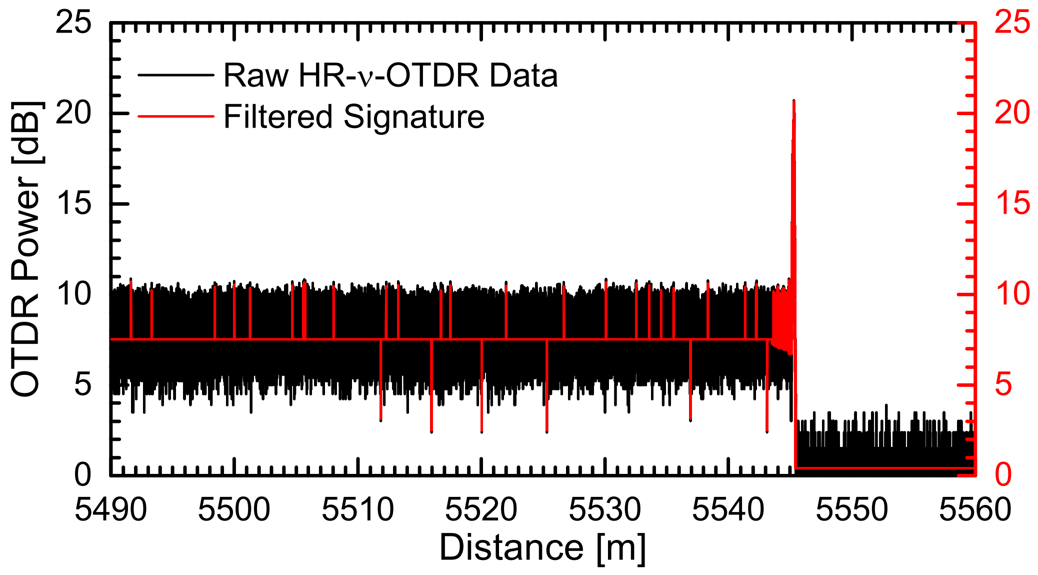
(b): DWDM Channel 40.

Figure 3.27: High-Dynamic ν-OTDR trace and the filtered signature (Linearized Bregman Filter [73]).

The results from the filter, also depicted in Fig. 3.27, clearly indicate the WDM and the fiber end as possible loss candidates. However, as can be seen, there were some spurious reflections, in Fig. 3.27 (b), that were analyzed and proved to be errors of detection of the trend filter. Interestingly, upon inspecting the WDM with the HR-ν-OTDR, several reflections were observed. This behaviour is consistent with the theoretical model of the device, i.e., a prism which separates the WDM channels based on the fact that different wavelengths experience different refractive indexes. The HR-ν-OTDR results are presented in Fig. 3.28 and in Fig. 3.29.

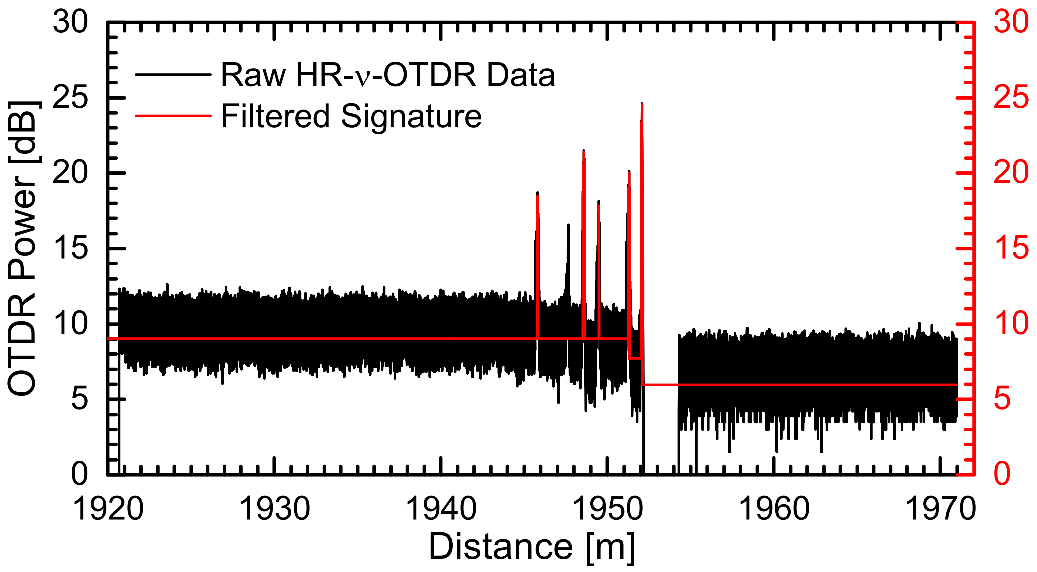


(a): DWDM Channel 37, showing the WDM reflection.

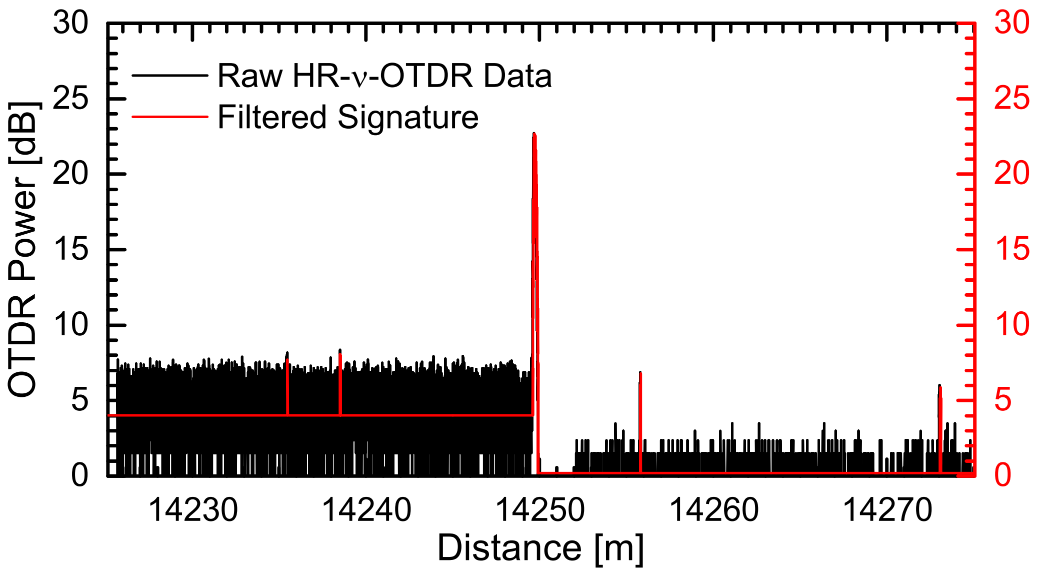


(b): DWDM Channel 37, showing the fiber end.

Figure 3.28: High-Resolution ν-OTDR trace and the filtered signature (ℓ_1 Trend Filter [72]) for DWDM channel 37.



(a): DWDM Channel 40, showing the WDM reflection.



(b): DWDM Channel 40, showing the fiber end.

Figure 3.29: High-Resolution ν-OTDR trace and the filtered signature (ℓ_1 Trend Filter [72]) for DWDM channel 40.

4 Conclusions

At the same time that the Internet is faster, new technologies and media are also emerging. With the growing popularity of video services on demand, gaming, cloud computing, IPTV, among others, fiber optic networking has become a standard in the telecommunications industry because of its virtually unlimited bandwidth. However, there is also a demand for monitoring techniques that can detect failures and test network performance with high dynamic range and high resolution. The present work presented the development of an Automatic ν-OTDR that aims to ally the high dynamic range (HD-ν-OTDR) and the high resolution (HR-ν-OTDR) and the automatic detection of faults in an optical link through the use of trend filters.

We have experimentally verified the technology for automatic detection of faults on an optical fiber. Initially, a High-Dynamic measurement is performed so the OTDR trace passes through a trend filter, and, by returning a list with the fault positions, the High-Resolution ν-OTDR checks around the positions of interest. The results of the High-Resolution ν-OTDR also go through the trend filter to facilitate the analysis of a technician.

We also noted that some points can be improved to allow this technology to be inserted in the context of monitoring fiber optic networks. These points will be left as future work. One of the most important points for the dissemination of this technology is to reduce the cost involved. The pulsed source is the most impacting element in this sense. The pulsed laser source is a Supercontinuum Light whose characteristic is to have a very broad spectrum. However, this does not mean that we can not use a cheaper pulsed source with a finer spectrum because we are limited by the detector's jitter and not by the pulse width.

Another point is the removal of the optical switch since this introduces unnecessary losses, this would be done by introducing a driver for the SOA that could adapt the pulse width for both the HR-ν-OTDR and the HD-ν-OTDR. The SOA Driver is already being developed and an initial release is already in the testing phase. In addition, the LB trend filter has some features that allow its implementation in an FPGA. Among the most important features are the memory usage and the complexity of these filters. Implementing the filter in an FPGA would lower the final cost and speed up processing time. Not least, when we observed the operation of the High-Dynamic ν-OTDR, we noted that it would be possible to improve the acquisition rate of the High-Resolution ν-OTDR by using the technique of firing a train of gates pulses to the detector. Although this can be done, communication between the FPGA and the TDC is not simple. But a methodology to address this problem is already being developed and we expect that the acquisition rate can increase even further and also optimize the process of data acquisition.

Bibliography

[1] A. Gokhale, *Introduction to telecommunications*. Cengage Learning, 2004.

[2] R. Horak, *Webster's New World Telecom Dictionary*. John Wiley & Sons, 2008.

[3] Wikipedia, "History of telecommunication — Wikipedia, the free encyclopedia," May 2017.

[4] W. L. Moran, *The Amarna Letters*. Johns Hopkins University Press, 1992.

[5] J. Kolger. (1986) Mechanical or String Telephones. [Online]. Available: http://strowger-net.telefoniemuseum.nl/tel_tech_mechanical.html

[6] A. Rees *et al.*, *The Cyclopædia*. Longman, Hurst, Rees, Orme and Brown, 1817, vol. 35, public domain.

[7] International Telecommunication Union, "International Morse code," *Recommendation ITU-R*, Oct 2009.

[8] S. Haykin, *Communication Systems*, 5th ed. Wiley Publishing, 2009.

[9] T. E. of Encyclopædia Britannica, "Transducer," in *Encyclopaedia Britannica*. Encyclopædia Britannica, Inc., Set 2013, online at https://www.britannica.com/technology/transducer-electronics.

[10] J. M. Senior and M. Y. Jamro, *Optical fiber communications: principles and practice*, 3rd ed. Pearson Education, 2009.

[11] G. P. Agrawal, *Fiber-Optic Communication Systems*. John Wiley & Sons, Inc, 1997.

[12] Xilinx, Inc., "Xilinx FPGAs," http://www.xilinx.com.

[13] D. J. Smith and A. Foreword By-Zamfirescu, *HDL Chip Design: A practical guide for designing, synthesizing and simulating ASICs and FPGAs using VHDL or Verilog*. Doone Publications, 1998.

[14] A. A. Circuits, "Ttl and gate," http://www.allaboutcircuits.com.

[15] Xilinx, Inc., “Spartan-3 Generation Configuration User Guide,” [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug332.pdf.

[16] J. Andrews, “IEEE Standard Boundary Scan 1149.1 An Introduction,” in *Electro International, 1991*, April 1991, pp. 522–527.

[17] Xilinx, Inc., “Spartan-3 Generation FPGA User Guide,” [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug331.pdf.

[18] acam mess-electronic, “The TDC Cookbook,” Oct 2002.

[19] ——, “ATMD (acam time measuring device) Manual,” [Online]. Available: http://www.acam.de/fileadmin/Download/pdf/English/DB_ATMD_e.pdf, Nov 2003.

[20] ——, “ATMD-GPX: TDC-GPX Evaluation System,” [Online]. Available: http://www.acam.de/fileadmin/Download/pdf/TDC/English/DB_AMGPX_en.pdf, May 2005.

[21] P. Eraerds, M. Legré, J. Zhang, H. Zbinden, and N. Gisin, “Photon Counting OTDR: Advantages and Limitations,” *Journal of Lightwave Technology*, vol. 28, no. 6, pp. 952–964, 2010.

[22] M. Bertolotti, *The history of the laser*. CRC press, 2004.

[23] A. Einstein, “Zur Quantentheorie der Strahlung,” *Physikalische Zeitschrift*, vol. 18, pp. 121–124, 1917.

[24] G. Keiser, *Optical Communications Essentials*. McGraw-Hill, 2003.

[25] L. Thévenaz, *Advanced Fiber Optics: Cconcepts and Technology*. EPFL press, 2011.

[26] J. Hayes, *Fiber optics technician’s manual*, 2nd ed. Delmar Pub, 2005.

[27] E. Hecht, *Optics*, 4th ed. Addison Wesley, 1998.

[28] Anritsu, “Understanding OTDRs,” [Online]. Available: https://dl.cdn-anritsu.com/en-au/test-measurement/files/Technical-Notes/White-Paper/Anritsu_understanding_otdrs.pdf, 11 2011.

[29] R. Ellis, “Explanation of reflection features in optical fiber as sometimes observed in otdr measurement traces,” *Corning White Paper*, 2007.

[30] J. E. Odhner, “Covert ground and port surveillance using hyperbox®: Rayleigh backscattering from fiber optics,” in *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, Oct 2016, pp. 1–5.

[31] M. K. Barnoski, M. D. Rourke, S. M. Jensen, and R. T. Melville, “Optical Time Domain Reflectometer,” *Applied Optics*, vol. 16, no. 9, pp. 2375–2379, Sep 1977. [Online]. Available: http://ao.osa.org/abstract.cfm?URI=ao-16-9-2375

[32] J. P. von der Weid, R. Passy, G. Mussi, and N. Gisin, “On the characterization of optical fiber network components with optical frequency domain reflectometry,” *Journal of Lightwave Technology*, vol. 15, no. 7, pp. 1131–1141, Jul 1997.

[33] D. Derickson, *Fiber Optic - Test and Measurement*. Prentice Hall, 1998.

[34] C. Hentschel, *Fiber optics handbook: an introduction and reference guide to fiber optic technology and measurement techniques*. Hewlett-Packard, 1983.

[35] N. Park, J. Lee, J. Park, J. G. Shim, H. Yoon, J. H. Kim, K. Kim, J.-O. Byun, G. Bolognini, D. Lee *et al.*, “Coded optical time domain reflectometry: principle and applications,” in *Asia-Pacific Optical Communications*. International Society for Optics and Photonics, 2007, pp. 678 129–678 129.

[36] R. Liao, M. Tang, C. Zhao, H. Wu, S. Fu, D. Liu, and P. P. Shum, “Harnessing oversampling in correlation-coded otdr,” *arXiv preprint arXiv:1705.05241*, 2017.

[37] Z. Xie, L. Xia, Y. Wang, C. Yang, C. Cheng, and D. Liu, “Fiber fault detection with high accuracy using chaotic signal from an soa ring reflectometry,” *IEEE Photonics Technology Letters*, vol. 25, no. 8, pp. 709–712, April 2013.

[38] X. Dong, A. Wang, J. Zhang, H. Han, T. Zhao, X. Liu, and Y. Wang, “Combined attenuation and high-resolution fault measurements using chaos-otdr,” *IEEE Photonics Journal*, vol. 7, no. 6, pp. 1–6, 2015.

[39] R. L. Jungerman and D. W. Dolfi, “Frequency domain optical network analysis using integrated optics,” *IEEE journal of quantum electronics*, vol. 27, no. 3, pp. 580–587, 1991.

[40] G. C. Amaral, A. Baldivieso, J. D. Garcia, D. C. Villafani, R. G. Leibel, L. E. Y. Herrera, P. J. Urban, and J. P. von der Weid, “A Low-Frequency Tone Sweep Method for In-Service Fault Location in Subcarrier Multiplexed Optical Fiber

Networks," *Journal of Lightwave Technology*, vol. 35, no. 10, pp. 2017–2025, May 2017.

[41] M. Legré, T. Lunghi, D. Stucki, and H. Zbinden, "Advantages of Silicon Photon Counters in Gated Mode," [Online]. Available: http://marketing.idquantique.com/acton/attachment/11868/f-0066/1/-/-/-/-/Advantages%20of%20Silicon%20Photon%20Counters%20in%20Gated%20Mode.pdf, Nov 2012, iD Quantique.

[42] G. C. Amaral, L. E. Herrera, D. Vitoreti, G. P. Temporão, P. J. Urban, and J. P. der von Weid, "WDM-PON monitoring with tunable photon counting OTDR," *IEEE Photonics Technology Letters*, vol. 26, no. 13, pp. 1279–1282, 2014.

[43] L. Herrera, G. Amaral, and J. P. von der Weid, "Ultra-high-resolution tunable PC-OTDR for PON monitoring in avionics," in *Optical Fiber Communications Conference and Exhibition (OFC), 2015*. IEEE, 2015, pp. 1–3.

[44] G. C. do Amaral, "FPGA Applications on Single Photon Detection Systems," Master's thesis, PUC-Rio, 2014.

[45] L. E. Y. Herrera, "Reflectometria óptica de alta resolução por contagem de fótons," Ph.D. dissertation, PUC-Rio, 2015.

[46] S.-J. Kim, K. Koh, S. Boyd, and D. Gorinevsky, "ℓ_1 Trend Filtering," *SIAM review*, vol. 51, no. 2, pp. 339–360, 2009.

[47] G. C. Amaral, J. D. Garcia, L. E. Herrera, G. P. Temporao, P. J. Urban, and J. P. von der Weid, "Automatic Fault Detection in WDM-PON with Tunable Photon Counting OTDR," *Journal of Lightwave Technology*, vol. 33, no. 24, pp. 5025–5031, 2015.

[48] B. L. Danielson, "Optical time-domain reflectometer specifications and performance testing," *Applied optics*, vol. 24, no. 15, pp. 2313–2322, 1985.

[49] M. Wegmuller, F. Scholder, and N. Gisin, "Photon-counting otdr for local birefringence and fault analysis in the metro environment," *Journal of lightwave technology*, vol. 22, no. 2, pp. 390–400, 2004.

[50] L. Herrera, G. Amaral, and J. von der Weid, "Investigation of bend loss in single mode fibers with ultra-high-resolution photon-counting optical time domain reflectometer," *Applied optics*, vol. 55, no. 5, pp. 1177–1182, 2016.

[51] L. E. Herrera, F. Calliari, J. D. Garcia, G. C. do Amaral, and J. P. von der Weid, "High Resolution Automatic Fault Detection in a Fiber Optic Link via Photon Counting OTDR," in *Optical Fiber Communication Conference*. Optical Society of America, 2016, p. M3F.4. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2016-M3F.4

[52] M. S. Ab-Rahman, N. B. Chuan, M. H. G. Safnal, and K. Jumari, "The overview of fiber fault localization technology in TDM-PON network," in *Electronic Design, 2008. ICED 2008. International Conference on*. IEEE, 2008, pp. 1–8.

[53] M. M. Rad, K. Fouli, H. A. Fathallah, L. A. Rusch, and M. Maier, "Passive optical network monitoring: challenges and requirements," *IEEE Communications Magazine*, vol. 49, no. 2, 2011.

[54] J. Brendel, "High-resolution photon-counting otdr for pon testing and monitoring," in *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference*. Optical Society of America, 2008, p. NThE4. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=NFOEC-2008-NThE4

[55] D. Renker, "Geiger-Mode Avalanche Photodiodes, History, Properties and Problems," *Nuclear Instruments and Methods in Physics Research A*, vol. 567, 2006.

[56] ID Quantique, "id210 - Advanced System for Single Photon Detection," Specifications Sheet, Tech. Rep., 2011.

[57] Avtech Electrosystems Ltd., "AVM Series - Pulse Generators with Ultra-Fast Rise Times," Specifications Sheet, Tech. Rep., 2016.

[58] Yenista Optics, "Tunics Plus - Tunable Laser Source," Specifications Sheet, Tech. Rep., 2009.

[59] Santec Corporation, "Ultra-Wideband Source UWS-1000H," Specifications Sheet, Tech. Rep., 2016.

[60] Alnair Labs, "Ultra-Narrow, Ultra-Sharp Tunable Filter," Specifications Sheet, Tech. Rep., 2016.

[61] Thorlabs, Inc., "Polarization-Independent Optical Shutter/Switch," Specifications Sheet, Tech. Rep., 2013.

[62] Stanford Research Systems, "DG645 Digital Delay Generator: User Manual," Specifications Sheet, Tech. Rep., 2008.

[63] C. Severance, “Guido van rossum: The early years of python,” *Computer*, vol. 48, no. 2, pp. 7–9, 2015.

[64] G. van Rossum *et al.*, “Python 2.7.13 documentation,” Python Software Foundation, Tech. Rep., 2016. [Online]. Available: https://docs.python.org/2/license.html

[65] G. van Rossum, “Python tutorial,” Corporation for National Research Initiatives, Tech. Rep., 2016. [Online]. Available: https://docs.python.org/release/1.4/tut/

[66] M. Telles, *Python power!: the comprehensive guide*. Cengage Learning, 2008.

[67] M. Lutz, *Programming Python*, 4th ed. "O’Reilly Media, Inc.", 2010.

[68] National Instruments, “What is visa?” [Online]. Available: https://www.ni.com/visa/, 2017.

[69] A. Vrancic, “System and method for transferring data over a communication medium using double-buffering,” US Patent US 20 040 044 811 A1, mar 4, 2004. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_20040044811/

[70] T. Bronger, G. Thalhammer, F. Bauer, and H. E. Grecco, “PyVISA: Control your instruments with Python,” [Online]. Available: https://pyvisa.readthedocs.io/en/stable/, 2016.

[71] D. Roberts, “Direct Port I/O and Windows NT,” [Online]. Available: http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/DDJ/1996/9605/9605a/9605a.htm, 1996.

[72] J. P. von der Weid, M. H. Souto, J. D. Garcia, and G. C. Amaral, “Adaptive filter for automatic identification of multiple faults in a noisy otdr profile,” *Journal of Lightwave Technology*, vol. 34, no. 14, pp. 3418–3424, Jul 2016. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-34-14-3418

[73] M. Lunglmayr and M. Huemer, “Efficient linearized bregman iteration for sparse adaptive filters and kaczmarz solvers,” in *Sensor Array and Multichannel Signal Processing Workshop (SAM), 2016 IEEE*. IEEE, 2016, pp. 1–5.

[74] D. V. Caballero, J. P. von der Weid, and P. J. Urban, “Tuneable otdr measurements for wdm-pon monitoring,” in *2013 SBMO/IEEE MTT-S International Microwave Optoelectronics Conference (IMOC)*, Aug 2013, pp. 1–5.

[75] K. Shimizu, T. Horiguchi, and Y. Koyamada, "Characteristics and reduction of coherent fading noise in rayleigh backscattering measurement for optical fibers and components," *Journal of Lightwave Technology*, vol. 10, no. 7, pp. 982–987, Jul 1992.

[76] S. M. Maughan, "Disributed fibre sensing using microwave heterodyne detection of spontaneous brillouin backscatter," Ph.D. dissertation, University of Southampton, 2001.

[77] Anritsu, "MS9710C Optical Spectrum Analyzer Operation Manual," Specifications Sheet, Tech. Rep., 2008.

A
PortIO extension for Python

This C extension for Python exposes the functions to read and write to the specified I/O ports. It's probably overkill to write a C extension for Python to expose these functions, but the performance achieved with this extension overcomes the performance of the PyVISA library by 6-20 times. The functions used are the following: _inp (unsigned short port), _inpw (unsigned short port), _inpd (unsigned short port), _outp (unsigned short port, int databyte), _outpw (unsigned short port, unsigned short dataword), and _outpd (unsigned short port, unsigned long dataword).

About 70% of the code is used to combine the types of variables in C and Python. The first thing to do is import the Python header in order to be able to access the C functions and types in the Python API. Then any communication between the C code and the Python interpreter must be done by passing a PyObject. The function PyArg_ParseTuple converts any Python variable into a C variable, then the C function is executed and the returned values are converted from C variables to Python variables by means of the function Py_BuildValue. Finally, the functions are exposed through the PyMethodDef structure.

portio.c

```c
/*
.context      : portio python extension
.title        : distutils setup
.kind         : python source
.author       : Felipe Calliari <felipe.calliari@opto.cetuc.puc-rio.br>
.site         : Rio de Janeiro - Brazil
.creation     : 30-Dec-2016
.copyright    : (c) 2006-2012 Fabrizio Pollastri   (linux version)
                (c) 2016 Felipe Calliari           (windows version)
.license      : GNU General Public License          (see .copying below)

.copying

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include "Python.h"

#include <conio.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <tchar.h>
#include <windows.h>
#include <windef.h>

/* reference functions
int _inp(
    unsigned short port
);
unsigned short _inpw(
    unsigned short port
);
unsigned long _inpd(
    unsigned short port
);
*/

static PyObject *
pio_inp(PyObject *self,PyObject *args)
{
    unsigned short port;
    int status;
    if (!PyArg_ParseTuple(args,"H",&port)) return NULL;
    status = _inp(port);
    if (status) status = errno;
    return Py_BuildValue("i",status);
}

static PyObject *
pio_inpw(PyObject *self,PyObject *args)
{
    unsigned short port;
    unsigned short status;
    if (!PyArg_ParseTuple(args,"H",&port)) return NULL;
    status = _inpw(port);
    //if (status) status = errno;
    return Py_BuildValue("H",status);
}

static PyObject *
pio_inpd(PyObject *self,PyObject *args)
{
    unsigned short port;
    unsigned long status;
    if (!PyArg_ParseTuple(args,"H",&port)) return NULL;
    status = _inpd(port);
    if (status) status = errno;
    return Py_BuildValue("k",status);
}

/* reference functions
int _outp(
    unsigned short port,
    int databyte
);
unsigned short _outpw(
    unsigned short port,
    unsigned short dataword
);
unsigned long _outpd(
    unsigned short port,
    unsigned long dataword
);
*/

static PyObject *
pio_outp(PyObject *self,PyObject *args)
{
    unsigned short port;
    int databyte;
    int status;
    if (!PyArg_ParseTuple(args,"Hi",&port,&databyte)) return NULL;
    status = _outp(port, databyte);
    if (status) status = errno;
    return Py_BuildValue("i",status);
}

static PyObject *
pio_outpw(PyObject *self,PyObject *args)
{
    unsigned short port;
    unsigned short dataword;
    unsigned short status;
    if (!PyArg_ParseTuple(args,"HH",&port,&dataword)) return NULL;
    status = _outpw(port, dataword);
    if (status) status = errno;
    return Py_BuildValue("H",status);
}

static PyObject *
pio_outpd(PyObject *self,PyObject *args)
{
    unsigned short port;
    unsigned long dataword;
    unsigned long status;
    if (!PyArg_ParseTuple(args,"Hk",&port,&dataword)) return NULL;
    status = _outpd(port, dataword);
    if (status) status = errno;
    return Py_BuildValue("k",status);
}

/* IOperm */
static PyObject *
pio_ioperm(PyObject *self, PyObject *args)
{
```

```c
    HANDLE h;
    h = CreateFile("\\\\.\\giveio", GENERIC_READ, 0, NULL,
                   OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if(h == INVALID_HANDLE_VALUE) {
        printf("Couldn't access giveio device\n");
        return Py_BuildValue("i",-1);
    }
    CloseHandle(h);
        return Py_BuildValue("i",0);
}

/* List of methods defined in the module */

static struct PyMethodDef methods[] = {
    {"inp",(PyCFunction)pio_inp,METH_VARARGS,NULL},
    {"inpw",(PyCFunction)pio_inpw,METH_VARARGS,NULL},
    {"inpd",(PyCFunction)pio_inpd,METH_VARARGS,NULL},

    {"outp",(PyCFunction)pio_outp,METH_VARARGS,NULL},
    {"outpw",(PyCFunction)pio_outpw,METH_VARARGS,NULL},
    {"outpd",(PyCFunction)pio_outpd,METH_VARARGS,NULL},

    {"ioperm",(PyCFunction)pio_ioperm,METH_VARARGS,NULL},

    {NULL, (PyCFunction)NULL, 0, NULL}
};

/* module init function */

static char documentation[] =" PortIO, python low level port I/O for Windows x86\n\
\n\
PortIO is a Python front end to the low level functions provided by the\n\
C library on Windows 386 platforms for the hardware input and output ports:\n\
_inp, _inpw, _inpd, _outp, _outpw and _outpd.\n\
\n\
";

/* module init for both python < 3.0 and >= 3.0 */

static PyObject *Error;

#if PY_MAJOR_VERSION < 3        /* is python < 3.0 */

PyMODINIT_FUNC
initportio(void)
{
    PyObject *m;
    m = Py_InitModule3("portio", methods, documentation);
    if(m == NULL)
        return;

    /* add module specific exception */
    Error = PyErr_NewException("portio.error", NULL, NULL);
    Py_INCREF(Error);
    PyModule_AddObject(m, "error", Error);
}

#else                           /* is python >= 3.0 */

```

```
static struct PyModuleDef portiomodule = {
     PyModuleDef_HEAD_INIT, "portio", documentation, -1, methods
};

PyMODINIT_FUNC
PyInit_portio(void)
{
    HANDLE h;
    h = CreateFile("\\\\.\\giveio", GENERIC_READ, 0, NULL,
                   OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
    if(h == INVALID_HANDLE_VALUE) {
        printf("Couldn't access giveio device\n");
        return -1;
    }
    CloseHandle(h);

    PyObject *m;
    m = PyModule_Create(&portiomodule);
    if(m == NULL)
        return NULL;

    /* add module specific exception */
    Error = PyErr_NewException("portio.error", NULL, NULL);
    Py_INCREF(Error);
    PyModule_AddObject(m, "error", Error);

    return m;
}

#endif

/* end */
```

The last thing to do is to compile and link this module so it can be called from Python. There are two ways to accomplish this step, the first, and more complicated, is to download the source code of Python and compile everything by command line. The second way to do this is to use the built-in Python distribution utilities, to do this one only need to create a setup.py file and run “python setup.py install”.

setup.py

```
#!/usr/bin/python
#
# .context    : portio for windows python extension
# .title      : distutils setup
# .kind       : python source
# .author     : Felipe Calliari <felipe.calliari@opto.cetuc.puc-rio.br>
# .site       : Rio de Janeiro - Brazil
# .creation   : 30-Dec-2016
# .copyright  : (c) 2006-2012 Fabrizio Pollastri  (linux version)
#               (c) 2016-2017 Felipe Calliari     (windows version)
#                   <felipe.calliari@opto.cetuc.puc-rio.br>
# .license    : GNU General Public License (see .copying below)
#
# .copying
#
```

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
#

from distutils.core import setup, Extension
import os
import os.path
import re
import string
import sys

classifiers = """\
Development Status :: 4 - Beta
Intended Audience :: Developers
License :: OSI Approved :: GNU General Public License (GPL)
Programming Language :: Python
Programming Language :: Python :: 2
Topic :: System :: Hardware
Topic :: Software Development :: Libraries :: Python Modules
Operating System :: Windows
"""

# check for proper python version
if sys.version < '2.7':
    print('\nportio requires python >= 2.7, found python', \
          sys.version.split()[0], ', exiting...')
    sys.exit()

module = Extension(
    'portio',
    define_macros = [('MAJOR_VERSION', '0'),('MINOR_VERSION', '2')],
    sources = ['portio.c'])

setup (
    name = 'portio',
    version = '0.2',
    author = 'Felipe Calliari',
    author_email = 'felipe.calliari@opto.cetuc.puc-rio.br',
    maintainer = 'Felipe Calliari',
    maintainer_email = 'felipe.calliari@opto.cetuc.puc-rio.br',
    url = '',
    license = 'http://www.gnu.org/licenses/gpl.txt',
    platforms = ['Windows'],
    description = 'PortIO, python low level port I/O for Windows x86\n',
    ext_modules = [module])

# cleanup
try:
    os.remove('MANIFEST')
except:
    pass

#### END
```