DESIGN AND TEST OF AN EVENT DETECTOR AND LOCATOR FOR THE REFLECTOACTIVE™ SEALS SYSTEM

Brad J. Stinson

Oak Ridge National Laboratory

Bldg. 3606 MS 6003, P.O. Box 2008

Oak Ridge, TN 37831 865/241-6757 FAX 865/574-4529

ABSTRACT

The purpose of this work was to research, design, develop and test a novel instrument for detecting fiber optic loop continuity and spatially locating fiber optic breaches. The work is for an active seal system called ReflectoActive™ Seals whose purpose is to provide real time container tamper indication.

A Field Programmable Gate Array was used to implement a loop continuity detector and a spatial breach locator based on a high acquisition speed single photon counting optical time domain reflectometer. Communication and other control features were added in order to create a usable instrument that met defined requirements. A host graphical user interface was developed to illustrate system use and performance.

The resulting device meets performance specifications by exhibiting a dynamic range of 27dB and a spatial resolution of 1.5 ft. The communication scheme used expands installation options and allows the device to communicate to a central host via existing Local Area Networks and/or the Internet.

Table of Contents

1.0 Introduction 1

2.0 Background 2
2.1 Introduction 2
2.2 Previous Work 3
2.2.1 OTDR Only Approach 4
2.2.2 Analog Immediate Detection Unit and Tektronix OTDR 5
2.2.3 Digital Immediate Detection Unit with Tektronix OTDR 7
2.3 Optical Time Domain Reflectometer Review 8
2.3.1 Design of a basic OTDR 8
2.3.1 OTDR Dead Zones 9
2.3.2 Analog OTDR Resolution Limits 10
2.3.3 OTDR Ghosting 10
2.3.4 Photon Counting OTDR 11

3.0 System Design Concepts 16
3.1 Introduction 16
3.2 Simple Deployment 16
3.3 Common Deployment 17
3.4 Event Detector and Locator Theory of Operation 20
3.4.1 EDL in the Event Detection Mode 21
3.4.2 EDL in the Event Location Mode 22

4.0 Hardware Design 24
4.1 Introduction 24
4.2 Power Circuitry 25
4.3 FPGA and Support Circuitry 25
4.4 Low Cost Picosecond Laser Driver 27
4.5 Control and Communication Module 32
4.6 Alarm Circuitry 35
4.7 Optical Components 37
4.7.1 Laser Diode 37
4.7.2 1x2 Coupler 37
4.7.3 Optical Switches 38
4.7.4 Dicon Fiber Optic Variable Attenuator 38
4.8 Perkin-Elmer Single Photon Avalanche Photo Detector 39
4.9 Display 40
4.10 Packaging 41

5.0 Software Design 42
5.1 Introduction 42
5.2 Mode Control Logic in VHDL 42
5.3 Loop Continuity Detection in VHDL 44
5.3.1 Encoded Word Generation 44
5.3.2 Encoded Word Transmission, Reception and Verification 46

5.4 Photon Counting OTDR in VHDL and Block Diagram 51
5.4.1 ν-OTDR Signal Generation 52
5.4.2 ν-OTDR Control 52
5.4.2 ν-OTDR High Speed Sampling Using Phase Offset PLL 57
5.4.3 ν-OTDR Waveform Transfer 59
5.5 Communication and System Control in Dynamic C 63
5.5.2 Network Communication 64
5.5.1 Communication with FPGA 70
5.5.3 Optical Switch Control 72
5.5.4 Attenuator Control and Calibration 72
5.5.5 Alarm Interface 73
5.5.6 Tamper Input 73
5.5.7 Display Control 74
5.6 Windows Graphical User Interface in Microsoft C# 74
5.6.1 EDL Status Display and Control 74
5.6.2 Event Logging 77
5.6.3 Item Addition and Removal 77
5.6.4 Illustration of Maintaining User Accountability 77
5.6.5 ν-OTDR Waveform Display and Characterization 78
5.6.6 Breach Location Algorithm 80

6.0 Design Results and Performance 83
6.1 Introduction 83
6.2 Event Detector Performance 83
6.3 Event Locator (ν-OTDR) Performance 84

7.0 Conclusions 88
7.1 Contributions 88
7.2 Future Work and Research 89

REFERENCES 92

List of Figures

Figure 1: Successive OTDR Scan Comparison 5

Figure 2: Analog IDU with Tektronix OTDR 6

Figure 3: First OTDR by Barnoski and Jensen 9

Figure 4: Histogram Based on Table 1 13

Figure 5: Single Loop RAS Deployment 17

Figure 6: Typical RAS Deployment 18

Figure 7: Typical RAS Facility Implementation 19

Figure 8: EDL Block Diagram in Event Detection Mode 22

Figure 9: EDL Block Diagram in Event Location Mode 23

Figure 10: EDL Circuitry 24

Figure 11: ECL Based Core of Picosecond Diode Driver 29

Figure 12: Input Stage of Picosecond Diode Driver 31

Figure 13: Picosecond Driver Delayed Pulse Response 31

Figure 14: Picosecond Driver Optical Output Pulse 33

Figure 15: Rabbit RCM3360 Core Module 33

Figure 16: Alarm Interface Circuitry 36

Figure 17: EDL Light Emitting Diode Circuit 36

Figure 18: 2x2 Optical Switch in Reverse Mode 38

Figure 19: SPCM-AQR Single Photon Counter Optical Efficiency 40

Figure 20: Event Detector and Locator Enclosure 41

Figure 21: FPGA Software Flowchart 43

Figure 22: VHDL Mode Control Process 45

Figure 23: Event Detection Signal Transmission VHDL 47

Figure 24: Event Detection Signal Reception VHDL..... 49

Figure 25: ν-OTDR Control State Machine in VHDL..... 53

Figure 26: High Speed Sampling in VHDL with Phase Offset PLL 58

Figure 27: Altera Block Diagram of High Bit Detector 60

Figure 28: Phase Delay Based High Speed Sampler 61

Figure 29: RCM3360 Firmware Flowchart 65

Figure 30: EDL Main Page in HTML..... 67

Figure 31: EDL Administrator Tools HTML Page..... 68

Figure 32: GetStatus.cgi Returned XML Document 69

Figure 33: RAS GUI Main Screen..... 76

Figure 34: RAS GUI User Authentication..... 78

Figure 35: RAS GUI ν-OTDR Screen 79

Figure 36: ν-OTDR Waveform Illustrating Breach Location..... 82

Figure 37: ν-OTDR Spatial Resolution Limit Test..... 85

List of Abbreviations

APD	Avalanche Photodiode Detector
CGI	Common Gateway Interface
DIDU	Digital Immediate Detection Unit
DPRAM	Dual Port Random Access Memory
ECL	Emitter Coupled Logic
EDL	Event Detector and Locator
FPGA	Field Programmable Gate Array
FWHM	Full Width Half Maximum
GUI	Graphical User Interface
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IDU	Immediate Detection Unit
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LFSR	Linear Feedback Shift Register
OTDR	Optical Time Domain Reflectometer
OTS	Off The Shelf
PECL	Positive Emitter Coupled Logic
PLL	Phase-Locked Loop
RAM	Random Access Memory
RAS	ReflectoActive™ Seals
RF	Radio Frequency
SNR	Signal to Noise Ratio
SPAD	Single Photon Avalanche photodiode Detector
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
TCP/IP	Telecommunications Protocol / Internet Protocol
TID	Tamper Indicating Devices
TTL	Transistor-Transistor Logic

VHDL	Very High Speed Integrate Circuit Hardware Description Language
XML	eXtensible Markup Language
ν-OTDR	Single Photon Counting Optical Time Domain Reflectometer

1.0 Introduction

In the mid 1990s a request was made for a material accountability system that could provide continuous near real-time container tamper indication. To meet the requirements, a system must use "active" seals affixed to containers in a manner such that opening or tampering with the lid of the container causes the seal to be broken. The requested systems replace or supplement passive tamper indicating devices (TIDs) that require manual visual inspection to determine their status. The systems must employ reusable seals, be low cost, low maintenance, easy to use, safe for sensitive processing areas and provide near real-time indication of which seal is broken or being tampered with.

The benefits of such systems include a dramatic improvement in the safeguarding and accountability of valuable assets. Because assets are monitored continuously, a tamper attempt is known immediately. This is in contrast to passive TIDs, where a tamper attempt remains unknown until the next physical inspection. In some cases, government agencies will approve an extension of the period between mandated physical inventories based on the use of continuous monitoring technology. Fewer physical inventories translate to reduced cost and in some cases reduced worker exposure to hazardous materials.

2.0 Background

2.1 Introduction

Past active seal designs provide a very important foundation for this work. As such, it is worthwhile to consider the requirements, challenges and constraints faced during the early designs. The next few paragraphs will explain in more detail the requirements and constraints associated with an active seal system and why a fiber optic based approach was originally chosen.

The first and perhaps most important requirement is that the system be secure and reliable. From the beginning, a wireless system was out of the question due to security regulations, so some form of wired system is required. An electrical system is vulnerable to splicing and patching around the seal. An optical system is much more difficult to splice into (requires specialized tools and specialized skills). Even if a fiber optic cable could be spliced without a disruption of signal, the loss of light caused by a splice is detected easily by an OTDR. In contrast to considered electrical solutions, light propagating through a fiber can travel several kilometers without needing regeneration due to signal degradation. Finally, fiber optic components are widely used in the telecom industry and have been proven reliable.

Second only to security is cost. In order to reduce cost and design time, the system needs to use off-the-shelf components where possible. The fiber optic industry is quite mature and components such as fiber, connectors, transmitters, receivers, switches and multiplexers are becoming more available and more affordable every day.

The next requirement is that the system be adaptable to a wide range of facilities and storage containers. The design needs to be able to form a seal on drum rings as well as other proprietary storage containers. The design needs to be capable of monitoring anywhere from a few dozen items up to thousands of items. Fiber optic connectors are available in a wide variety and are easily adapted to form seals on different types of enclosures or containers. The ST connector is employed because it is easy to use but unlikely to be accidentally unplugged. The ST connector is a keyed-bayonet style connector suitable for multimode or singlemode connections. Multimode fiber was used because of an increased alignment tolerance (due to a larger fiber core) and because multimode connections generate larger Fresnel reflections.

The final requirement is that the system be robust enough to withstand a shop floor environment. The fiber optic cable is Kevlar reinforced and has proven to be quite durable. When the system is closed (secure) it is insensitive to dust and other contaminants. In addition, fiber optics are insensitive to temperature variance, work well in radio frequency (RF) noisy areas and are safe for OSHA class I, II and III facilities [1].

2.2 Previous Work

The initial design of a fiber optic seal system began in the mid 1990s. The first design (as well as all future designs) was given the name ReflectoActive™ Seals (RAS). The RAS system has been through several iterations over the last 10 years. Each version has provided a significant improvement over the last. The next few sections will describe the previous RAS systems.

2.2.1 OTDR Only Approach

The initial approach was to use an off-the-shelf Tektronix TFP2A FiberMaster® OTDR controlled by a computer via the General Purpose Interface Bus (GPIB or IEEE-488). The TPF2A is an analog OTDR and works by injecting a short pulse of light into a fiber. Fresnel reflections occur at fiber discontinuities causing minute amounts of light to be reflected back to the transmitter. Given that the phase velocity of light in the fiber core is constant, the time of flight and the received amplitude is used to generate a waveform of reflected signal strength versus distance [3]. More details on OTDR operation will be given in Section 2.3.

In the original setup, the OTDR would repeatedly scan a loop of fiber optic cable, comparing successive scans for changes. A sudden signal drop would be perceived as a breach. The waveform would then be downloaded to a computer and compared to a predefined "secure" waveform to determine which seal was breached [2]. A fiber optic switch was present that allowed the system to scan in the both the forward and reverse directions.

The waveform in Figure 1 illustrates an ideal case of two successive scans. The first scan (in blue) is the secure waveform. The second scan (in red) is taken some time later after connector 3 is opened. The reflection from connector 4 is no longer present because it is isolated from the system. This is an indication of a breach at seal 3.

The main problem with this approach was speed. The acquisition time for a high resolution waveform took several minutes, leaving the fiber optic seals vulnerable to undetected tampering. The system also suffered from repeatability problems. The successive OTDR scans varied in shape and peak amplitude making the design of a

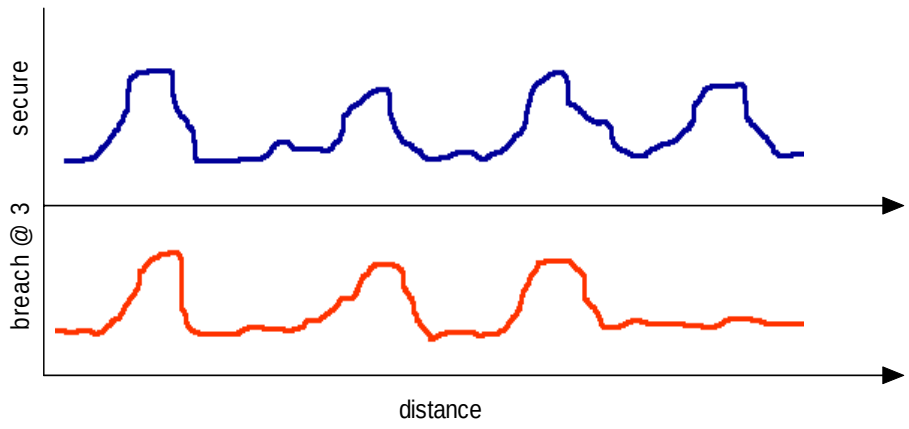

Figure 1: Successive OTDR Scan Comparison

waveform comparison algorithm more difficult. Generally, clever signal processing solved the last problem but the solution was far from ideal.

2.2.2 Analog Immediate Detection Unit and Tektronix OTDR

To overcome the speed issue discussed in Section 2.2.1, a new piece of hardware called the Immediate Detection Unit (IDU) was introduced that could quickly detect a fiber breach. This hardware removed the breach detection burden from the OTDR. The IDU changed the system design slightly by requiring that the seals be arranged in a loop. Light was injected into one end of the loop and verified at the opposite end. The IDU used a light emitting diode (LED) based transmitter and matched photo detector. The output of the photo detector was amplified by a potentiometer controlled analog amplifier. A binary (secure/breached) output was provided.

In the event of a seal breach detected by the IDU, the Tektronix TFP2A OTDR was connected to the loop (looking either forward or reverse down the loop) via a computer controlled fiber optic switch. The OTDR and IDU used two different

wavelengths, so waveform division multiplexers were used to combine the two signals onto a single fiber. The OTDR would then acquire a waveform and download it to the computer. By comparing previously acquired secure waveforms, the breach location was identified. Figure 2 is a simplified drawing of this system.

The problems with this version were limits in the seal capacity due to the signal to noise ratio (SNR), repeatability problems with the OTDR like those previously discussed and a design which made large scale deployments difficult. The analog circuitry in the IDU was temperature dependent and required setting a potentiometer for correct levels. Due to the low output power of the LED, the maximum seal count was limited to about 50 seals per loop. To monitor hundreds of items required many of these expensive setups. Furthermore, they were all isolated instruments. It was desired that all seals in a facility be monitored by a single computer.

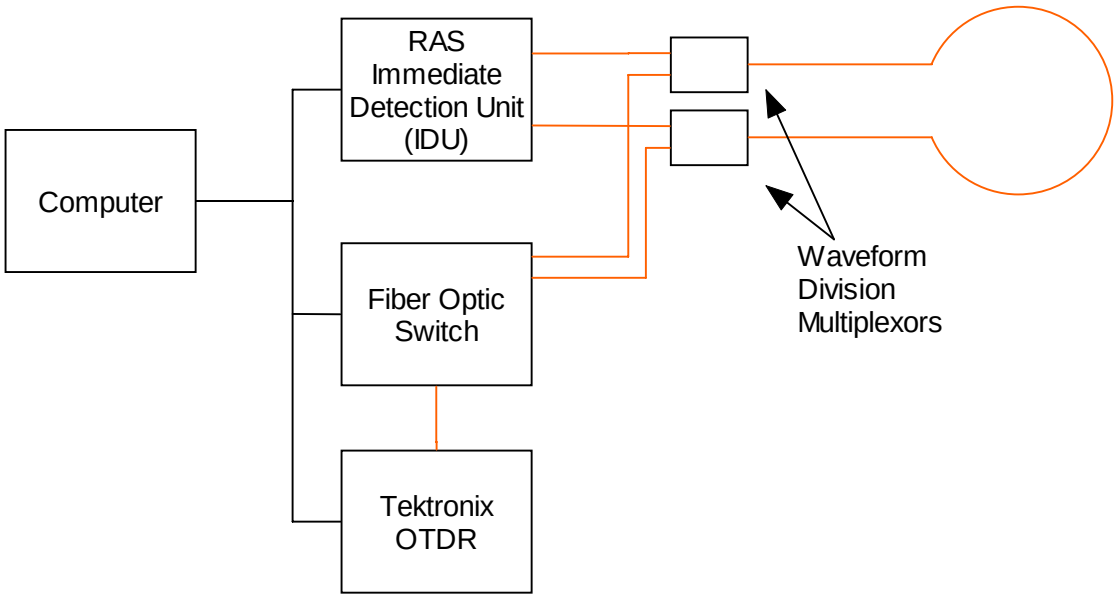

Figure 2: Analog IDU with Tektronix OTDR

2.2.3 Digital Immediate Detection Unit with Tektronix OTDR

The desire for support of large seal counts throughout a facility led to the design of the Digital Immediate Detection Unit (DIDU). The basic IDU design was leveraged and improved by using a laser diode and more sensitive receiver electronics. An added layer of security was providing by using a Field Programmable Gate Array (FPGA) to generate and verify encoded signals. Finally, an 8-bit microprocessor was added to handle communications using the Telecommunication Protocol/Internet Protocol (TCP/IP) over a standard Ethernet network.

The addition of a laser diode led to an increase in the SNR and allowed up to 150 seals to be used per loop. More importantly, since each DIDU could now be placed on a network, one computer could be used to log data and control multiple DIDUs throughout an entire facility. The DIDU also represented the first effort to miniaturize the system by using surface-mount electronics.

The main problem with the DIDU is associated with the Tektronix OTDR. By the time the DIDU was designed, Tektronix had discontinued production of the TFP2A in favor of smaller, handheld field instruments. Better OTDR performance than the TFP2A could provide had been desired for some time, and with the TFP2A no longer available there was no choice but to look for an OTDR alternative.

The DIDU also suffered from scalability issues. Each DIDU supported twenty fiber optic loops and contained an onboard switch that injected the OTDR signal into any one of them. For small deployments, the excess capacity served no functional purpose, but greatly increased system cost. The “sharing” of the OTDR between twenty loops led

to an increase in the time required to locate a breach, especially in the event of multiple breaches on different loops.

2.3 Optical Time Domain Reflectometer Review

The RAS system dictates somewhat unique requirements for an OTDR. The most common use of commercial OTDRs is in the telecom industry when a technician needs to locate a cut line or identify a bad splice or connector. In all of these cases, the number of connectors, splices or breaks are low and the length of fiber is high. In the RAS system, there are hundreds of connectors with only a few meters between them. Phenomena such as event dead zones and ghosting make detecting a large number of connectors separated by only a few meters a very difficult task for most commercial OTDRs. These phenomena will be discussed in the next few sections. A new approached called single photon counting optical time domain reflectometry will be introduced that solves the dead zone problem.

2.3.1 Design of a basic OTDR

The first OTDR was conceived by Barnoski and Jensen almost 30 years ago. They were studying steady-state attenuation of fiber optic cable and realized that they could determine the attenuation coefficient by analyzing backscattered light (Rayleigh scattering). This was a great advancement because previous measurement techniques required cutting the fiber and measuring the optical power transmitted through two lengths. Barnoski and Jensen's approach led the way to testing nondestructively via one end face [3]. The setup used in their experiment is shown in Figure 3. The result most important to this work is the large reflection Barnoski and Jensen observed as a result of

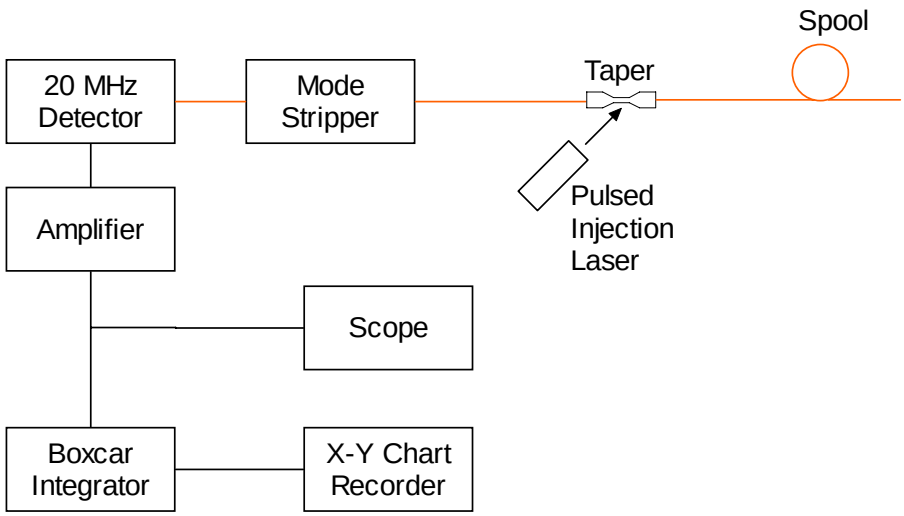

Figure 3: First OTDR by Barnoski and Jensen

the cleaved end of the 360 meter spool (a reflection resulting from a feature such as a cleaved end or connector is called a Fresnel reflection). They noted that the time scale could be converted to propagation distance given a known phase velocity of light in the core. The cut end of the fiber and other discontinuities that create Fresnel reflections could therefore be located along the length of fiber optic cable.

2.3.1 OTDR Dead Zones

OTDRs inject a short pulse of light into one end of a fiber and then use a highly sensitive avalanche photodiode receiver to measure the minute amount of light (typically -65 to -70 dB/m) reflected back [4]. The sensitive receivers are easily saturated by strong reflective events such as a connector or splice. The avalanche process causes a macroscopic current to flow, which lowers bias voltage. Since the photodiode has a finite capacity, some amount of time is required to recover [5]. This recovery time is known as an event dead-zone. During the dead-zone period the fiber status is unknown. The

Tektronix TFP2A has an event dead zone of 3 meters at a wavelength of 1300nm in multimode fiber. In terms of the RAS system, this dead-zone translates to a minimum required spacing between active seals. An ideal OTDR has no event dead zone.

2.3.2 Analog OTDR Resolution Limits

The spatial resolution for a standard, non-gated, analog detection OTDR is limited to ~1m. This limitation is due to pulse duration and a tradeoff between fast detector response (bandwidth) and detector noise. The detector noise is proportional to the square-root of the detector bandwidth. The detector sensitivity, $P_{\min}$, can be expressed as

$$P_{\min} \approx NEP_{rc}\sqrt{bw/N}, \qquad \text{Equation 1}$$

where NEP_{rc} is the noise-equivalent power of the receiver, *bw* is the bandwidth and N is the number of measurements [6].

2.3.3 OTDR Ghosting

An effect called ghosting is the result of reflections that bounce between Fresnel events one or more times before reaching the receiver. The net effect is that an apparent reflection appears at some distance multiple of real events [7]. In typical RAS deployments, it is common to use the same length of fiber between container seals. When connectors occur at regular intervals, the ghost signals can overlap with real connector signals. The result is a false indication of the true reflection strength of each victim connector. Compounded ghosting becomes an even greater problem when a seal is opened, because ghosts can make a connection appear as if it is still there when in fact it has been removed from the loop. Ghosts are minimized in the RAS system by

attenuating the signal, which turns out to be a requirement for the photon counting OTDR anyway (see the next section). In addition, careful signal analysis is used to help ignore the effects.

2.3.4 Photon Counting OTDR

The invention of the ν-OTDR is a logical progression brought about by a desire to increase dynamic range. By employing very sensitive avalanche photodiodes the receiver is able to detect a single photon. Detector availability at room temperatures is no longer a challenge; InGaAs/InP avalanche photodiode detectors cooled by Peltier coolers are readily available [8]. The challenge with designing a ν-OTDR is due to the binary nature of avalanche photodiode detectors (APD). Single photon detectors operate in the Geiger mode, meaning that a single photon acts as a trigger that uses up all the stored energy in the device creating a macroscopic current that is easily detectable [5]. Therefore, the detector reacts the same whether it receives one photon or multiple photons. Since the goal of an OTDR is to provide intensity information about reflected light, a probabilistic, multi-measurement approach must be taken.

To implement the probabilistic approach successfully, light attenuation must be controlled so that during each gating period the average number of received photons is less than one [5]. Lacaita, Francese and Cova demonstrated that the SNR can be maximized by adjusting the power so that the probability of detecting a photon is 63% for each output pulse [9]. To understand the need for attenuation, consider two Fresnel reflections, A and B. Reflection A is very strong and is immediately followed by the very weak reflection B. The attenuation must be set so the stronger reflection A does not always overshadow the weaker reflection B. Once a photon is detected and the avalanche

begins, there is a finite recharging time during which the detector is not sensitive. For the sake of this explanation assume that the time of flight for light in the fiber from reflection A to B is less that the APD recharge time. If we are to ever see reflection B, then the attenuation must be set such that on average a reflection does not occur from either reflection A *or* B. This will ensure that there is some probability that a reflection originates from point B. Over multiple scans, reflection probability will be "built-up" into a histogram. Table 1 and Figure 4 provide an example of how the probabilistic data is received. In this example, a photon is received for only 20% of the output pulses. Since event A at 20 meters is more intense than event B at 21 meters, 5 events are recorded from event A, while only one occurs from event B. This makes sense logically; since reflection A is stronger than B, there is a higher probability that received photons will have been generated by event A.

It should be obvious by now that in order to determine optical intensity accurately, many (thousands or tens of thousands) successive scans are required. The measured probability is proportional to incident signal power provided the detector is not saturated and the power is greater than the thermal detector noise [10 and 4]. This is both a blessing and a curse for ν-OTDRs. It is a blessing in the sense that since each pulse leads to at most one detection event, there is no dead zone in the classical sense (this is ignoring multi-hit mode which will be discussed later). The lack of a classical dead zone is the primary reason for choosing to implement a ν-OTDR in the RAS system. It is a curse in the sense that detection time increases because multiple scans are required. However, a multi-hit mode can be used (and is used in RAS) to decrease detection time dramatically.

Table 1: Sample ν-OTDR data for 20 scans

Scan Number	Event Detection
Scan 1	none
Scan 2	none
Scan 3	none
Scan 4	20
Scan 5	none
Scan 6	none
Scan 7	none
Scan 8	20
Scan 9	none
Scan 10	none
Scan 11	21
Scan 12	20
Scan 13	none
Scan 14	none
Scan 15	none
Scan 16	20
Scan 17	none
Scan 18	none
Scan 19	none
Scan 20	20

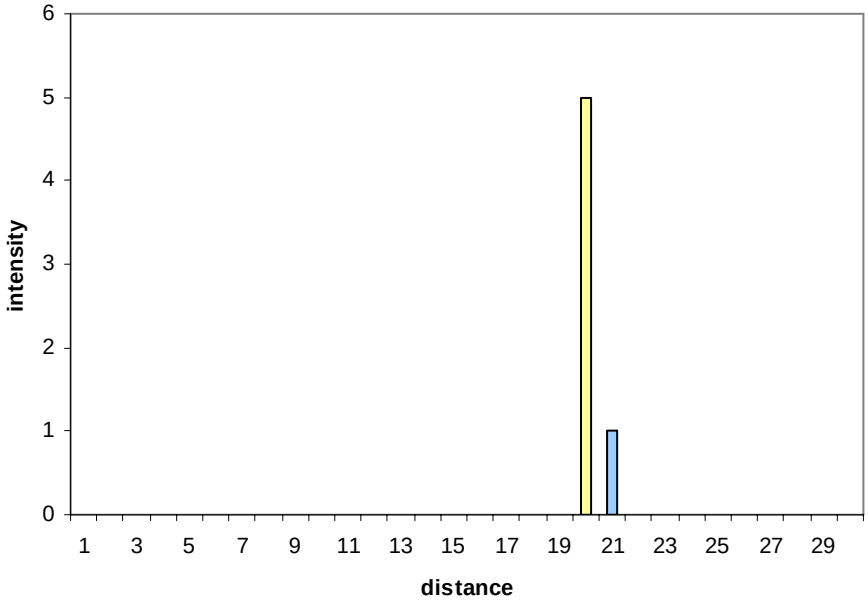

Figure 4: Histogram Based on Table 1

The increased dynamic range of a ν-OTDR over a standard analog detection OTDR is predicted at ~7dB [6]. This leads to 2-point spatial resolution improvement of around a factor of 100 based on Equation 1. Increased spatial resolution is an obvious benefit for any system designed to locate tightly spaced fiber faults. However, as spatial resolution is increased, minimization of the pulse duration becomes critical if one is to maintain high resolution. Since a reflected photon could have come from any one of the millions of photons comprising a pulse of light, spatial resolution is limited to the pulse width. The proceeding statement assumes that the light pulse is an ideal step function. In reality, the pulse is quasi-Gaussian or may contain multiple peaks within the width. The probability of reflection is highest at the pulse peak and falls off toward the leading and trailing edges. In a real sense, the spatial resolution is less than or equal to the pulse width, which means that the design of a low cost, very short pulse (hundreds of picoseconds) laser diode driver is a vital aspect of the system. The design of such a driver is detailed in Section 4.4.

To improve the SNR further, ν-OTDR implementations often gate the APD so that it is only active over a specific region of interest [11]. The starting point of the region of interest is delayed with each scan so that over successive scans the entire fiber length is analyzed. Given a constant total scan time, SNR is increased as the gating region is decreased.

A multi-hit ν-OTDR shares a similarity with a gated ν-OTDR in that fiber under test is virtually divided into many smaller sections. However, for a multi-hit ν-OTDR, the APD is gated "on" for more than one region per output pulse. The result is that waveform acquisition speed is increased proportional to the number of active regions.

This is in contrast to a single-hit gated ν-OTDR in which only one backscattered pulse is measured for each output pulse. Since the RAS system needs the data as fast as possible, a multi-hit approach is used, and the region size is set to the same period as the recharge time of the avalanche photodiode detector. In the case of the Perkin-Elmer SPCM-AQR used in this design, the recharge time is 30ns. Multi-hit and gated ν-OTDRs are typically more complex to design, as they require precise timing. As will be explain in the following chapters, the Event Detector and Locator employs a variant of the multi-hit approach that simplifies the design at a cost of decreased dynamic range.

3.0 System Design Concepts

3.1 Introduction

This chapter aims to explore the system design from a high level. Sample system deployments will be examined and major design choices will be described. A high level functional diagram will be presented that will lay the foundation for more detailed explanations in chapters 4 and 5.

The device's primary functions are detecting an event (seal breach) and locating where that breach is. As such, the device was given the name Event Detector and Locator (EDL). This device will replace the IDU and the OTDR from previous generations of RAS systems and provide a smaller, faster, more flexible and more secure hardware architecture.

3.2 Simple Deployment

To understand the role of the EDL in the RAS system, consider Figure 5. A simple deployment is illustrated in which the EDL is connected to a single loop of fiber with multiple seals along that loop. In the event of a breach, a local visual alarm on the EDL will activate, and the computer will be notified of an unacknowledged breach. The EDL will then begin taking ν-OTDR measurements in both the forward and reverse directions and will pass this data to the computer via an Ethernet connection. ν-OTDR scans are taken repeatedly until the user acknowledges the breach.

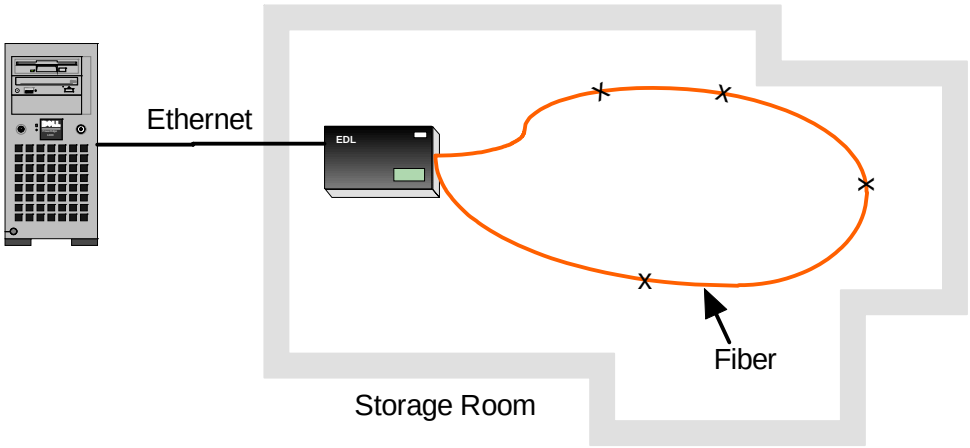

Figure 5: Single Loop RAS Deployment

The direct Ethernet connection between the EDL and computer can be established with a crossover cable. While the EDL can certainly be used in this way, it is a rare case in practice and does not leverage the networking features designed into the device. The next section will discuss a more common deployment arrangement.

3.3 Common Deployment

Most facilities have several rooms or vaults which often span multiple facilities. It is often desirable to monitor all RAS data in one central location. The EDL was designed as a network device that operates on existing Local Area Network (LAN) architecture or over the Internet. In either case, a group of EDL clients can communicate with a single server or host. Figure 6 and Figure 7 illustrate a more common deployment scheme.

Leveraging existing networks is a large advantage for potential deployments because it minimizes wiring and installation costs for most facilities. However, it places

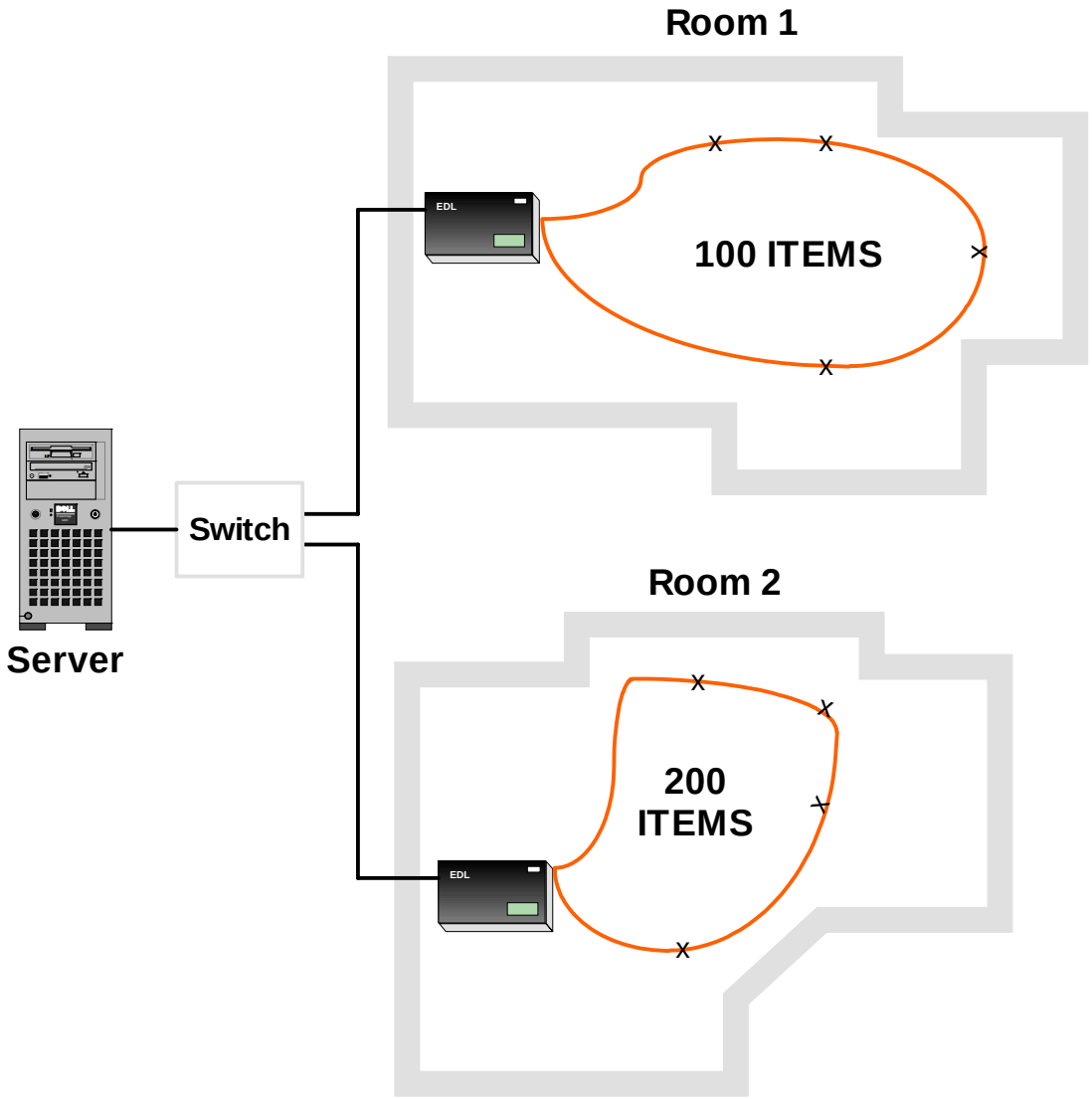

Figure 6: Typical RAS Deployment

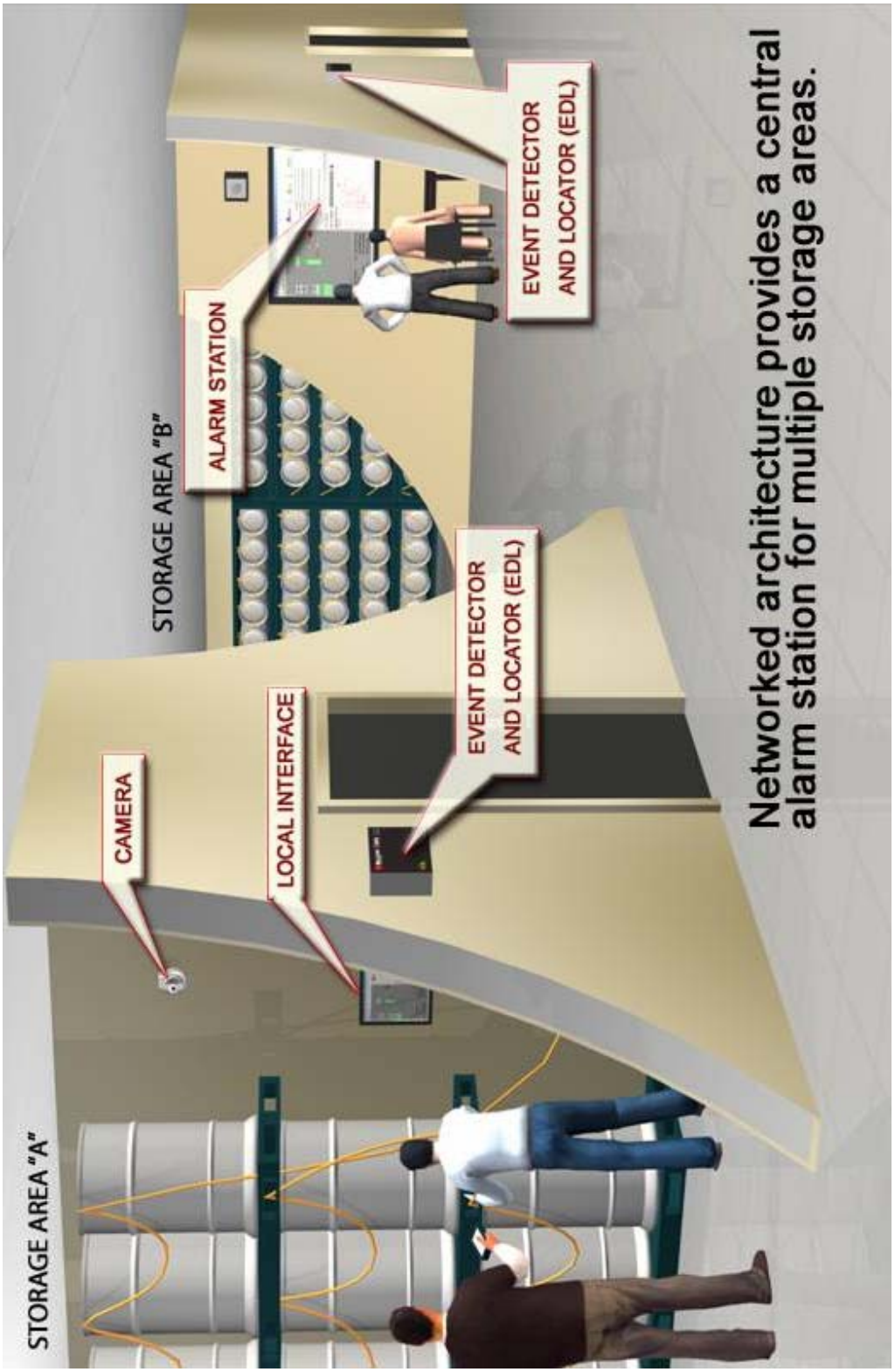

Figure 7: Typical RAS Facility Implementation

an additional burden on data security because the data is more exposed and can no longer be physically protected. To mitigate this risk, security features built into standard protocols were leveraged, including Hyper Text Transfer Protocol over Secure Socket Layer (HTTPS), the MD5 hashing algorithm and eXtensible Markup Language (XML). These tested and proven protocols greatly decreased software development time and effort by permitting the use of software libraries provided by the microprocessor core module vendor and the Microsoft .NET framework.

3.4 Event Detector and Locator Theory of Operation

A Field Programmable Gate Array (FPGA) was chosen to implement the ν-OTDR digital signal generation and timing as well as an encoding scheme for loop continuity verification. Off the shelf (OTS) equipment was used wherever possible, including an 8-bit microprocessor core module with on-board support for the Telecommunications Protocol / Internet Protocol (TCP/IP). OTS optical equipment and a modular power supply were used. Use of OTS equipment minimized the amount of custom circuit design and fabrication, which decreased development time.

Custom electronics were designed for support of the FPGA and microprocessor. Custom electronics were also designed to implement an alarm board, alarm interface and high speed laser diode driver. Firmware was developed for the FPGA and communication module in Very High Speed Integrated Circuit Hardware Description Language (VHDL) and Dynamic C. An advantage that comes with the use of an FPGA and the flexible microprocessor core module is that nearly all system functions can be modified with simple firmware changes.

A major goal of this design is to incorporate event detection and event location via a ν-OTDR into a single device. The EDL does just that, and it is easiest to understand the device if the reader continues to think of it as two instruments merged into one. Under normal operations, the EDL can be thought of as a simple fiber optic loop continuity checker (event detection instrument). It isn't until a breach is detected that the EDL begins to function as a ν-OTDR (event location instrument). The EDL theory of operation will be explained for each mode of operation in the next sections.

3.4.1 EDL in the Event Detection Mode

The diagram in Figure 8 shows the EDL in the event detection mode. The signal flow for this mode will now be discussed from a high level perspective. More specific details will be provided in the following chapters.

Starting at the FPGA, a 32-bit encoded digital word is generated and sent to the high speed laser driver. For each edge received here, a short duration (~800 picosecond) optical pulse is output from the laser diode. The light travels through a coupler which serves no purpose in this mode. From here the light goes through a 2x2 optical switch which is set in the straight through mode (this switch serves no purpose in this mode). The light then travels out of the EDL and through the fiber optic loop. Once back in the EDL (assuming the loop was not breached) the light travels through the 2x2 optical switch to the 1x2 optical switch. As indicated in Figure 8, the switch is set to allow light to pass to the variable attenuator and on to a multimode-to-singlemode adaptor. Finally, the light is sensed by the Single Photon Avalanche photodiode Detector (SPAD). The light pulse is now digitized and is input back to the FPGA where it is processed.

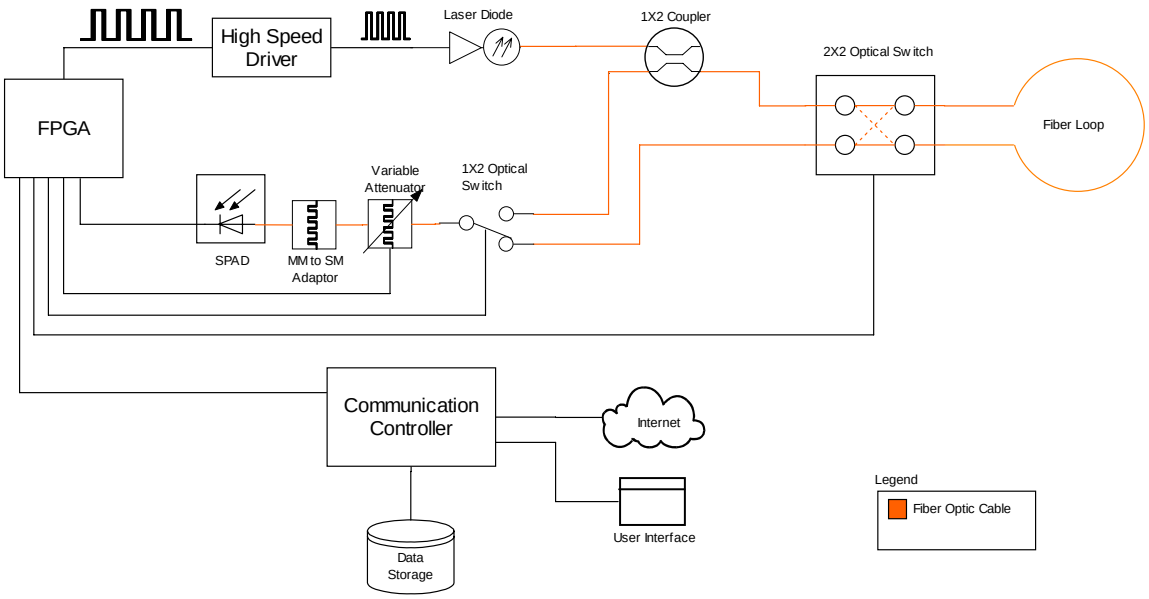

Figure 8: EDL Block Diagram in Event Detection Mode

3.4.2 EDL in the Event Location Mode

In Figure 9 the EDL is shown in the event location (or ν-OTDR) mode. In this case a TTL pulse is generated by the FPGA. The pulse is converted to a short optical pulse by the high speed driver and laser diode. The light passes through the coupler and through the 2x2 switch. The light then travels through the loop until it hits the breach. A Fresnel reflection occurs at the breach, which sends a minute amount of light back toward the transmitter. The 1x2 coupler directs some of this light to the 1x2 optical switch, through the attenuator and multimode-to-singlemode adaptor and into the SPAD. The signal is digitized by the SPAD and a signal is output to the FPGA for timing analysis.

The 2x2 switch is used for varying which end of the fiber the ν-OTDR uses for measurements. Being able to change this switch provides the ability to locate a seal

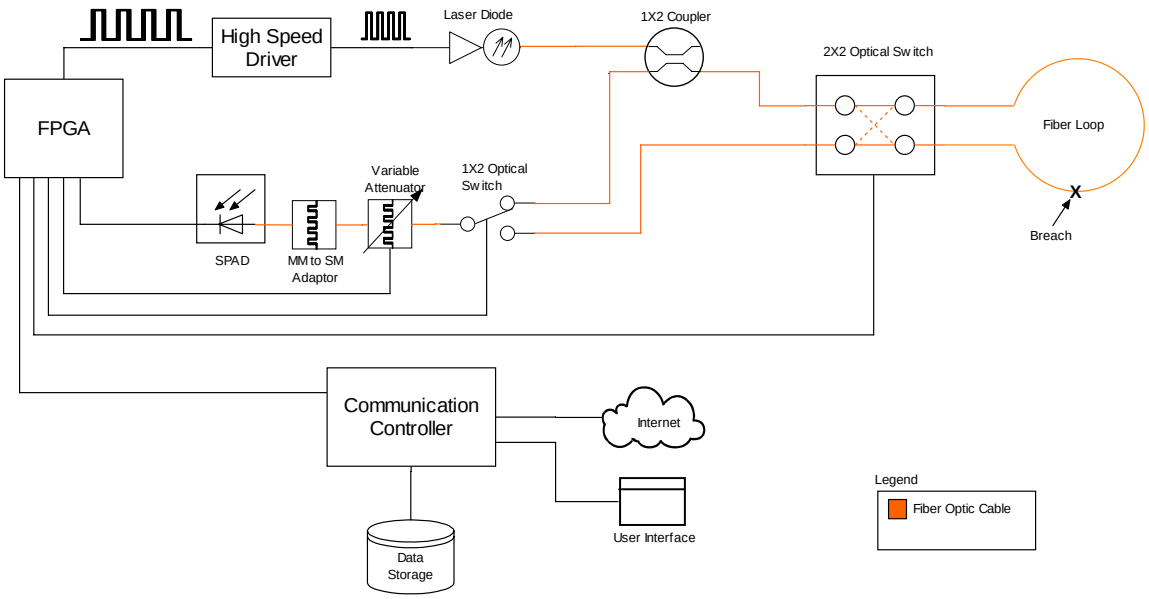

Figure 9: EDL Block Diagram in Event Location Mode

breach from both ends of the fiber. It also allows determination of an open span of fiber caused by two open seals.

It is obvious that Figure 8 and Figure 9 are nearly identical. This is by design and results in a smaller, lower cost and better performing instrument. Common components are used where possible between the event detection mode and event location mode. From a hardware perspective, the 1x2 optical switch controls the EDL mode of operation.

The use of the sensitive SPAD results in a ~7dB improvement in dynamic range for the event detection part of the instrument. Because each ST style connector has an average loss of 0.2dB (varies by type and manufacturer), the net result is an increased capacity of about 35 seals per EDL over a standard photo detector based system. It should be mentioned that the signal must be heavily attenuated to avoid saturation of the SPAD, but this is not a problem as attenuators are already in place for ν-OTDR operation.

4.0 Hardware Design

4.1 Introduction

The components of the hardware design include the FPGA and support circuitry, a low cost picosecond laser driver and laser diode, an avalanche photodiode detector, an 8-bit microprocessor core module and support circuitry, alarm circuitry, power module, power filtering, a liquid crystal display and optics components including a coupler, switches and attenuator. A picture of the hardware is shown in Figure 10. In the following sections, each portion of the hardware design will be described in detail.

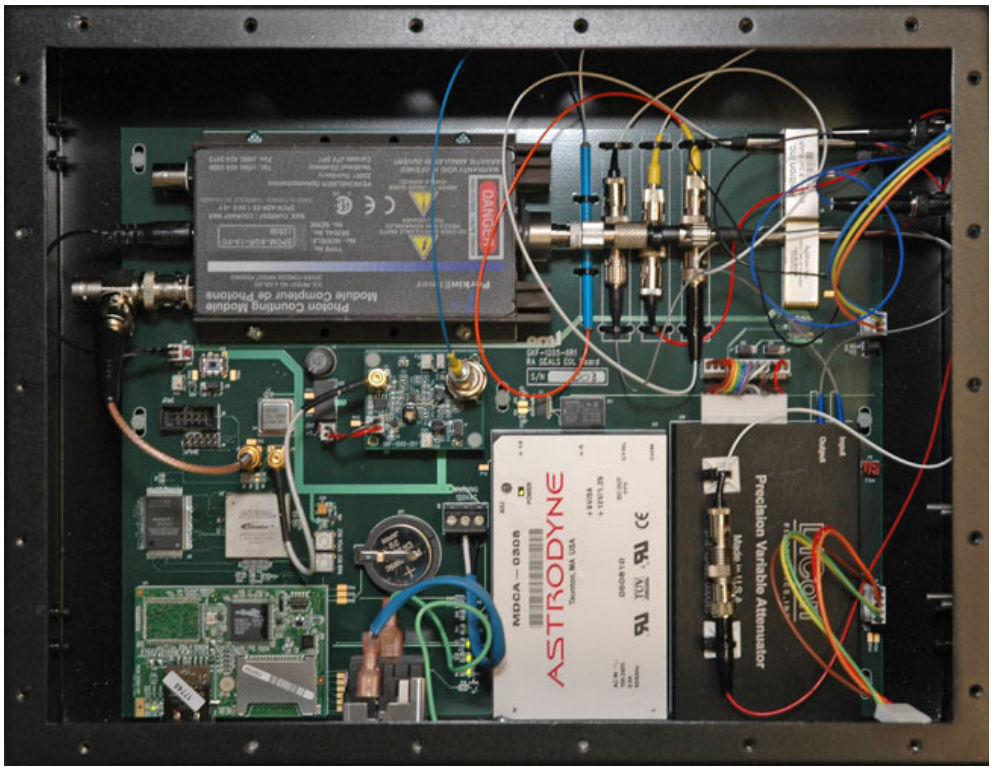

Figure 10: EDL Circuitry

4.2 Power Circuitry

The EDL is designed to operate from 120 volt 1 phase AC power (60 Hz). In order to save space and to minimize design time, an Astrodyne MDCA-0308 AC-DC power module was used. This module accepts input voltages from 85 VAC to 265 VAC and provides two outputs, 5 VDC @ 3A and 12 VDC @ 1.25A [12]. A supply ripple filter circuit consisting of parallel 100μF tantalum and 1.0μF ceramic capacitors is applied at each output of the power supply module.

The various components in the EDL necessitate multiple supply voltages. The FPGA requires 3.3 VDC and 1.5 VDC. The microprocessor board requires 3.3 VDC, the alarm circuitry and optics require 5 VDC, and the laser diode driver requires an isolated 5 VDC supply. To generate 3.3 VDC and 1.5 VDC for the FPGA, two LM317D2T adjustable regulators are used to step down the 5 VDC supplied from the Astrodyne power module. The laser diode driver's isolated power is generated by using an isolated dc-dc converter (NML1205). The output ripple is reduced by an L-C circuit consisting of a series inductor and parallel capacitor (relative to the input). The L-C circuit is a low-pass filter and is tuned to a cutoff frequency of about 500Hz. To isolate the laser diode driver circuit, a separate ground plane is used and is connected to the main ground plane through a high frequency noise suppressing ferrite bead.

4.3 FPGA and Support Circuitry

An Altera Stratix EP1S10F484C7 FPGA was used for this design. The FPGA is based on a 1.5V, 0.13-um process. It has 10,570 logic elements, 920,488 RAM bits, 6

phase-locked loops (PLLs) and 426 user Input/Output pins [13]. The FPGA is clocked by an external 80 MHz clock.

The Stratix FPGA was chosen based on performance, cost, the inclusion of dedicated RAM and the availability of PLLs. The included RAM allows ν-OTDR data to be stored on-chip, which reduces board complexity and increases read/write speed. A PLL is used to generate several phase-shifted clocks which drive a VHDL sampler at a rate of 8 times the clock frequency. This topic will be discussed at length in Section 5.4.2.

The FPGA must be supplied with 3.3V and 1.5V for internal logic and input buffers. In addition to the use of bypass capacitors at each power entry point, Altera recommends that the board layout be customized for the power and ground pins of each PLL. Since the PLL circuits contain analog components embedded in a digital device, the power pins are isolated to improve noise immunity [14]. To maintain this isolation several board layout recommendations are provided by Altera. The chosen method was a thick power trace from the power supply to each PLL power pin. Each power pin is decoupled with a 0.1μF and a 0.001μF parallel combination of ceramic capacitors.

A reset switch is included in the design. This switch connects an FPGA pin to ground when pressed. The switch is useful for debugging and is not used during normal operation of the instrument.

An Altera EPC8 contains 8Mbit of flash memory used to store the configuration data for the FPGA. At power-on, the EPC8 is configured to automatically load the configuration data to the FPGA. The EPC8 supports the Joint Test Action Group (JTAG) interface which enables in-system programmability (ISP) of the flash memory [15]. This

feature provides the ability to reprogram the FPGA in the field with a notebook computer. An appropriate connector is included on the main circuit board to support ISP.

4.4 Low Cost Picosecond Laser Driver

As mentioned in Section 2.3.4, minimizing the laser pulse duration is critical to maintaining high spatial resolution measurements. After an exhaustive vendor search, the only viable solution was the id300 offered by ID Quantique in Switzerland. The id300 produces 300 picosecond pulses but is cost prohibitive. The RAS system can meet defined specifications with pulses up to 1.5ns full width half maximum (FWHM). This requirement stems from the fact that the phase delay based sampler on the FPGA (to be discussed later) has a maximum sampling rate of 8 times the clock frequency. At a clock frequency of 80 MHz, the sampling rate is 640 MHz or once every 1.56 nanoseconds. Spatial resolution of the ν-OTDR is therefore constrained by the sampling rate as long as the laser pulse duration is kept below 1.56 nanoseconds.

The ratio of speed of light in a vacuum to the speed of light in a material is called the index of refraction. The Corning multimode fiber used for RAS has an effective group index of refraction of 1.496 at 850nm [16]. The phase velocity, v, of light in the fiber is inversely proportional to the refractive index as shown in Equation 2, where c is the speed of light and n is the group index of refraction [17]. Light therefore travels about 67% as fast in the fiber as it does in a vacuum. 1.56 nanoseconds equates to 0.31 meters in the fiber. If the laser pulse is kept below 1.56 nanoseconds then the resolution of the ν-OTDR will be 0.31 meters or about a foot.

$$v = c/n \qquad \text{Equation 2}$$

To meet the cost requirement while still supplying pulse widths of ~800ps meant that a custom laser driver was necessary. The basic design concept is taken from a current switch circuit that is commonly used in emitter-coupled logic (ECL) gates. This core section of the laser diode driver is shown in Figure 11. The collector current differences in terms of the base-emitter voltages can be expressed as

$$i_{C1} - i_{C2} = \alpha_F I_{EE} \tanh\left(\frac{\upsilon_{BE1} - \upsilon_{BE2}}{2V_T}\right) [18]. \qquad \text{Equation 3}$$

Equation 3 reveals that a small change in the base-emitter voltage of Q1 with respect to Q2 or of Q2 with respect to Q1 causes nearly all of the current to switch from one transistor to the other. More importantly, both transistors are held in the forward-active region at all times. The "off" transistor flows a small current at all times and can switch rapidly into high conduction when its base-emitter voltage rises just a few tenths of a volt over the paired transistor. The saturation region is therefore avoided, allowing current to switch between the two transistors at sub-nanosecond speeds.

In Figure 11, the Q1 side of the switch is normally flowing. When the voltage at V2 increases so that V2>V1 by more than a few tenths of a volt, then current will flow though Q2. The amplitude of the current flowing through the switch (Q1 or Q2) is set by the current source composed of Q4, R17, R18 and R16. The current source is variable based on the setting of potentiometer R16. Another current source composed of Q3, R15, R10 and R7 provides a constant bias current to the laser diode D1. It was expected that the diode would switch faster if biased just under its minimum lasing current. In practice, the bias current does not increase switching speed but does cause the laser diode to output

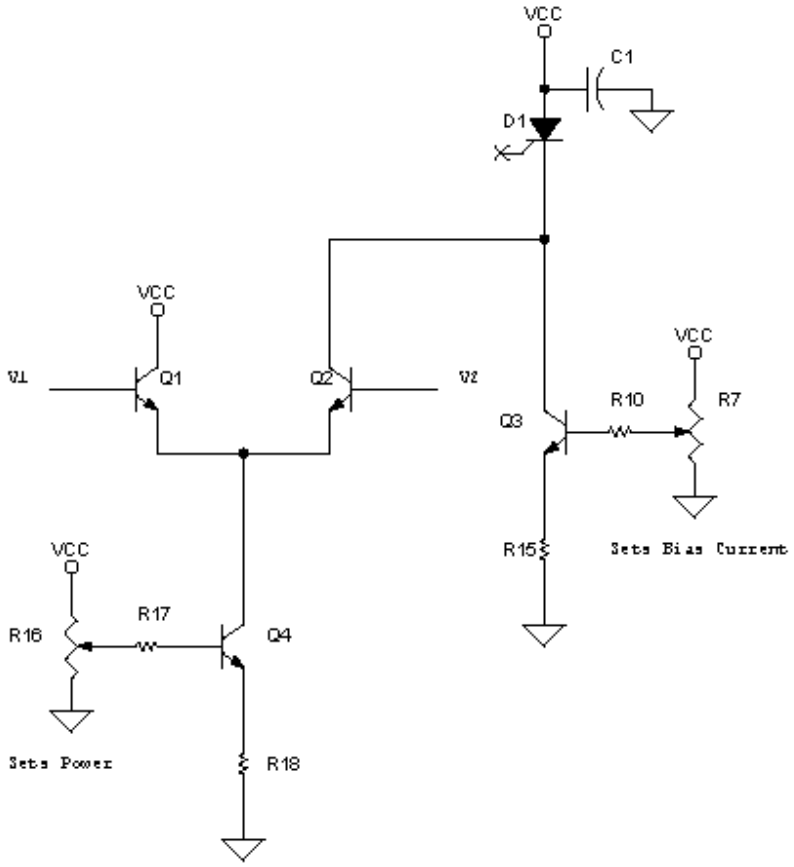

Figure 11: ECL Based Core of Picosecond Diode Driver

steady low-level light which increases noise in the system. The bias current is set to zero in the final design.

The next design challenge was to generate the switching voltages at the inputs of Q1 and Q2. The inputs to Q1 and Q2 must be at ECL voltage levels and have pulse widths of ~800ps. In order for the transistors to remain in the forward-active region, they must be held at a voltage above breakdown and switched around that voltage. An easy way to meet the ECL requirements was to drive the current switch with an ECL device. The input stage is shown in Figure 12.

The circuit works by accepting a transistor-transistor logic (TTL) level pulse sourced from the FPGA and converting it to positive emitter-coupled (PECL) levels. One side of the PECL signal, Q, is then split it into two signals while the other side ($\overline{Q}$) is terminated and not used. The top side of the split signal is biased by R1 and R4 to be at a lower voltage than the bottom side which is biased by R11 and R14. Under quiescent state the input to the comparator at the negative terminal is greater than the positive terminal. This translates into Q1 being active in the quiescent state as expected. When a pulse is received, the response at the negative input of the comparator lags that of the positive input because of the time it takes to charge C3. This effect is shown in Figure 13. During the lag, the signal at the positive side of the comparator (in purple) becomes greater than the signal at the negative terminal (in teal). Hence a very short PECL pulse is output from the comparator for each incoming pulse. The length of the pulse is proportional to the value of C3 (larger = longer pulse). It does not matter how long the TTL input pulse is, but duty cycle must be such that the capacitor has time to discharge.

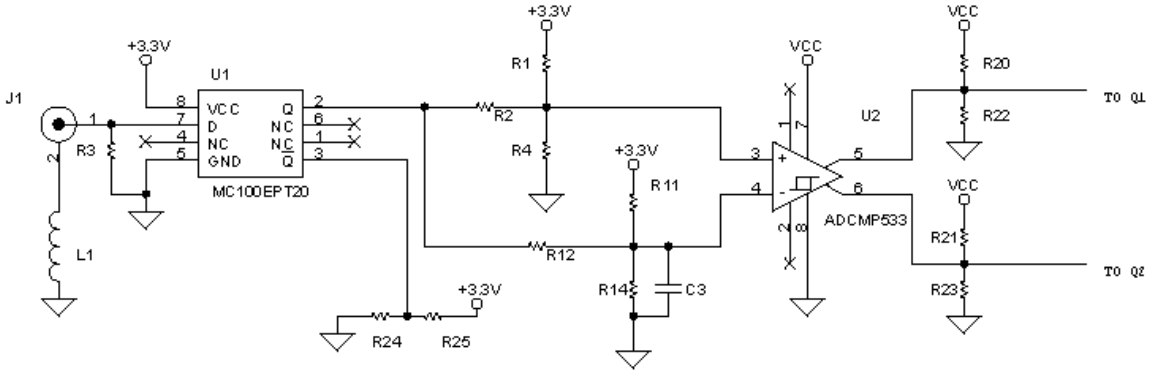

Figure 12: Input Stage of Picosecond Diode Driver

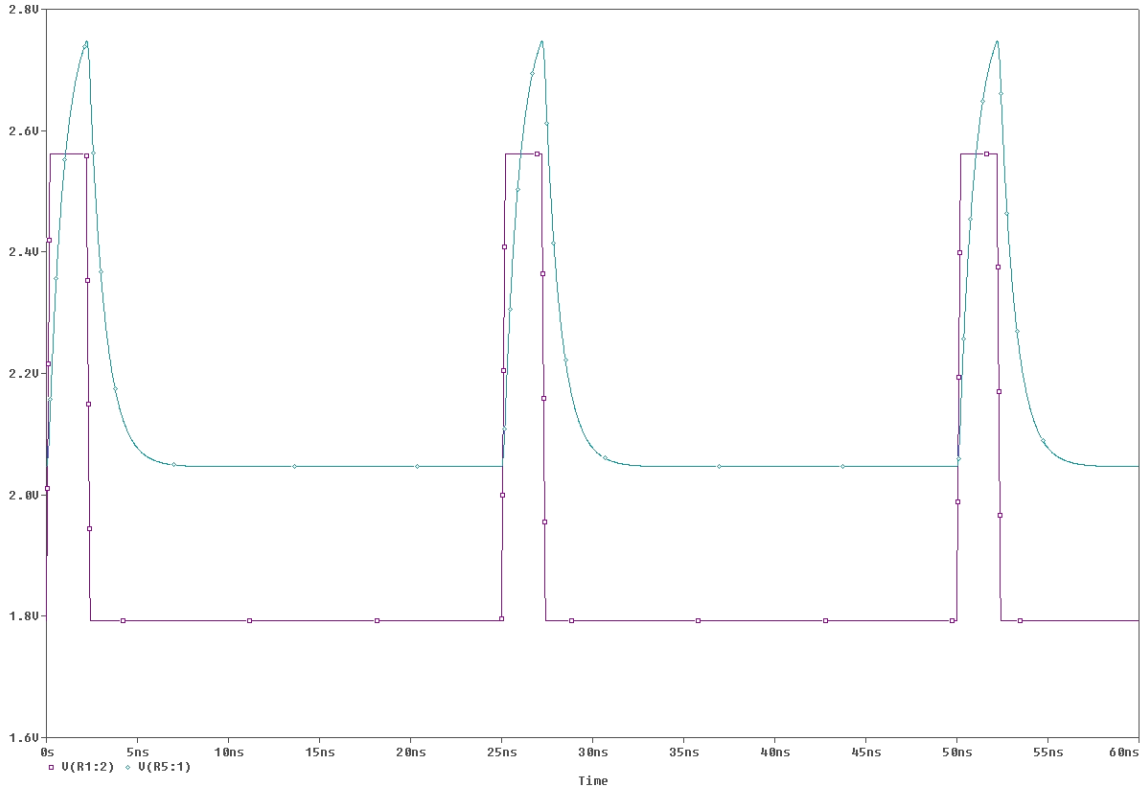

Figure 13: Picosecond Driver Delayed Pulse Response

The PECL outputs of the comparator are terminated and directly feed transistors Q1 and Q2 in the current mirror.

The resulting circuit cost less than $20 in parts and consumes only 2.38 square inches of printed circuit board real estate. The value of C3 was chosen so that 800ps FWHM optical pulses are generated. The waveform in Figure 14 shows the optical pulse acquired using a Thor Labs SV2 photo detector and a 1 GHz oscilloscope.

The driver fails to work at pulse widths shorter than about 650ps. It is assumed that this limitation is due to differential input voltage converging to near zero as the delay caused by smaller values of C3 gets shorter. This can be visualized as the teal waveform becoming more similar in shape to the purple waveform in Figure 13. This is not a surprise as the comparator switching performance declines as the input voltage difference decreases [19].

4.5 Control and Communication Module

Control and communication for the EDL is handled by a Rabbit Semiconductor RCM3360 core module. This module (shown in Figure 15) contains a Rabbit 3000 8-bit microprocessor running at 44 MHz, a 10/100Base-T Ethernet chip, 512KB flash, 512KB program Static Random Access Memory (SRAM), 512KB data SRAM and support for an extreme digital (also know as xD, a memory card format used by Olympus and Fuji) memory card up to 128 MB. The module features 52 digital input/output ports and 6 serial ports [20].

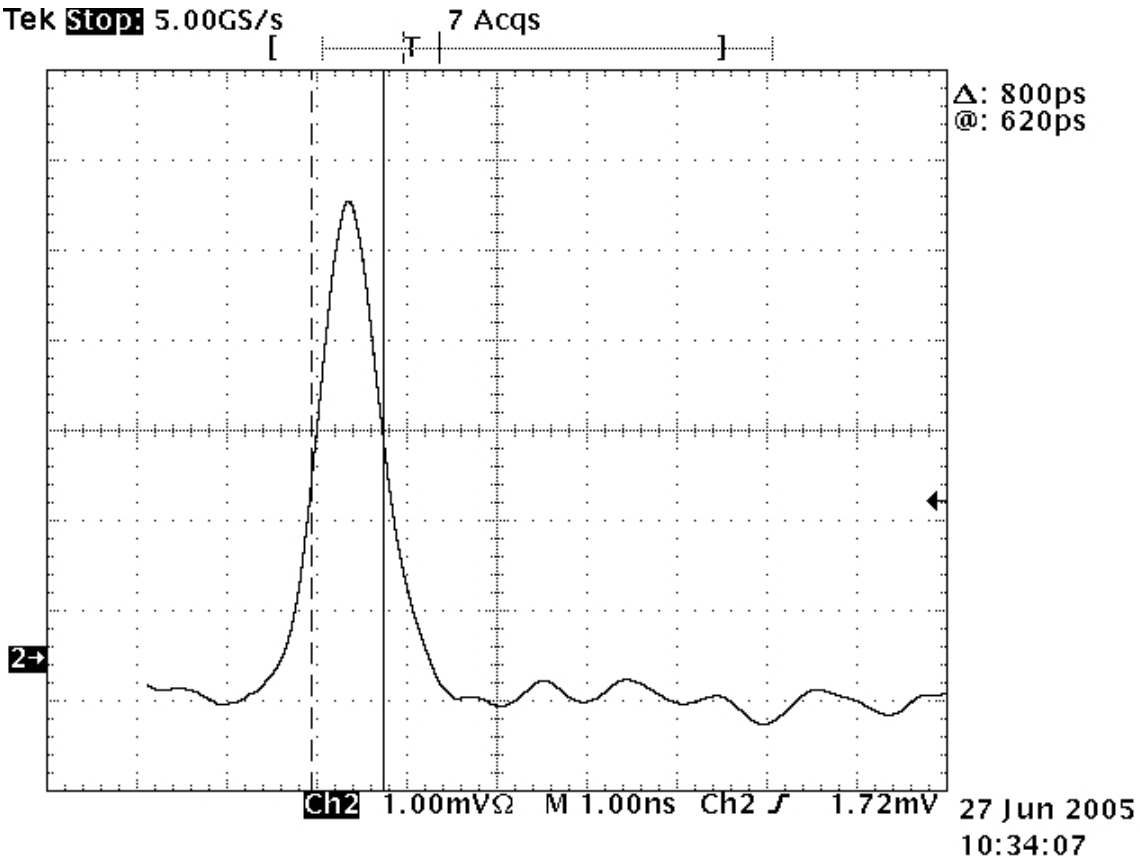

Figure 14: Picosecond Driver Optical Output Pulse

Figure 15: Rabbit RCM3360 Core Module

The RCM3360 includes a development environment called Dynamic C which integrates code editing, compiling, linking, loading and debugging of the Rabbit 3000 processor into one package. Dynamic C follows ANSI C standards but adds several enhancements that help develop embedded applications [21]. These enhancements are listed below (taken from the Dynamic C user manual).

- Function chaining allows segments of code to be embedded within one or more functions.
- Costatements allow concurrent parallel processes to be simulated (much like multithreading).
- Cofunctions allow cooperative processes to be simulated in a single program.
- Slice statements allow preemptive processes in a single program.
- Dynamic C supports embedded assembly code and stand-alone assembly code.
- Dynamic C Has *shared* and *protected* keywords that help protect data shared between different contexts or stored in battery-backed memory.
- Dynamic C supports features that allow the programmer to make fullest use of extended memory.

The RCM3360 is a plug in module that requires 3.3 VDC to operate. The programming port is self contained on the module. The main board contains a reset switch wired to the reset pin on the RCM3360. This switch is useful for debugging and restarting code execution during testing but is not used in normal operation. The main board also contains a 3.3 VDC battery that is wired to the RCM3360 to maintain the real-time clock and provide data backup. Should the EDL lose power, the battery is used to preserve variables stored in flash memory. For example, breach acknowledgement state variables are held in battery-backed memory so the state remains after a power cycle.

4.6 Alarm Circuitry

The EDL contains a visual alarm that flashes in the event of an unexpected seal breach. The alarm output comes from the RCM3360 module and through a SN74LVCC4245 bus transceiver which converts to voltage levels from 3.3 VDC to 5 VDC. The outputs of the bus transceiver toggle a latching relay. This relay controls a 5 VDC signal to the light emitting diode (LED) indicator board. The relay circuit is shown in Figure 16. A latching relay is used to ensure that the alarm state remains set until it is acknowledged by a user, even if the power is cycled.

The output of the latching relay drives the LED indicator circuit in Figure 17. This circuit has a 555 timer running in the astable multivibrator mode driving one input of an AND gate. When the control input signal is high, the output of the AND gate follows the astable multivibrator output. The frequency of the astable multivibrator is given by:

$$f = \frac{1.44}{(R_{21} + 2R_{22})C_7} \quad [22]. \qquad \text{Equation 4}$$

The output of the AND gate drives the shutdown pin of a MAX1698 high-efficiency step-up current regulator. Under non-alarm conditions the current regulator is held in shutdown mode. During an alarm condition the current regulator is continuously cycled between on and shutdown which causes the LED array to flash. The high intensity surface mount white LEDs are arranged in a tight 4x3 cluster.

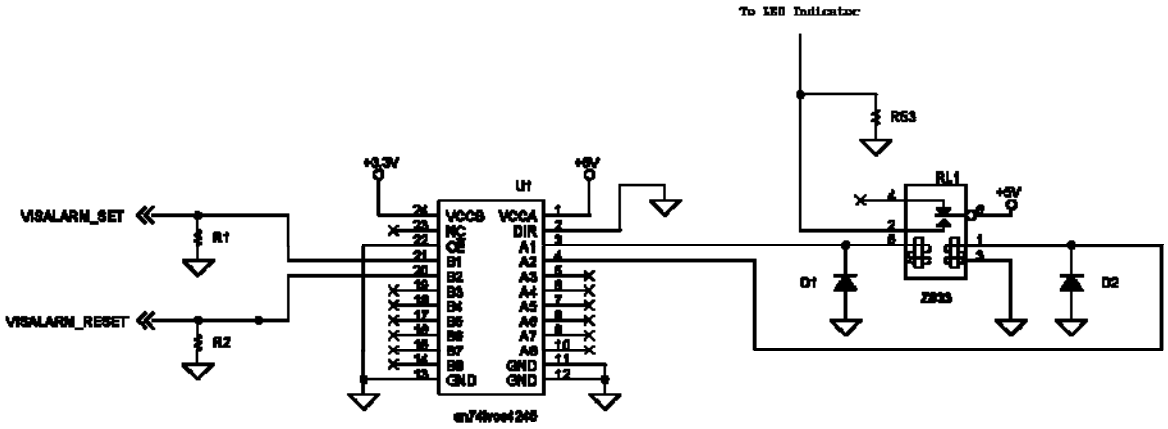

Figure 16: Alarm Interface Circuitry

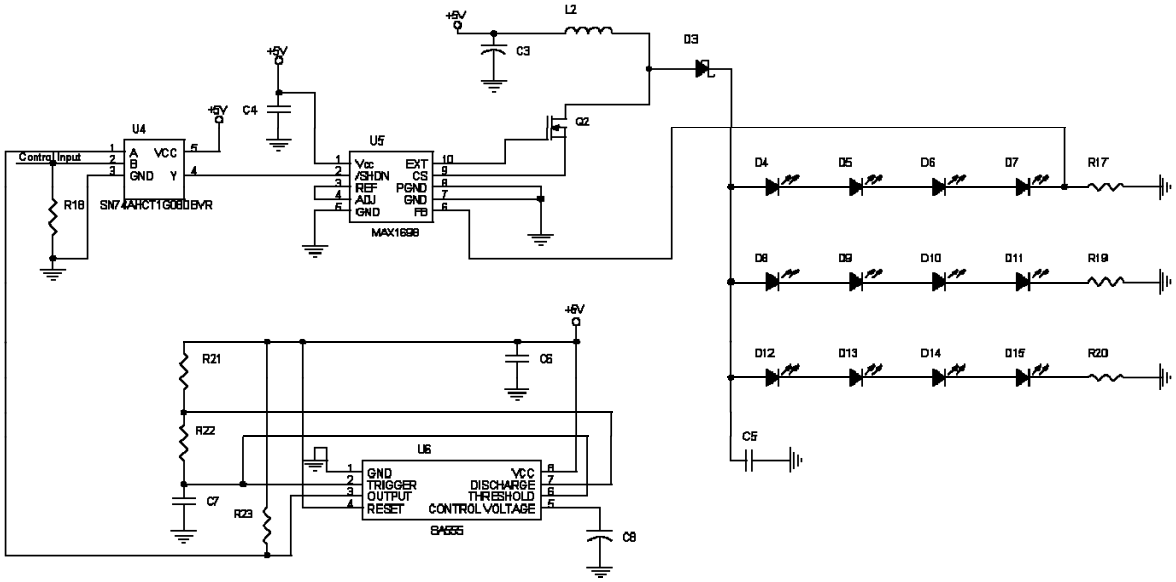

Figure 17: EDL Light Emitting Diode Circuit

4.7 Optical Components

The optical components used in the design are commercially available off-the-shelf items. The fiber optic cable used in the RAS system is 62.5/125 micron multimode fiber (62.5 micron core, 125 micron glass cladding). Multimode fiber receives its name due to a core that is much larger than the wavelength of transmitted light. The large core allows multiple optic pathways. In typical applications, multiple wavelengths (modes) are used in the same fiber. Multimode fiber was originally used because the connections are generally higher loss (cause large Fresnel reflections) and are therefore easier to see with an OTDR [23]. Today, multimode fiber is used mainly because of legacy equipment (i.e. the ability to retrofit older installations with new hardware without replacing fiber). Each optical component is described in the following sections.

4.7.1 Laser Diode

The laser diode is manufactured by PD-LD Inc., part number PL85R001ST73-T-O. It operates at 850nm with a minimum fiber coupled power of 1mW. The threshold current is 30mA. The diode is soldered directly to a circuit board and is coupled to 62.5/125 multimode fiber via an ST style connector.

4.7.2 1x2 Coupler

The 1x2 coupler is manufactured by Gould Fiber optics, part number 15-21200-50-1212. The role of the coupler is to direct backscattered light to the receiver when the EDL is in ν-OTDR mode. The coupler is non-directional, meaning that light entering any node will be distributed evenly between the other two nodes. The coupler is supplied with three six inch 62.5/125 multimode fiber pigtails with ST connectors.

4.7.3 Optical Switches

A 1x2 optical switch and a 2x2 optical switch from Agiltron are used. Both switches are controlled with a simple 5 VDC interface. The 1x2 switch (part number LB-12011316) is normally set so the EDL is in the event detection mode (see Figure 8). The switch changes state when 5 VDC is applied. The switch is non-latching and reverts when voltage is removed. The 2x2 switch (part number LB-22011316) is normally set to the pass through mode, which sets up the ν-OTDR for a forward direction scan as in Figure 8 and Figure 9. When a reverse direction scan is requested, 5 VDC is applied, changing the optical path to that of Figure 18. The optical switches have a cycle time of less than 10ms and are rated for 10 million cycles.

4.7.4 Dicon Fiber Optic Variable Attenuator

As previously mentioned, both the ν-OTDR and event detector require careful attenuation. Attenuation characteristics of the fiber under test are based on fiber length, number of connections (seals) and quality of connector couplings. Since these factors are unknown for a newly setup EDL, a variable attenuator is required to adjust the receiver sensitivity to match the fiber under test. The Dicon NAT500-8-62-ST attenuator has a

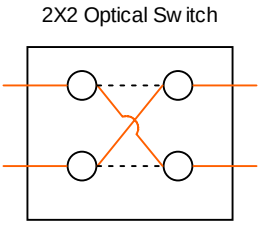

Figure 18: 2x2 Optical Switch in Reverse Mode

30dB range and can be controlled via the I^2C protocol (a two-wire bi-directional communication protocol developed by Philips) [24]. During startup, the attenuator is used to calibrate the system for the characteristics of the fiber attached. During ν-OTDR operation, the attenuation is set to levels specified by the user during characterization. Calibration and characterization will be discussed in more detail in Section 5, Software Design.

4.8 Perkin-Elmer Single Photon Avalanche Photo Detector

The Perkin-Elmer SPCM-AQR single photon avalanche photo detector (SPAD) uses a thermoelectrically cooled, temperature controlled silicon avalanche photodiode with a circular active area. For each photon detected, a TTL pulse is triggered with an amplitude of 2.5 volts (minimum) into a 50Ω load. The pulse lasts 35ns and is followed by a 50ns dead time [25]. The SPCM-AQR is available in several models corresponding to average dark count (the number of false outputs triggered per second when no light is present on the detector). The dark counts range from 500 to 25 counts per second. The EDL has been tested with the 250 count/second and 100 count/second models and no discernable difference was found in performance. The price of the device increases dramatically as the dark count decreases.

The SPCM-AQR comes with a fiber optic connection and supports gating via a TTL level input. When the voltage at the gating input is held high, the device will not respond to incoming light. When it is low (or not connected) an output will be generated for each received photon (recharging time still applies). The EDL operates in a multi-hit

mode based on the 50ns dead time of the detector and does not rely on the SPCM-AQR gating input. More details about how gating is implemented will be given in Section 5.4.

The SPCM-AQR operates from a single 5 VDC supply. It is optically sensitive from 400nm up to 1050nm, with a peak near 700nm as shown in Figure 19. A system wavelength of 850nm was chosen based on the optical efficiency of the SPAD and the availability of commercial optical equipment at that wavelength.

4.9 Display

The system status of the EDL is continually updated via a panel mounted liquid crystal display (LCD). The LCD is a Matrix Orbital model GLK12232-25-WBL. The

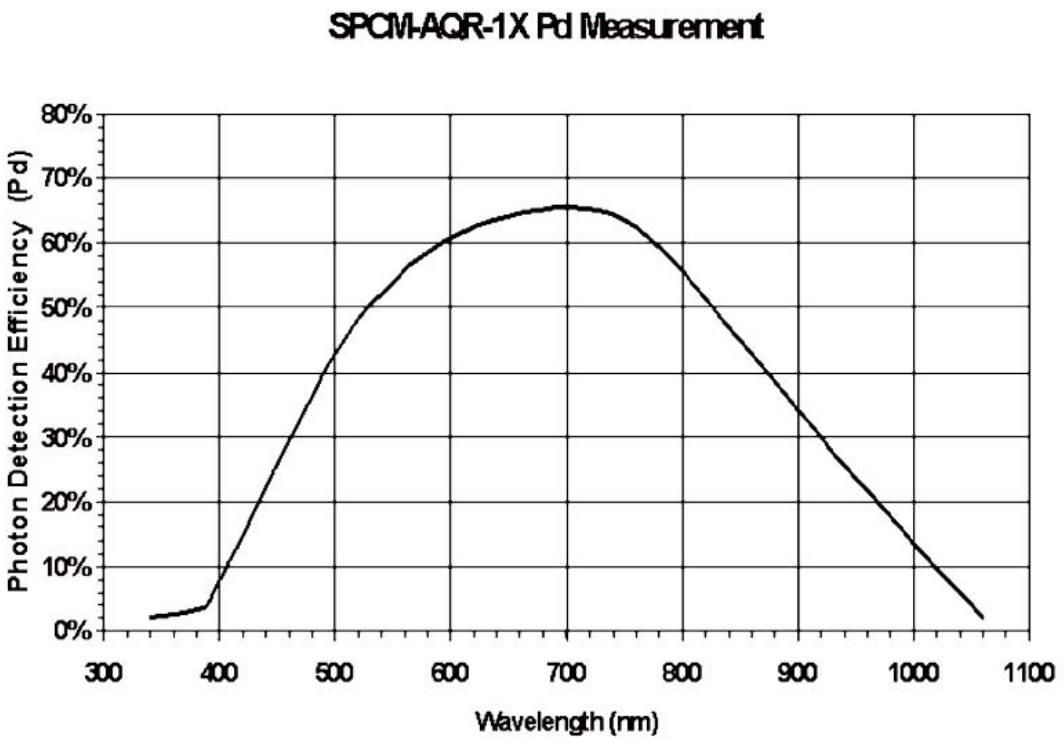

Figure 19: SPCM-AQR Single Photon Counter Optical Efficiency

display has a 122 x 32 pixel graphic display, supports built-in and user-supplied fonts, has a white backlight and supports I^2C communication [26]. The display is powered by 5 VDC and is controlled by the Rabbit RCM3360 module using the I^2C bus.

4.10 Packaging

The components were assembled on a single circuit board measuring 8x10 inches. An enclosure was fabricated to house the circuit board and provide mounting locations for the alarm board, LCD and optics connectors. The packaged EDL complete with silk-screening is shown in Figure 20.

Figure 20: Event Detector and Locator Enclosure

5.0 Software Design

5.1 Introduction

The Event Detector and Locator software consists of firmware for the FPGA, embedded software for the microprocessor and a graphical user interface (GUI) host running on an external computer. The firmware for the FPGA was developed in a mixture of VHDL and Altera's Quartus II block diagram editor. The firmware for the RCM3360 core module was developed in Rabbit Semiconductor's Dynamic C. The PC software was developed in Microsoft C# .NET. These three applications work together and form a software hierarchy that makes the EDL fully operational. In the following sections, the software will be described with respect to each major software function including mode control, loop continuity detection, ν-OTDR measurements, EDL communication/control and the external GUI/host.

5.2 Mode Control Logic in VHDL

As previously mentioned, the EDL is really two instruments in one. This distinction is easily visualized by examining the FPGA flowchart in Figure 21. Notice that the flowchart has two distinct columns. The left column provides the functionality for loop continuity detection. The right column provides the ν-OTDR measurements. The selection of which of the two columns is active is based on the *scan* input to the FPGA.

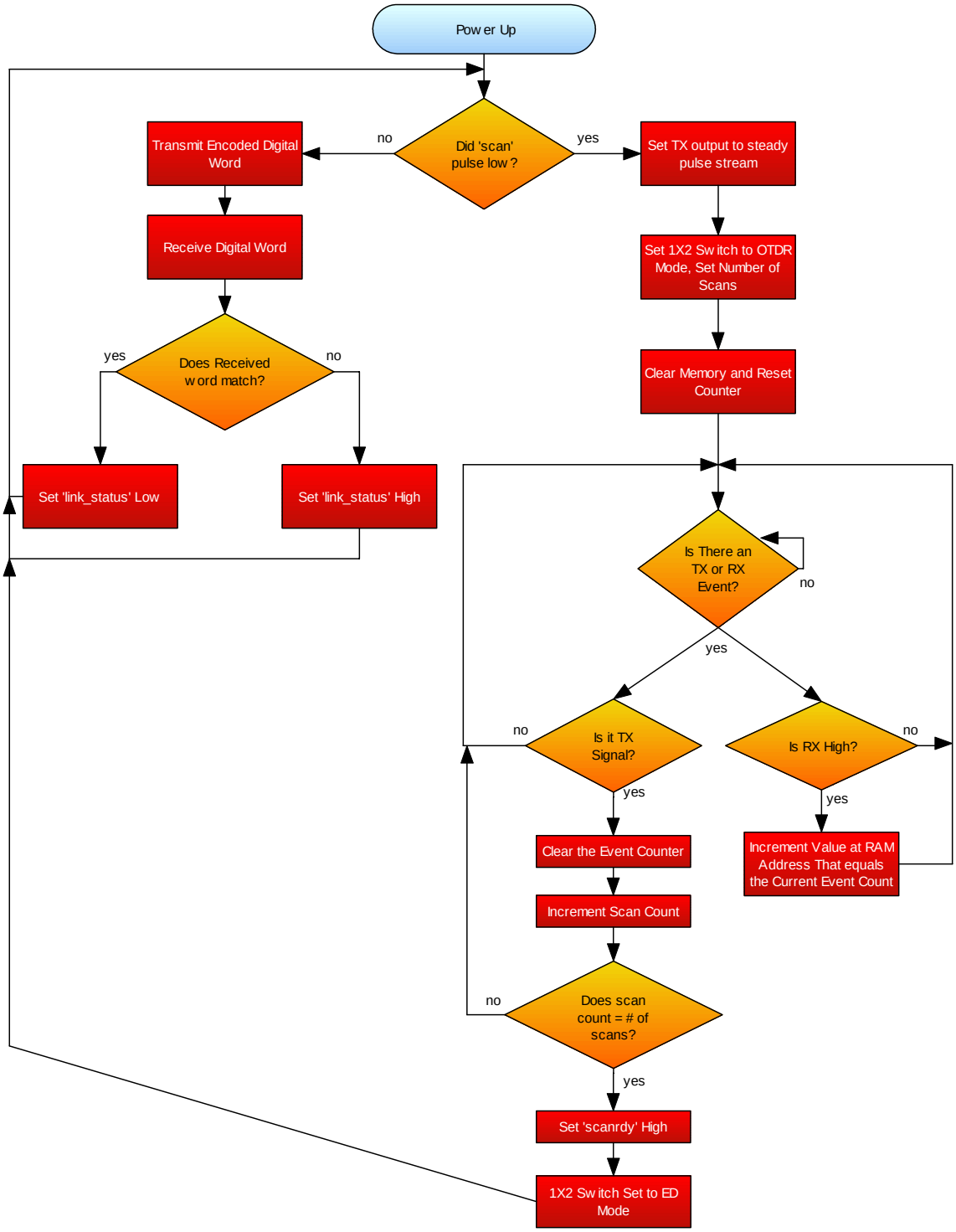

Figure 21: FPGA Software Flowchart

At power up, assuming *scan* is unchanging, the loop continuity detector executes. When a high to low pulse occurs on the *scan* input, the right side (ν-OTDR) of the flowchart is activated and executes until completion. This mode switching is controlled by a finite state machine implemented in the VHDL process shown in Figure 22. With each clock edge, a synchronized copy of the scan input (*scan_latched*) is compared to logic zero. When the condition is true (meaning a ν-OTDR waveform has been requested), the transmitted signal (TX) is changed from the coded word used for event detection to the steady stream of pulses needed for the ν-OTDR. The 1x2 optical switch is then set to the ν-OTDR mode (see Figure 9).

The mode control state machine remains unchanged until the input *scanrdy* goes high. This signal indicates that the ν-OTDR has completed taking measurements. The mode is now switched back to event detection.

5.3 Loop Continuity Detection in VHDL

The left side of Figure 21 represents the loop continuity detection (or event detection) algorithm. In this mode, the FPGA continually generates 32-bit encoded words, transmits them and verifies that the received words match what was transmitted. Using encoded words provides a layer of security by greatly decreasing the feasibility of injecting a forged signal into the fiber and spoofing the receiver.

5.3.1 Encoded Word Generation

A 32-bit word is generated and transmitted once every 20μs. The encoded word is the result of a 32-bit wide randomly-seeded linear feedback shift register (LFSR). The

```
mode_control: process (RST, CLK)
begin
      if RST='0' then
            modeFSM <= EDmode;
            X12SET1 <= '0';
      elsif rising_edge(CLK) then
            case modeFSM is
            when EDmode =>
                  if (scan_latched = '0') then
                        TX <= OTDR_TX;
                        X12SET1 <= '1';   --switch to OTDR mode
                        modeFSM <= OTDRmode;
                  else
                        X12SET1 <= '0';   --switch to ED mode
                        TX <= ED_TX;
                        modeFSM <= EDmode;
                  end if;
            when OTDRmode =>
                  TX <= OTDR_TX;
                  if (scanrdy = '1') then
                        X12SET1 <= '0';   --switch back to ED mode
                        modeFSM <= EDmode;
                  else
                        modeFSM <= OTDRmode;
                  end if;
            when others =>
                  X12SET1 <= '0';
                  TX <= ED_TX;
                  modeFSM <= EDmode;
            end case;
      end if;
end process mode_control;
```

Figure 22: VHDL Mode Control Process

LFSR is seeded at power-up and a new word is generated after each transmission. This results in a sequence of $2^{32} = 4{,}294{,}967{,}295$ 32-bit words. The sequential output of the LFSR is pseudo-random.

A new LFSR value is generated with each master clock tick (12.5 ns). The value used for transmission is grabbed at a regular interval. While it is theoretically possible for someone to tap into the line and figure out the LFSR sequence, it is highly improbable. First, it is quite difficult and requires special tools to tap a fiber without disrupting the signal. Second, the transfer from real to forged signal would have to be seamless. This would require special tools and a very high level of expertise. Third, it would take a very long time to figure out the LFSR sequence because the output word is a sample of an LFSR sequence (i.e. two sequentially generated LFSR words are never output). In typical safeguard applications, the intent of a seal is to increase the time and expertise required for access. By increasing the access time and requiring special tools, the odds are greatly increased that other security layers will detect the perpetrators.

5.3.2 Encoded Word Transmission, Reception and Verification

A 4-bit start word is appended to the LFSR generated word. The now 36-bit word is output serially at a baud rate of 50 Mbps. The transmitter VHDL process (TX_FSM) is implemented as a state machine and shown in Figure 23. During the load state, the LFSR process is sent a signal causing a new word to be generated and put in the signal *LFSR_POUT*. During the idle state, a clock signal *WORDGEN* starts the transmitting sequence. The LFSR generated word is shifted out the *TX* pin based on the

```
TX_FSM: process (RST, CLK)
begin
      if RST='0' then
            Tx_Reg <=(others => '0');
            TxBitCnt <= 0;
            TxFSM <= Seed;
            PDN_count <= 0;
            TX <= '0';
            PDN <='0';
            LFSR_SEED <= "32 bit seed word omitted for security";
      elsif rising_edge(CLK) then
            case TxFSM is
            when Seed =>
                  LFSR_LOAD <= '1';
                  TxFSM <= Idle;
            when Idle =>
                  LFSR_LOAD <= '0';
                  if WORDGEN='1' then --cue to start a cycle
                       TxFSM <= GenLFSR;
                  end if;
            when GenLFSR =>
                  TXWORD <= LFSR_POUT;
                  TxFSM <= Load_Tx;
            when Load_Tx =>
                  if TXCENTERCOM='1' then  --baud rate clock
                        TxFSM <= Shift_Tx;
                        TxBitCnt <= (NDBits + 4); --start plus data
                        Tx_Reg <= '0' & TXWORD & "1011";
                        -- adding start word
                  end if;
            when Shift_Tx =>
                  if TXCENTERCOM='1' then
                        TX <= Tx_Reg(0);
                        TxBitCnt <= TxBitCnt - 1;
                        Tx_Reg <= '0' & Tx_Reg (Tx_Reg'high downto 1);
                        if TxBitCnt = 1 then
                              TxFSM <= Stop_Tx;
                        end if;
                  else
                        TX <= '0';
                  end if;
            when Stop_Tx =>
                  TX <= '0';
                  if TXCENTERCOM = '1' then
                        PDN <='0';
                        TxFSM <= Idle;
                  end if;
            when others =>
                  TxFSM <= Idle;
      end case;
      end if;
end process TX_FSM;
```

Figure 23: Event Detection Signal Transmission VHDL

TXCENTERCOM clock. Once the entire word is transmitted, the state machine is returned to the idle state.

The signal reception is asynchronous. A start word is used instead of a start bit because of the spurious dark counts generated by the SPAD. If a single start bit were used then the receiver would mistake a dark count for the beginning of a transmission. A 4-bit start word was found experimentally to provide ideal immunity to SPAD induced noise without unnecessary complication.

The receiver state machine is shown in Figure 24. The receiver is modeled after a standard asynchronous receiver used for RS-232 communication. The difficulty associated with the receiver process is dealing with the short pulses output from the SPAD. Recall from Section 4.8 that the SPAD generates 30ns pulses for each photon received. The FPGA is running at 80 MHz (clock period of 12.5ns) so true mid-bit sampling is not possible. The best solution is to detect an edge and sample 12.5ns (one clock cycle) later. Since 15ns would be mid-bit, this is a good compromise.

The Rx_statmachine process shown in Figure 24 begins by waiting for a logic "1" on the *RX* pin. Once a "1" is received, the ComDelayPre state delays the next sampling cycle so that the following sample occurs as near mid-bit as possible. Similar delay states maintain mid-bit alignment throughout the entire word. The start word is verified during the first four samples. If the start word is corrupt, then it is assumed that the received signal was noise, so the receiver is reset. If the start word is correct, then the next 32 bits are sampled and stored for verification.

The received buffer is verified against the transmitted word 15μs after the initial word transmission. A time of 15μs is adequate to allow for light propagation in the fiber

```
Rx_statemachine: process (RST, CLK)
begin
      if RST='0' then
            Rx_Reg <= (others => '0');
            RXWORD <= (others => '0');
            RxBitCnt <= 0;
            RxFSM <= Wait_for_Start;
            ComDelayCount <= 0;
      elsif rising_edge(CLK) then
      case RxFSM is
            when Wait_for_Start =>
                  RxBitCnt <= 0;
                  if RX='1' then
                        RxFSM <= ComDelayPre;
                  else
                        RxFSM <= Wait_for_Start;
                  end if;

            when ComDelayPre =>
                  if ComDelayCount = ComDelayDivisor then
                  --adjust shift to get into the heart of the rx pulse
                        ComDelayCount <= 0;
                        RXFSM <= Edge_Rx;
                  else
                        ComDelayCount <= ComDelayCount + 1;
                        RXFSM <= ComDelayPre;
                  end if;
            when ComDelay1 =>
                  if ComDelayCount = ComDelayDivisor then
                        ComDelayCount <= 0;
                        RXFSM <= Edge_Rx;
                  else
                        ComDelayCount <= ComDelayCount + 1;
                        RXFSM <= ComDelay1;
                  end if;
            when ComDelay2 =>
                  if ComDelayCount = ComDelayDivisor then
                        ComDelayCount <= 0;
                        RXFSM <= Shift_Rx;
                  else
                        ComDelayCount <= ComDelayCount + 1;
                        RXFSM <= ComDelay2;
                  end if;
            when Edge_Rx =>
                  case RxBitCnt is  -- comparing against 110
                        when 1 =>
                              if Rx_Reg(Rx_Reg'high) = '1' then
                                    RxFSM <= ComDelay2;
                              else
                                    RxFSM <= Wait_for_Start;
```

Figure 24: Event Detection Signal Reception VHDL

```
                        end if;
                        when 2 =>
                              if Rx_Reg(Rx_Reg'high) = '0' then
                                    RxFSM <= ComDelay2;
                              else
                                    RxFSM <= Wait_for_Start;
                              end if;
                        when 3 =>
                              if Rx_Reg(Rx_Reg'high) = '1' then
                                    RxFSM <= ComDelay2;
                              else
                                    RxFSM <= Wait_for_Start;
                              end if;
                        when NDBits =>
                              RxFSM <= Stop_Rx;
                        when others =>
                              RxFSM <= ComDelay2;
                  end case;
            when Shift_Rx =>
                        RxBitCnt <= RxBitCnt + 1;
                        Rx_Reg <= RX & Rx_Reg(Rx_Reg'high downto 1);
                        RxFSM <= ComDelay1;
            when Stop_Rx =>
                        RXWORD <= Rx_reg(Rx_Reg'high downto 3);
                        RxRdyi <= '1';
                        RxFSM <= Wait_for_Start;
            when others =>
                  RxFSM <= Wait_for_Start;
            end case;
      end if;
end process Rx_statemachine;
end a;
```

Figure 24: Continued

over the 1-3km distances typically encountered in RAS deployments. If the received word does not match, a miss counter is incremented. Once the miss counter exceeds a preset threshold, the system is considered breached and the *link_status* bit is set high. The miss counter is reset to zero only by a successful match. The inclusion of a miss counter helps reduce false breach alarms caused by overdriving the SPAD, the presence of abnormally high continuous background light, or dark count noise introduced by the SPAD itself.

5.4 Photon Counting OTDR in VHDL and Block Diagram

When *scan* pulses low, the mode switches from event detection mode to ν-OTDR mode. The transmitted signal is changed from an encoded digital word to a steady pulse stream. The 1x2 optical switch is set to pass Fresnel reflections to the SPAD. The memory used to hold the ν-OTDR data is cleared and all counters are reset.

To understand the implementation of the ν-OTDR on an FPGA, it is critical to understand the basics of ν-OTDR design detailed in Section 2.3. Specifically note that at its core, a ν-OTDR is a device that generates a short pulse of light and measures with high accuracy the time it takes to receive a reflection. The timing resolution and accuracy are vitally important because they translate directly to the spatial resolution and accuracy of the system. To obtain the desired spatial resolution of one foot requires that the FPGA be able to sample at a rate of approximately once every 1.47ns (or ~680 MHz). This calculation is based on Equation 5 where *c* is the speed of light in a vacuum (3×10^8 m/s), R is the inverse refractive index (0.69) and 3.28 is the conversion factor from meters to feet. Sampling at the required rate of near 680 MHz on-chip required the

development on a high speed sampler based on three phase shifted clocks output from a phase-locked loop. This sampler will be discussed in Section 5.4.2.

$$\frac{1}{ft} = \frac{1}{c \times R \times 3.28 \frac{ft}{m}} \qquad \text{Equation 5}$$

5.4.1 ν-OTDR Signal Generation

The transmitted signal for the ν-OTDR is a steady stream of pulses 12.5ns (one FPGA clock cycle) in duration occurring at a rate of 30 kHz. The electrical pulse stream is converted into ~800ps FWHM optical pulses at a rate of 30 kHz by the laser diode driver. The pulse rate was chosen based on the light propagation time of the maximum length of fiber expected in a RAS installation. 30 kHz limits the maximum fiber length to 3.5 km.

5.4.2 ν-OTDR Control

The ν-OTDR is controlled at its highest level by the state machine shown in Figure 25. When *scan* pulses low, the state is changed to clear_mem and the Dual Port Random Access Memory (DPRAM) used to store the ν-OTDR waveform is cleared. The use of DPRAM provides simultaneous asynchronous access from two ports. This functionality is useful because the communication controller can read out the ν-OTDR waveform as the ν-OTDR controller is putting data in.

The ν-OTDR controller state machine then enters the scan_count1 state. It remains in this state until the ν-OTDR receives a hit from the SPAD signified by the *COUNT_RDY* signal going high. This timestamp is input to the controller in the form of a count that represents the amount of time since the output pulse was transmitted. This

```
state_tr: process (nRST, CLK, SCAN, COUNT_RDY, SCAN_DONE )
begin
if nRST = '0' then
      next_state <= idle;
      wraddr <= (OTHERS => '0');
      idat <= (OTHERS => '0');
      WREN <= '0';
      SCANRDY <= '0';
      COUNT_EN <= '0';
elsif rising_edge(CLK) then
      case next_state is
            when idle =>
                  wraddr <= (OTHERS => '0');
                  idat <= (OTHERS => '0');
                  COUNT_EN <= '0';
                  if SCAN = '0' then
                        next_state <= clear_mem;
                        WREN <= '1';
                        SCANRDY <= '0';
                  else
                        next_state <= idle;
                        WREN <= '0';
                        SCANRDY <= SCANRDY;
                  end if;
            when clear_mem =>
                  idat <= (OTHERS => '0');
                  WREN <= '1';
                  SCANRDY <= '0';
                  COUNT_EN <= '0';
                  if wraddr >= max_addr then
                        next_state <= scan_count1;
                        wraddr <= wraddr;
                  else
                        next_state <= clear_mem;
                        wraddr <= wraddr + 1;
                  end if;
            when scan_count1 =>
                  wraddr <= LTDR_CNT;
                  idat <= idat;
                  WREN <= '0';
                  COUNT_EN <= '1';
                  if COUNT_RDY = '1' then
                        SCANRDY <= '0';
                        next_state <= scan_count2;
                  elsif SCAN_DONE = '1' then
                        SCANRDY <= '1';
                        next_state <= idle;
                  else
                        SCANRDY <= '0';
                        next_state <= scan_count1;
                  end if;
            when scan_count2 =>
```

Figure 25: ν-OTDR Control State Machine in VHDL

```
                  wraddr <= LTDR_CNT;
                  idat <= idat;
                  WREN <= '0';
                  COUNT_EN <= '1';
                  next_state <= scan_count3;
            when scan_count3 =>
                  wraddr <= LTDR_CNT;
                  idat <= DATIN + 1;
                  WREN <= '1';
                  COUNT_EN <= '1';
                  if SCAN_DONE = '1' then
                        SCANRDY <= '1';
                        next_state <= idle;
                  else
                        SCANRDY <= '0';
                        next_state <= scan_count1;
                  end if;
            when Others =>
                  next_state <= idle;
                  wraddr <= (OTHERS => '0');
                  idat <= (OTHERS => '0');
                  WREN <= '0';
                  SCANRDY <= '0';
                  COUNT_EN <= '0';
      end case;
end if;
end process state_tr;
```

Figure 25: Continued

counter is called *LTDR_CNT*. Recall that a ν-OTDR must perform many scans to construct a signal whose amplitudes are representative of light intensity. A counter that keeps track of the scan count controls the signal *SCAN_DONE*. Assuming the system has more scans to do, the value of *LTDR_CNT* is loaded onto the address bus of the dual port RAM. Over the next two states the value of the memory location at this address is read, incremented by one and written back. If the *SCAN_DONE* input is high (indicating that the requested number of scans is complete), then the *SCANRDY* bit is set high and the ν-OTDR controller returns to the idle state.

As successive scans complete, the ν-OTDR waveform is built up as a histogram in the DPRAM. The x-axis of the histogram consists of time bins made up of 15-bit

memory addresses (0-32768). The y-axis represents the number of hits recorded at each time bin. Based on the sampling speed and the speed of light in the fiber, the time bins can ultimately be converted to spatial distance.

Referring back to Figure 25, while in the scan_count3 state, the next state assignment is based on the status of *SCAN_DONE*. *SCAN_DONE* goes high when a preset number of scans (measured by the number of output pulses) have been performed. More scans equate to an improved SNR in the waveform at the expense of time. If *SCAN_DONE* is high then the ν-OTDR waveform acquisition is complete and the *SCANRDY* signal is set high. The instrument is returned to event detection mode by first setting the 1x2 optical switch to pass light from the loop (see Figure 21). If, on the other hand, there are remaining scans to be taken, then the state machine is returned to the scan_count1 state. Once in the scan_count1 state the system is effectively re-armed and it will record the timestamp of the next photon hit. The important result of this is that the ν-OTDR can receive multiple hits for each output pulse sent.

As mentioned in Section 2.3.4, multi-hit gating can greatly improve the acquisition speed of a ν-OTDR waveform. Classic gating can be implemented in the form of an optical shutter, or more commonly by raising the bias voltage of the APD above breakdown only during the gating region [11]. In either implementation, the goal of multi-hit gating is to break up the signal into blocks so that multiple samples can be taken in multiple blocks. The gating implementation in the EDL is different in that it is based on the recharge period of the SPAD. Immediately after a pulse is detected the SPAD enters a recharging period. In this design, the minimum time before the next pulse can be detected is based solely on the SPAD recharge time. The advantage of this

approach is that no extra hardware is needed. The downside is that if the input to the SPAD is too strong, after-pulses may occur directly after recharge which will be registered as valid hits. The reason for this is an effect called "charge remanence". Light falling on the SPAD when it is not active (during recharge) can create trapped electron-hole pairs that will be released later with exponentially decaying probability [11]. The "charge remanence" effect can be minimized (although not completely eliminated) by adjusting the attenuation such that backscattered pulses are at a sufficiently low intensity. As long as the attenuation is carefully set, this method of multi-hit gating works well for this application and leads to improved signal acquisition speeds of about 150ms for 32,768 pulses over 3km.

It should now be clear that careful calibration is required to prevent SPAD after-pulses, but recall from Section 2.3.4 that calibration must also be setup correctly to ensure that the histogram amplitudes are relative to optical intensity. In a gated approach, the calibration must be set so that on average, much less than one reflection is received per output pulse per gating region. The problem with meeting this criterion in this design is that the gating region is not defined, but rather is based on the fiber under test, the output pulse power and the attenuator setting. It is therefore impossible to calibrate the EDL correctly so that histogram values relate to optical intensity. This isn't quite as much of a problem as one might think considering the purpose of the EDL: to detect and locate breaches in a fiber optic loop. To perform this task necessitates that the ν-OTDR provide clear indication of reflection location, but makes no requirements that calibrated intensity information be gathered. In practice, the increase in signal acquisition speed

and the simplicity of the hardware make SPAD recharge based multi-hit gating an acceptable approach.

5.4.2 v-OTDR High Speed Sampling Using Phase Offset PLL

Hardware simplicity was a primary goal of the EDL design. In sticking with this goal, a solution was needed in VHDL that would be capable of sampling the received ν-OTDR reflections at a rate of near 680 MHz. This sampling rate meets the goal of 1 foot spatial resolution as previously discussed.

The implemented solution is illustrated in Figure 26. The master clock is running at 80 MHz, and an algorithm was designed to increase the sampling rate to eight times the clock frequency. This results in an effective sampling rate of 640 MHz which yields a spatial resolution of slightly more than one foot. This sampling rate is achieved by using a single PLL with three clock outputs, all at 80 MHz but with different phase delays of 45, 90, and 135 degrees.

A Stratix "fast" PLL (the Altera Stratix device has two types of PLLs, "fast" and "enhanced") was used to generate the phase delayed clocks. The delay of each clock can be set programmatically via the Quartus II software. A desired phase shift is entered in degrees and the software automatically sets the phase taps and counter settings to achieve the closest delay possible. The delay accuracy is limited by the step resolution, which is equal to an eighth of the voltage controlled oscillator period [14].

An illustration of the PLL based sampler is shown in Figure 26. The master clock and the three phase delayed clocks provide four clock edges during the first half of the master clock cycle. By inverting those four signals, the next four clock edges are

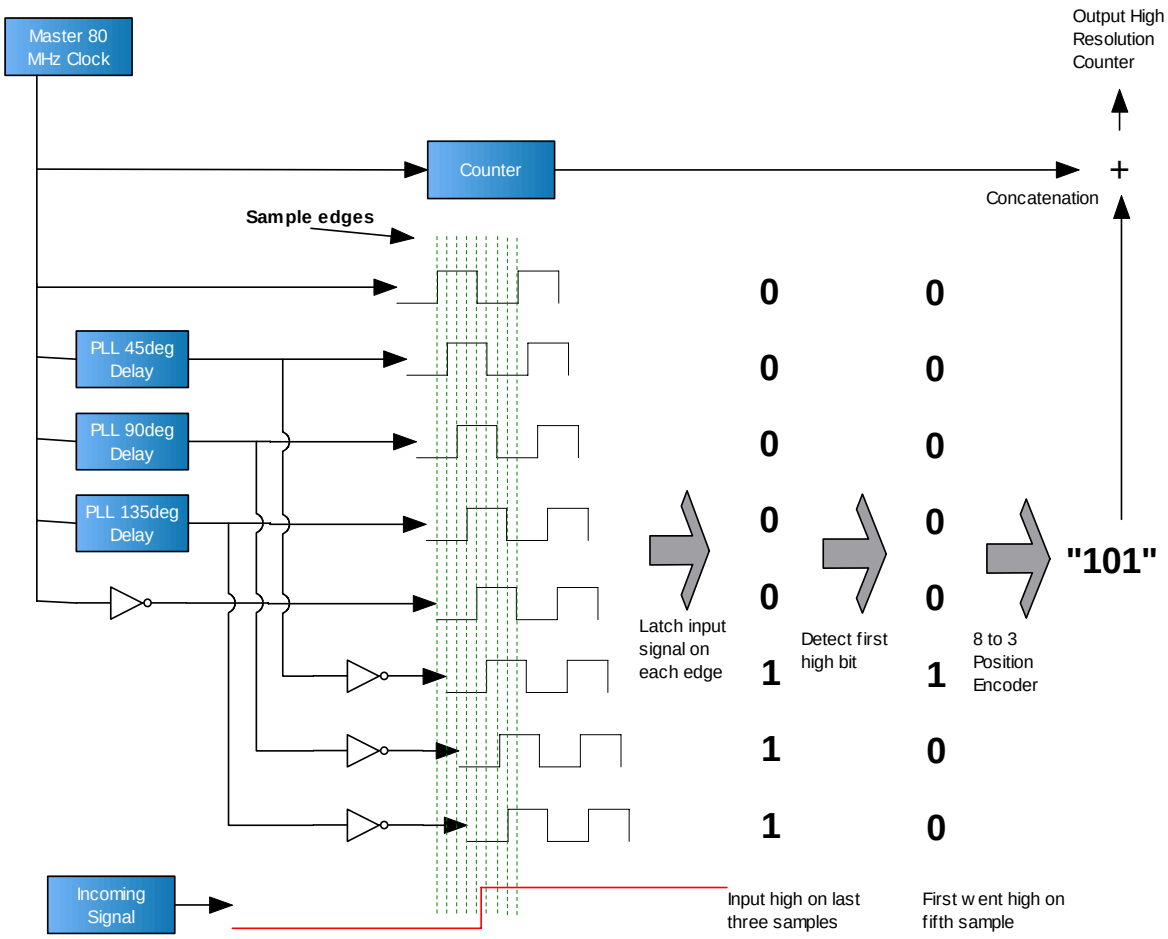

Figure 26: High Speed Sampling in VHDL with Phase Offset PLL

generated. Each one of the sample edges in Figure 25 drive a D flip-flop which latches the incoming signal into a buffer. The output of each flip-flop is concatenated into an 8-bit word. For the signal shown in red in Figure 26, this 8-bit word is “00000111”. This means that the input signal was high when sampled by the last three flip-flops.

The 8-bit word is then run through the high-bit detector shown in Figure 27. This detector keeps the first high bit, starting from the least significant bit in the word, and sets all other bits to zero. From there, the 8-bit word is encoded into a 3-bit binary coded decimal value based on which bit in the word is set (i.e. bit 0 = “000”, bit 5 = “101”). This 3-bit word is now concatenated with a counter running off the 80 MHz clock. By concatenating the three bits as the LSB of the counter value, the sampling resolution is increased from 80 MHz to 640 MHz.

Additional logic is in place to notify the control module that a photon hit has been recorded. The counter value is then used as a memory address by the control module to build a histogram as previously discussed. The implementation of the phase delay based sampler was done in Altera Quartus II block diagram format. It is shown in Figure 28.

5.4.3 ν-OTDR Waveform Transfer

With the ν-OTDR waveform acquired and saved to dual port RAM, the next step is to transfer the data from the FPGA to the RCM3360 core module. This transfer is based on the Serial Peripheral Interface (SPI). SPI is a serial bus standard established by Motorola but is supported by several vendors. It is a synchronous protocol and can operate in full duplex mode [27].

Under the SPI protocol, devices communicate using a master/slave relationship. The advantage of using SPI is the simple implementation of an SPI slave on the FPGA.

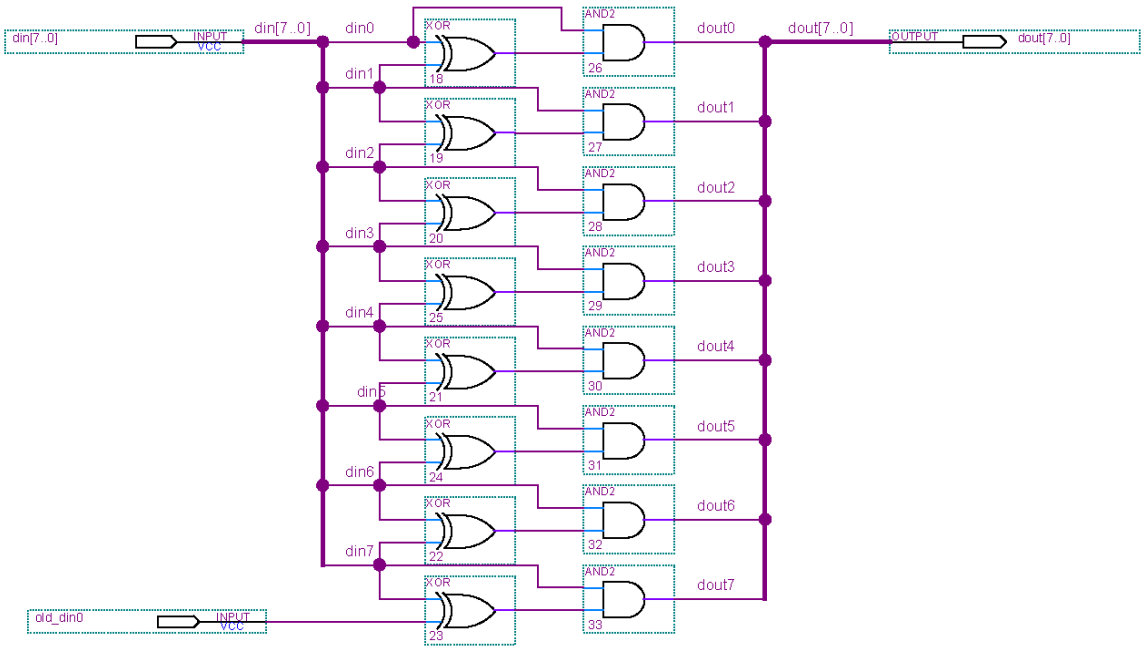

Figure 27: Altera Block Diagram of High Bit Detector

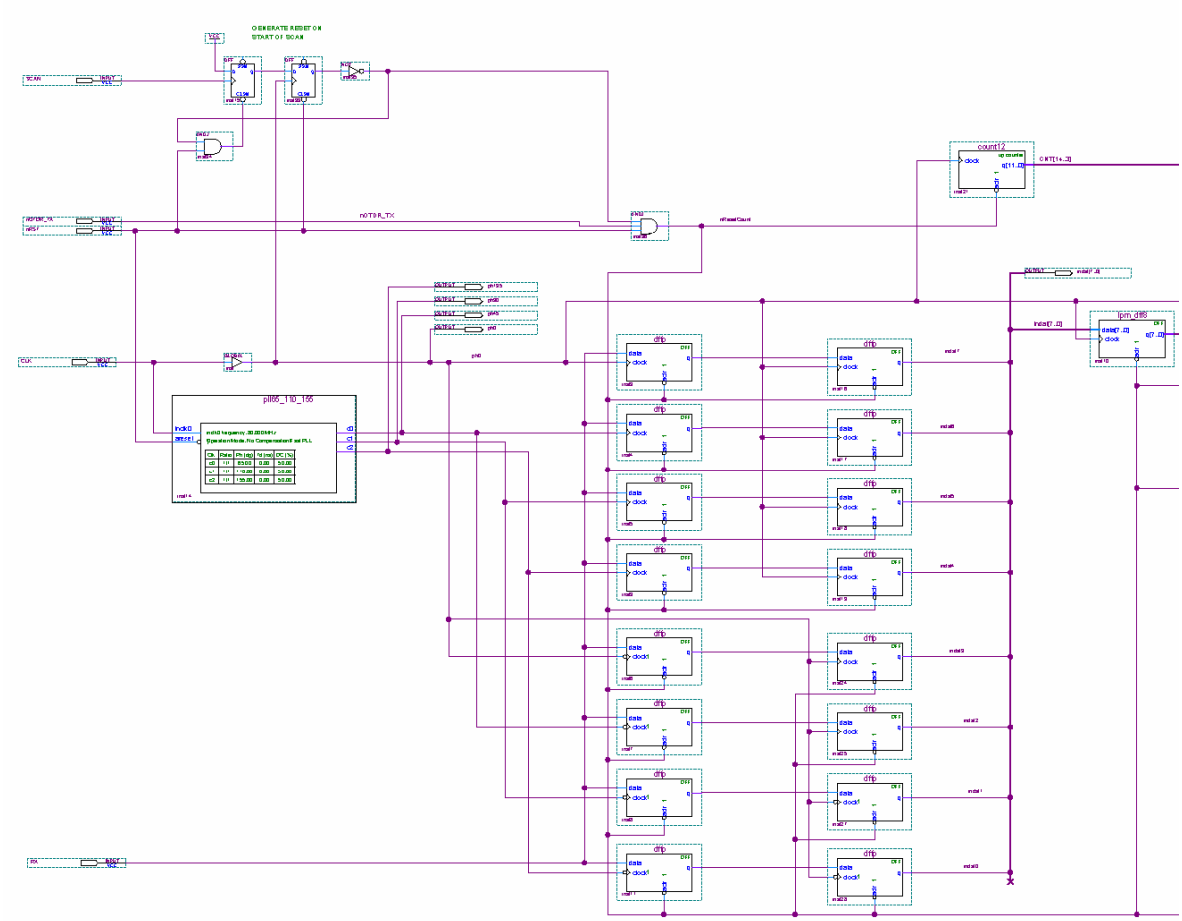

Figure 28: Phase Delay Based High Speed Sampler

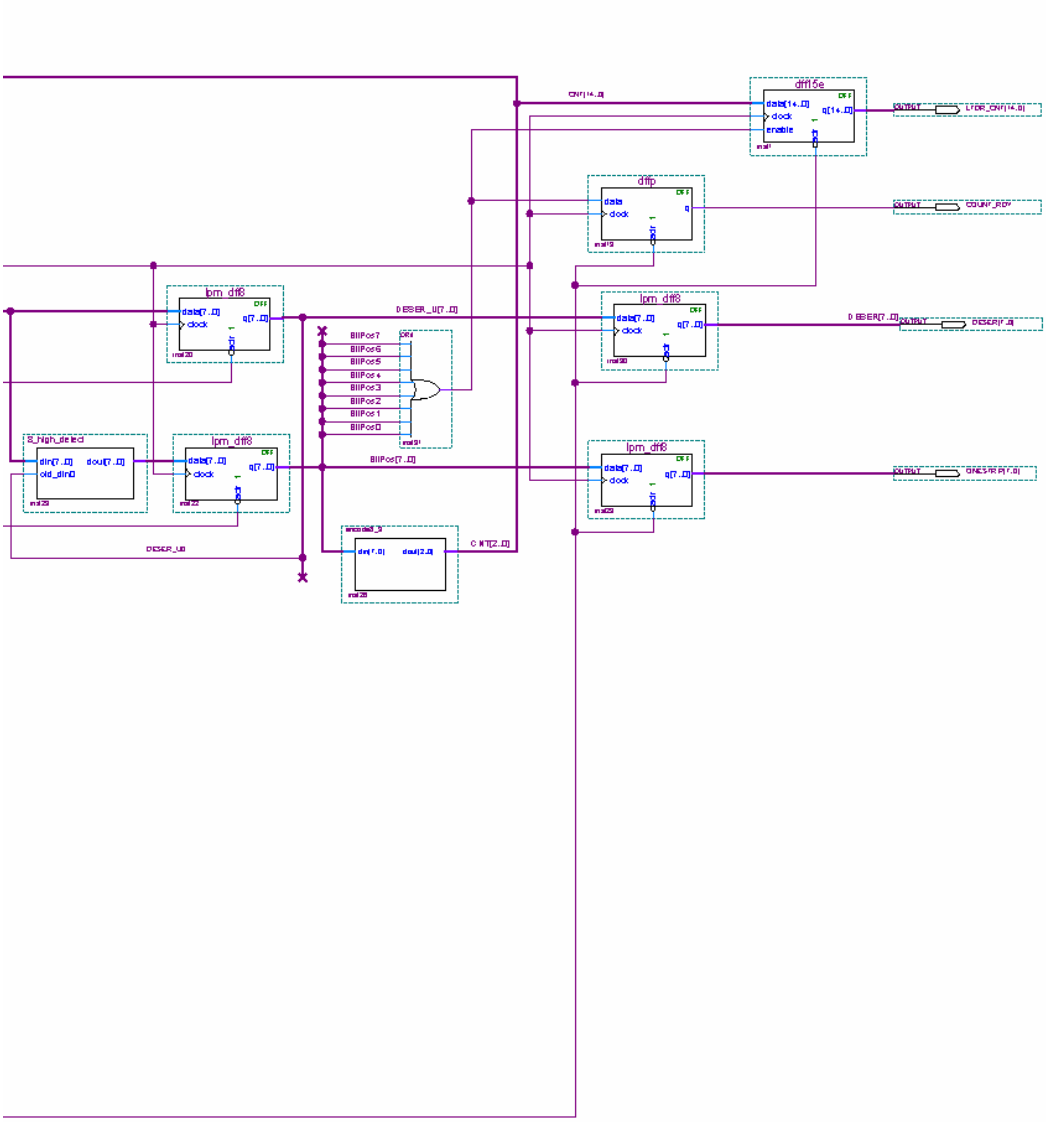

Figure 28: Continued

There is also code available in libraries supplied with the RCM3360 to implement the SPI master.

The SPI bus consists of four signals, clock (SCLK), master data output/slave data input (MOSI), master data input/slave data output (MISO) and a slave select line (SS). In this implementation there is only one slave, so the SS line is not used. Additionally, the slave does not receive data from the master, so the MOSI line is not used (i.e. the FPGA does not receive data from the RCM3360 over the SPI bus). The FPGA implementation is therefore limited to synchronous data transmission.

The SPI MISO signal is generated by a 16-bit shift register created with the Altera library of parameterized modules (LPMs). Based on control signals from the RCM3360, data is loaded into the shift register from the FPGA's DPRAM and each bit is synchronously shifted out on the MISO line.

5.5 Communication and System Control in Dynamic C

The communication and higher level system control for the EDL are handled in firmware executing on the Rabbit RCM3360 core module. This firmware's primary responsibility is to provide a middleware layer between the FPGA and the rest of the world. The firmware's secondary priority is to provide local control of the EDL including controlling the state of the optical switches, controlling the variable optical attenuator, setting or disabling the local alarm and controlling the local liquid crystal display (LCD).

The flowchart illustrating the RCM3360 firmware is shown in Figure 29. Starting from system initialization and characterization, the default mode of operation is breach detection. The fiber loop is tested for integrity based on an output from the FPGA. If the loop is not secure, the local visual alarm is turned on and a *breached* variable is set. A check is then made to see if a ν-OTDR scan was requested. If a scan is requested, the remaining steps setup the components and communicate with the FPGA to download the ν-OTDR waveform. If a ν-OTDR scan was not requested, a check is made to see if the user has acknowledged previous breaches. If so, the alarm is deactivated and the loop repeats.

For simplicity sake, the flowchart is shown as if the firmware operates in a single threaded, line by line model. In reality the firmware is multithreaded and uses cooperative multitasking to perform several functions in parallel. For example, the seal status check, ν-OTDR scan request (in the form of the "getwaveform" request), and LCD control are actually occurring in parallel.

5.5.2 Network Communication

The EDL communication layer is meant to provide a means for the device to be remotely monitored, configured and controlled over an existing local area network (LAN) or the Internet. Additionally, security features were desired to ensure that data from the EDL was authentic and that sensitive EDL functions could not be accessed by unauthorized users. To meet these goals and provide the added benefit of cross-platform compatibility, HTTPS was chosen as the data protocol. Every computer produced today comes with a browser compatible with HTTPS, and most current software languages provide an easy way to request a document using HTTPS.

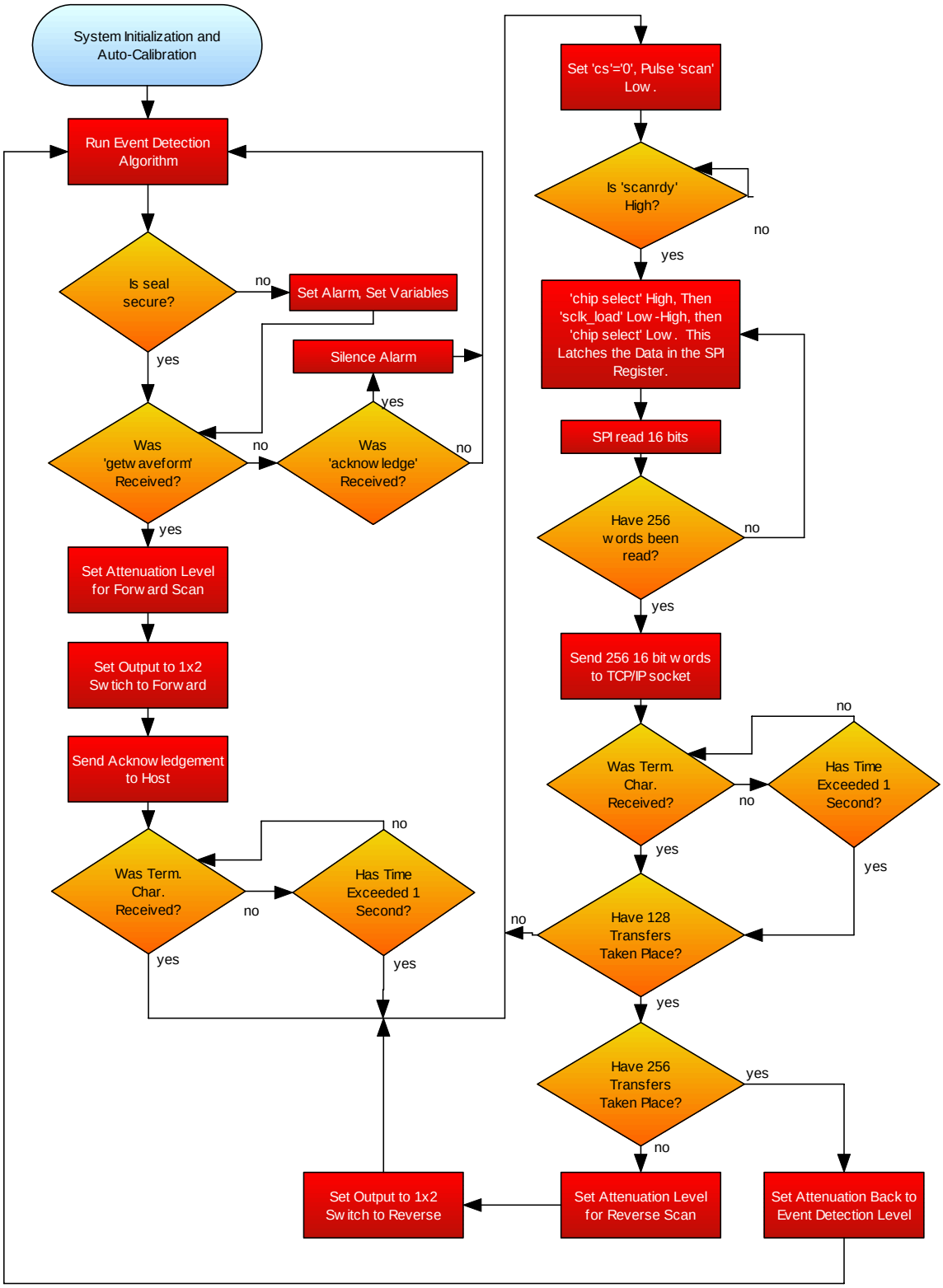

Figure 29: RCM3360 Firmware Flowchart

In order to meet the HTTPS specifications, the RCM3360 is running as a simple web server capable of executing common gateway interface (CGI) scripts, serving images, serving hyper-text markup language (HTML) documents and serving eXtensible markup language (XML) documents. HTML pages are served when the EDL is accessed with a browser. A screenshot of the main HTML page is shown in Figure 30. From the main menu the user can request access to administration tools, get the system status, acknowledge a breach, get device info or get help.

The "get status", "acknowledge breach" and "get device info" functions from the main menu execute CGI scripts and return XML documents. An XML document is similar to an HTML document with the main difference being that the tags in an XML document are defined by the user. The result is an open file with an easy to understand data structure that can be parsed by a wide variety of tools, browsers and programming languages.

The administrator tools page is a password protected page that lets the user change the IP address, digital signature key, administrator password and time and date. A screen shot of the administrator tools page is shown in Figure 31.

While scripts can be executed by clicking the links from a browser, the intention was that the scripts be executed by a GUI running on a remote host which could parse the returned XML document. The scripts available on the EDL are shown in Table 2.

A typical XML document returned by the getstatus.cgi script is shown in Figure 32. The XML tags have been defined so that each parameter can be easily understood. In normal operations, a host program would parse this file and display and process the data appropriately.

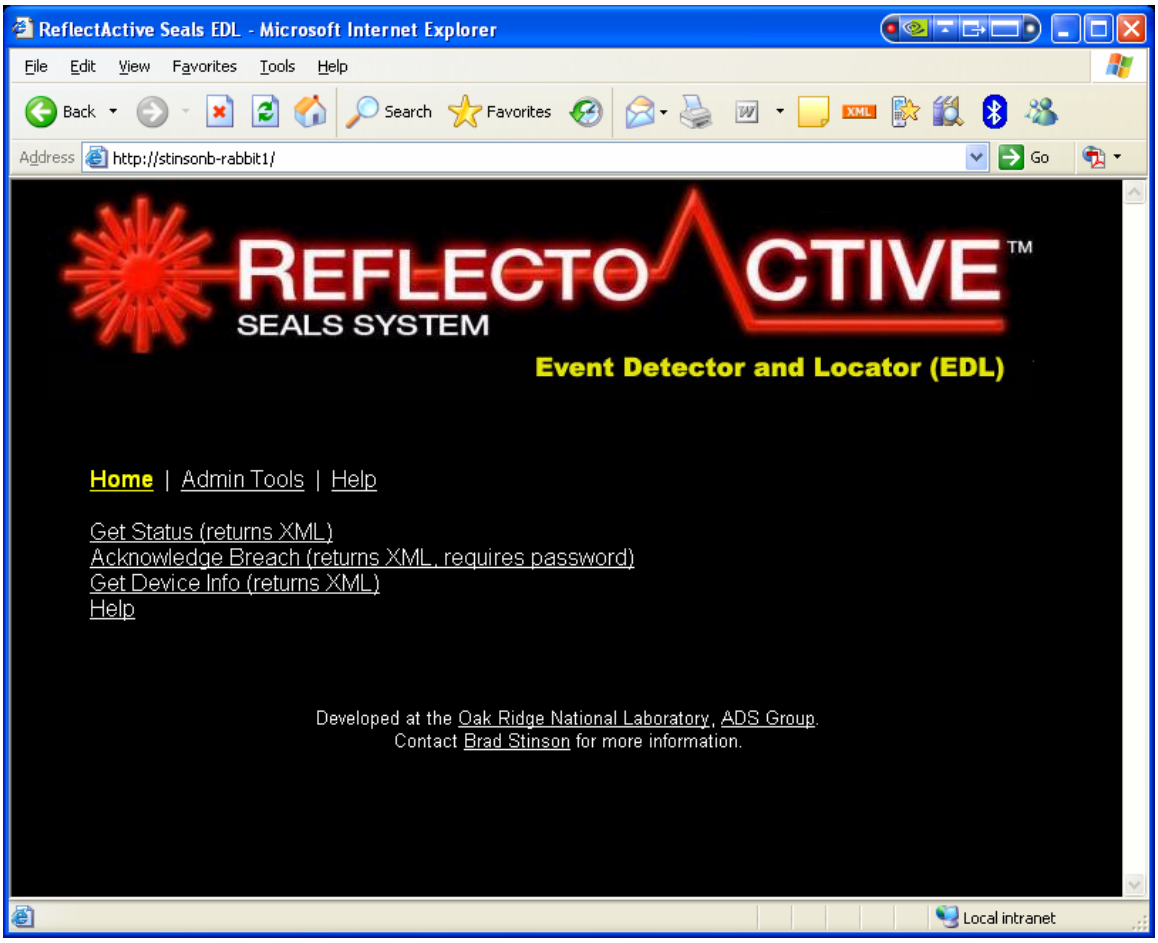

Figure 30: EDL Main Page in HTML

Figure 31: EDL Administrator Tools HTML Page

Table 2: CGI Scripts Available on the EDL

Script Name	Script Description
getstatus.cgi	This script causes the RCM3360 to sample the appropriate inputs and memory values and return the data in an XML document.
acknowledge.cgi	This password protected script acknowledges a breach. The local alarm is turned off if it is currently on. A basic empty XML document is returned to verify that the script execution was successful.
deviceinfo.cgi	This script returns information about the EDL in an XML document including Media Access Control (MAC) address, IP address, firmware versions and attenuation level.
recalibrate.cgi	This password protected script recalibrates the event detector instrument by adjusting the variable optical attenuator. This function is useful when a connected loop undergoes a significant change like the addition or removal of multiple seals. A basic XML document is returned to verify that the script execution was successful.

```
<?xml version="1.0" encoding="UTF-8" ?>
<RASeals>
     <LinkStatus>Secure</LinkStatus>
     <AckStatus>Acknowledged</AckStatus>
     <TamperStatus>Secure</TamperStatus>
     <AlarmStatus>Off</AlarmStatus>
     <DateTime>Thu, 26 Jan 2006 13:00:34 GMT</DateTime>
     <Signature>7c13fe9826f904b3baeeb51d26d738c1</Signature>
</RASeals>
```

Figure 32: GetStatus.cgi Returned XML Document

Security and data integrity for a system of this nature are very important. Two software features are in place to ensure that the data is valid and that only authorized users have access to sensitive functions. First, the status message from the EDL is digitally signed with an MD5 hash of a concatenation of the message data and a private key known only to the EDL and the host software. MD5 is a one way hashing algorithm that is primarily used to create digital signatures for data packets [28]. The host is responsible for generating and verifying the same hash based on message data and the private key. The signature prevents message spoofing. Secondly, all sensitive scripts on the EDL are password protected. HTTPS not only encrypts all transmissions, but provides a Digest authentication mode which handles encrypting passwords automatically [29]. The Digest mode is used because passwords are not sent in plain text (unlike Basic mode). This prevents unauthorized sources from being able acknowledge breaches or recalibrate the EDL.

5.5.1 Communication with FPGA

To determine loop integrity status, the RCM3360 firmware simply reads the status of the FPGA output *link_status.* When a breach occurs and the RCM3360 receives the "getwaveform" command from a host, the RCM3360 initiates a ν-OTDR scan. Referring to Figure 29, the first thing to happen is the attenuator level is set. Recall that event detection and event location each require a different attenuation level. The specific attenuation level to use for event location is specified by the host. Next, the 1x2 optical switch is set to the forward direction. An acknowledgement message is sent back to the host which includes the successfully set attenuation value.

The defined communication protocol requires that the host send a termination character after each transmission. The EDL therefore enters a wait state until this termination character is received. Once received, the ν-OTDR waveform acquisition begins by first pulsing *scan* low. Based on Section 5.4.2, the *scan* input starts the FPGA waveform acquisition and the *scanrdy* signal is output when completed. The RCM3360 reads the *scanrdy* signal and begins the waveform download using the SPI protocol via the *chip_select* and *sclk_load* signals.

The XML-based communication approach described in the previous section works well for small data packets but is very inefficient for large amounts of data like the ν-OTDR waveform. The ν-OTDR waveform consists of 32,768 16-bit words for a total of 65KB of data. XML is an ASCII based document, which means that a 16-bit value could take up to 5 bytes to be represented. As an example, consider the integer number 65535. After being encoded into ASCII, this number would be represented as "00110110 00110101 00110101 00110011 00110101" versus the binary representation of "11111111 11111111" [30]. This means that to send the same data in XML would take up to 164KB for the data plus the XML tags which would put the total size over 300KB. This method is wasteful and not practical considering the RCM3360 memory structure. The alternative is to open a socket connection to the EDL and use the TCP/IP to transmit binary data.

A socket connection is established from the host computer to the EDL on port 21. The data is read in blocks of 256 16-bit words. As before, the host is responsible for generating a termination character after each block is received. 128 total blocks are received. Once the transfer of the forward scan is complete, the RCM3360 firmware sets

the 2x2 optical switch to the reverse direction. The attenuator is reset if necessary and the waveform acquisition and download process is begun for the reverse direction. The socket is maintained until both forward and reverse waveforms are transmitted. Once the waveform is downloaded, the socket is closed by the host.

5.5.3 Optical Switch Control

Digital output lines from the RCM3360 control the 1x2 and 2x2 optical switches. The outputs are switched appropriately based on commands received from the host that start a ν-OTDR scan and dictate the scan direction. As previously discussed, the 1x2 switch controls the EDL mode (event or location detection) and the 2x2 switch controls the ν-OTDR scanning direction.

5.5.4 Attenuator Control and Calibration

Careful calibration is crucial to the successful operation of both the event detector and the event locator. The firmware on the RCM3360 has the task of controlling the variable optical attenuator. Attenuator adjustment happens during power up, before and after a ν-OTDR scan (see Figure 29), and when requested by an authorized host.

At startup, the firmware must determine the attenuation level that is best matched to the fiber under test based on the output of the FPGA (which depends on the output of the SPAD). Remember that over-driving the SPAD causes signal degradation. Under-driving the SPAD leads to undetected hits. There is an optimum setting that must be determined by the RCM3360 firmware.

The calibration is performed by starting at an attenuation value of 0dB and stepping upwards toward a maximum of 30dB with a one second pause at each step.

During that second, the FPGA *link_status* bit is sampled several times. Once an attenuation value is reached at which the FPGA *link_status* bit indicates a secure seal, this value is marked as the lower limit. The process is now repeated, but this time starting at the maximum attenuation value of 30dB and stepping down toward 0dB. When the upper limit is found, the attenuation is set at the midpoint of the upper and lower bounds as shown in Equation 6.

$$attenuation = \frac{upper\ bound + lower\ bound}{2} \qquad \text{Equation 6}$$

When a ν-OTDR scan is requested, the forward and reverse attenuation values are sent from the host as part of the request. The firmware sets the attenuator to the requested value and inserts a small delay to give the attenuator time to respond before continuing to acquire the ν-OTDR waveform. Once the waveform acquisition is complete, the attenuation is set back to its previous level.

5.5.5 Alarm Interface

The firmware on the RCM3360 sets the output to the latching relays that control the EDL visual alarm. When a breach is detected, an output pulse is generated which latches the visual alarm relay on. When an "acknowledge" command is sent to the RCM3360 by the host, an output is generated that resets the latching relay and turns off the alarm.

5.5.6 Tamper Input

The EDL contains a tamper indicating circuit that is connected to a digital input on the RCM3360. The tamper circuit generates a logic high input when the enclosure is

opened during normal operation. A tamper indication results in notification to the host, activation of the local alarm and an LCD update indicating a tamper condition.

5.5.7 Display Control

The Matrix Orbital LCD is controlled by the RCM3360 firmware via the I^2C bus. Rabbit Semiconductor provides an I^2C library with Dynamic C, which led to a quick implementation of the display controller. The display serves the purpose of providing an indication of loop continuity status, tamper status and current acknowledgement state. There are five possible operating screens as shown in Table 3.

5.6 Windows Graphical User Interface in Microsoft C#

An application was developed in Microsoft C# to act as a host for the EDL. The software provides a GUI for displaying data, provides a means to test the event location algorithm and illustrates event logging and the general application of a ReflectoActive™ Seals system.

5.6.1 EDL Status Display and Control

The RAS GUI displays the EDL status by using a timer to poll the getstatus.cgi script periodically. The returned XML document is parsed and displayed on the screen as shown in Figure 33. The GUI continually monitors the status data to detect and log status changes. For example, if the link status changed from “Secure” to “Breached” then a “Seal Opened” event would be logged.

The menu of the main screen provides easy access to EDL functions. Under the “System” menu the user can connect/disconnect to an EDL and initiate a recalibration.

Table 3: LCD Operating Screens

Screen Name	**Screen Description**
Secure	The loop is not breached and all previous breaches have been successfully acknowledged.
Secure Unacknowledged	The loop is not breached but previous breaches have not been acknowledged.
System Breach	The loop is currently breached.
Tamper	The enclosure has been opened during normal operations without authorization.
Initialization	During startup the LCD provides visual indication of each subsystem's status. For example, if the calibration of the attenuator was not successful (a secure link could not be obtained at any attenuation value) then the display would report "Optics… Failure" during startup. Likewise if the Ethernet adaptor init failed this display would report "Ethernet… Failure."

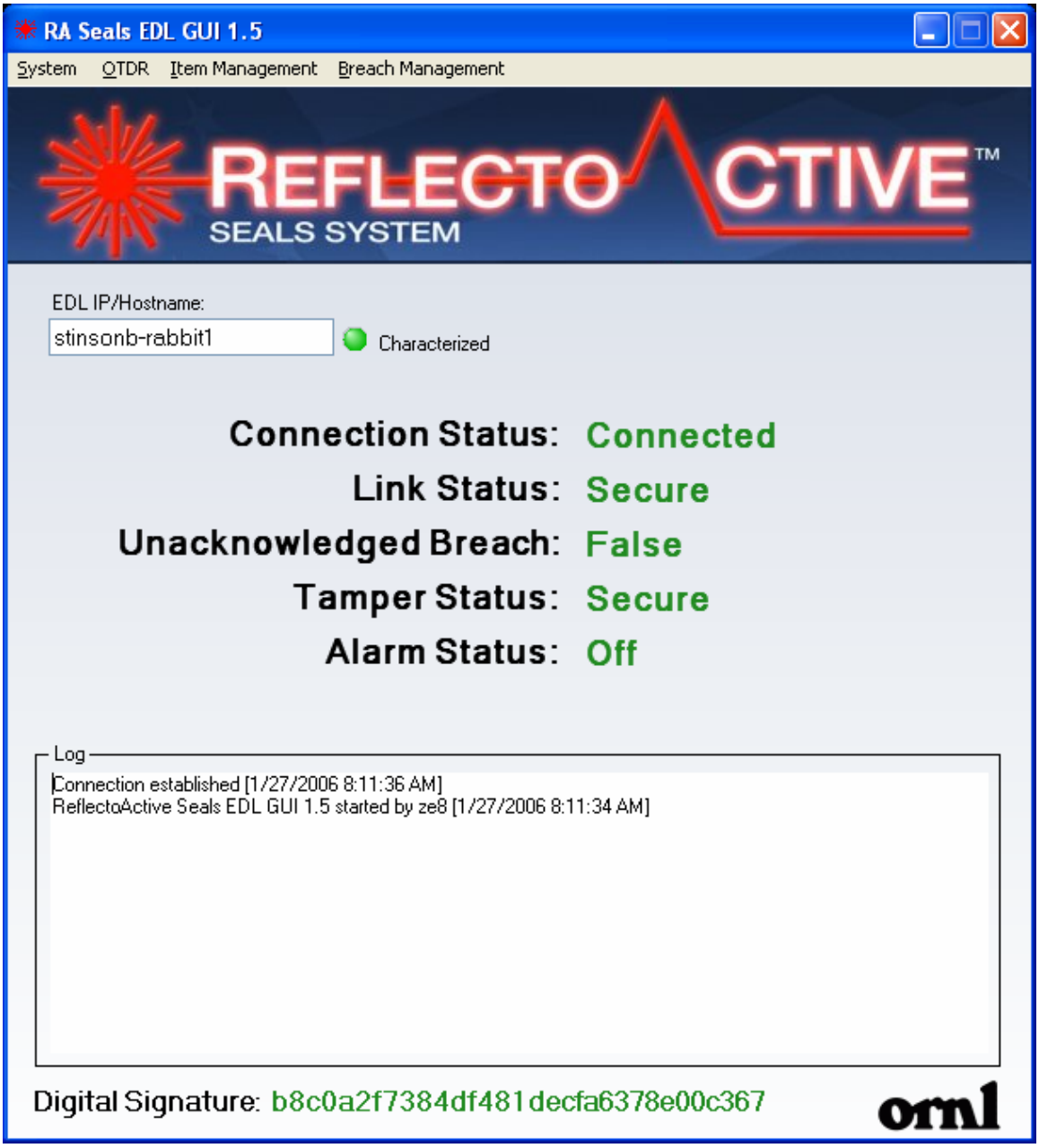

Figure 33: RAS GUI Main Screen

Under the “Breach Management” menu the user can acknowledge a breach. Under the “Item Management” menu the user can add or remove an item and provide seal pseudonyms (i.e. change the name “Seal 1” to “Container 1”).

5.6.2 Event Logging

The RAS GUI logs events to the local screen as well as the Windows event log. The Windows event log is a low level, well protected module within the operating system. As such, it is an ideal place to store sensitive logs that cannot be lost.

5.6.3 Item Addition and Removal

The “Add Item” and “Remove Item” functions under the “Item Management” menu put the GUI into a state where the next seal breach is expected and automatically acknowledged. The ν-OTDR waveform is analyzed to verify that the expected seal opening is indeed the opened seal and that no other seals are opened. After the item addition or removal is complete, user authentication is required to confirm the change.

5.6.4 Illustration of Maintaining User Accountability

The host is responsible for maintaining user accountability for all RAS actions. The RAS GUI was developed to illustrate one method of achieving this. The GUI tracks users by requiring a username and password before sensitive events including startup, shutdown, breach acknowledgement, recalibration, item addition and item removal. The user is authenticated against a Windows domain if available; otherwise the user is authenticated against the user account on the individual computer. When requesting a sensitive action, the user is prompted with the dialog box show in Figure 34.

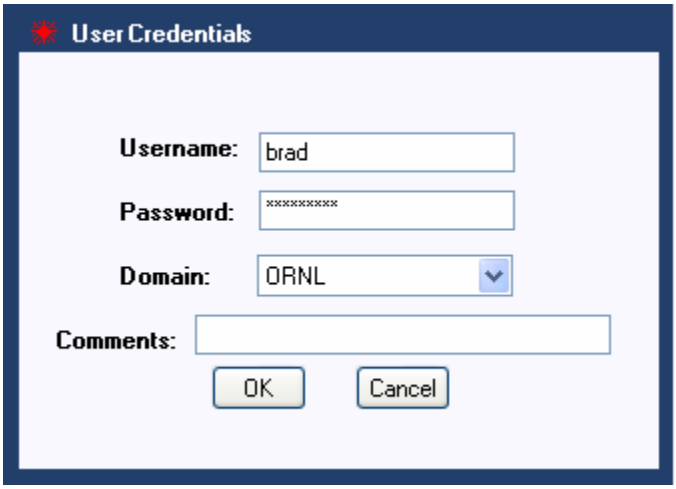

Figure 34: RAS GUI User Authentication

5.6.5 ν-OTDR Waveform Display and Characterization

Once the host downloads the ν-OTDR waveform, it is useful to display the waveform to the user and provide a method for marking the location of known seals. Marking the seals is required so that location can be found when a future breach occurs. This process is called characterizing the loop. A National Instruments waveform control was used in Microsoft C# to provide waveform graphing, panning, zoom and cursor control. The ν-OTDR dialog screen is shown in Figure 35.

The waveform displayed in Figure 35 shows a small loop with 8 seals. Each seal is represented by a sharp peak, indicating a strong Fresnel reflection at that point in the fiber. The y-axis represents the number of hits recorded; the x-axis is distance in meters. The controls underneath the waveform are used for characterization and to adjust the attenuation value and scanning direction.

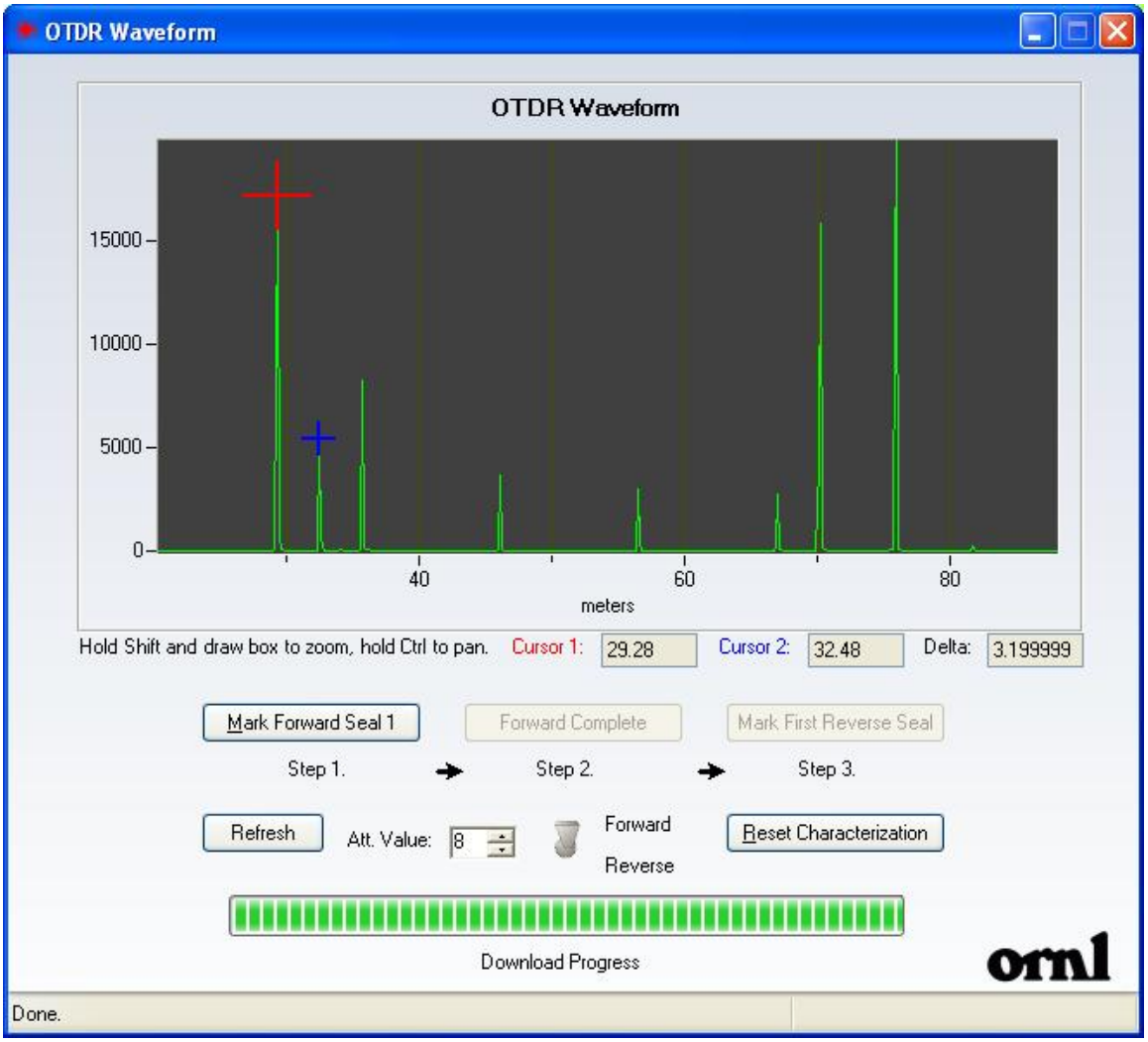

Figure 35: RAS GUI v-OTDR Screen

The red and blue cursors are provided to measure distances between peaks and to mark the location of known seals. In order to characterize the system the user must first adjust the attenuation levels so that the reflections are strong, but after-pulses and ghosts are minimized.

To characterize a loop, the attenuation is set appropriately and a forward scan is downloaded. The red cursor is used to snap to the first peak and the "Mark Forward Seal 1" button is pressed. The red cursor can now be moved to the next seal location and the "Mark Forward Seal 2" button is pressed (the text of the button will update after each seal is marked). Once all seals have been marked the "Forward Complete" button is pressed. A reverse scan is now automatically acquired. The user can adjust attenuation and rescan if necessary. The user must then mark the location of the first reverse seal by using the red cursor to snap to the first peak and then pressing the "Mark First Reverse Seal" button. Based on the spacing of the forward seals, the reverse seals are marked automatically. A message box will appear indicating success or failure. If successful, an XML characterization file is written that saves the seal locations, peak values and attenuation levels.

5.6.6 Breach Location Algorithm

When a seal breach occurs, the host controller (in this case the GUI) can determine the location of the breach by requesting a ν-OTDR scan and then comparing the acquired signal with the previously saved characterization data. The algorithm works by comparing signal amplitudes (number of hits) at each of the expected seal locations. An open seal is detected by a lack of a peak at the next expected seal location.

In the waveform acquired in Figure 36, the last two seals are expected to be located at 22.08 and 32.32 meters. Since no peak is present at 32.32 meters, the algorithm would conclude that the seal at 22.08 meters is opened. This method works most of the time, but there are two observed cases when it does not.

The first case is when the reflection amplitude of a seal drops substantially from when it was characterized. This can happen because of fiber alignment varying in the coupler as a result of openings and closings, or can be caused by contaminated connectors. In either case, the result is that the peak detection algorithm will see the sharp value drop-off as an indication that the seal is no longer present. To avoid this, the algorithm considers the next two seal locations. If peaks are found at those locations then it is safe to assume that the amplitude drop-off is due to a factor other than a seal breach.

The second case occurs in systems with repeated fiber lengths (i.e. the distance between seals is always the same). In this case compounded ghosting can occur, causing false signals to appear at expected seal locations even if the seal is not present. Usually the fake signals are of much lower amplitude than the characterization signals, so tolerances in the peak detector can filter these out.

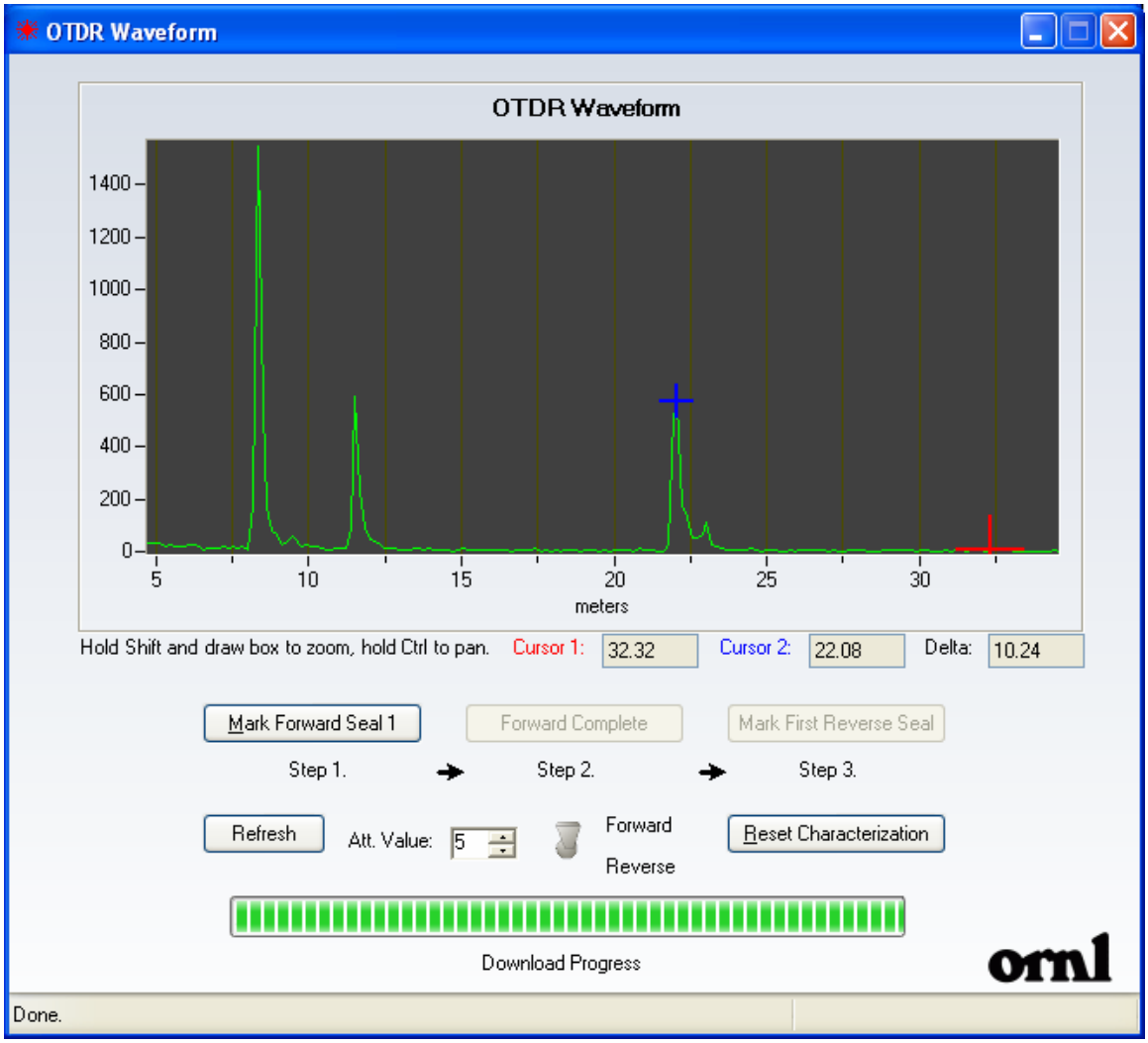

Figure 36: v-OTDR Waveform Illustrating Breach Location

6.0 Design Results and Performance

6.1 Introduction

This chapter will discuss the design results and how well the device meets performance criteria. Optical performance expectations are that each EDL can monitor 100 or more items. Operational specifications are that the instrument must support common network architectures and be well suited for a large facility, multi-room deployment. Additionally, the instrument is expected to be reliable, robust and tamper proof.

6.2 Event Detector Performance

The metric associated with event detector performance is the number of seals per loop that the EDL can monitor while staying far enough away from the noise floor that false alarms are not generated. This number of seals is difficult to quantify because all seals are not created equal. Testing of a few dozen ST-ST couplers revealed that the attenuation due to coupling varies from 0.01dB to 0.4dB. It is assumed that the variance is due to manufacturing tolerances leading to slight fiber misalignment in the coupler. To model a typical system, it was assumed that each connector will cause a loss of 0.2dB. A variable attenuator was used to model multiple connectors. The attenuator value was increased until the EDL indicated a seal breach. At this point the EDL was recalibrated to determine if it could compensate for the loss of light. The process was continued until the EDL could no longer compensate for the modeled seals.

The tests revealed that the EDL has a dynamic range of 27 dB. Assuming 0.2dB per connector, this equates to a maximum seal count of 135. In practice, the number of seals used per link should remain well under this threshold (e.g. 100 seals).

6.3 Event Locator (ν-OTDR) Performance

The metrics associated with the event locator are spatial resolution, event dead zone and breach location performance. Spatial resolution is easily measured, but the next two metrics are based on several factors and cannot be easily quantified.

Spatial resolution is limited in this design by the sampling rate of the FPGA. The theoretical resolution is about 0.46 meters (1.5 feet) based on a sampling rate of 640MHz. To measure the spatial resolution empirically, a test was conducted on a 10 meter multimode fiber optic cable connected to a 3.2 meter multimode fiber optic cable using an ST-ST coupler. This test cable was connected to the EDL. ν-OTDR scans were taken and the distance between the observed Fresnel reflections was calculated via the cursors in the RAS GUI. The 3.2 meter cable was then cut-back 0.1 meters at a time with ν-OTDR scans taken after each shortening. The peaks remained well separated until the 3.2 meter cable was cut back to a length of 0.48 meters (the scan at 0.48 meters is shown in Figure 37). At this length, the two peaks are beginning to converge but they remain well-resolved enough to determine the peak centers. Cuts past this point result in a waveform with difficult to distinguish individual peaks. The 0.48 meter length is equivalent to 1.57 ft, so the measured resolution is consistent with the theoretical limit.

Recall that a classic ν-OTDR has no dead zone. However, the implemented ν-OTDR is not a classic design due to the SPAD recharge based multi-hit method. For a ν-

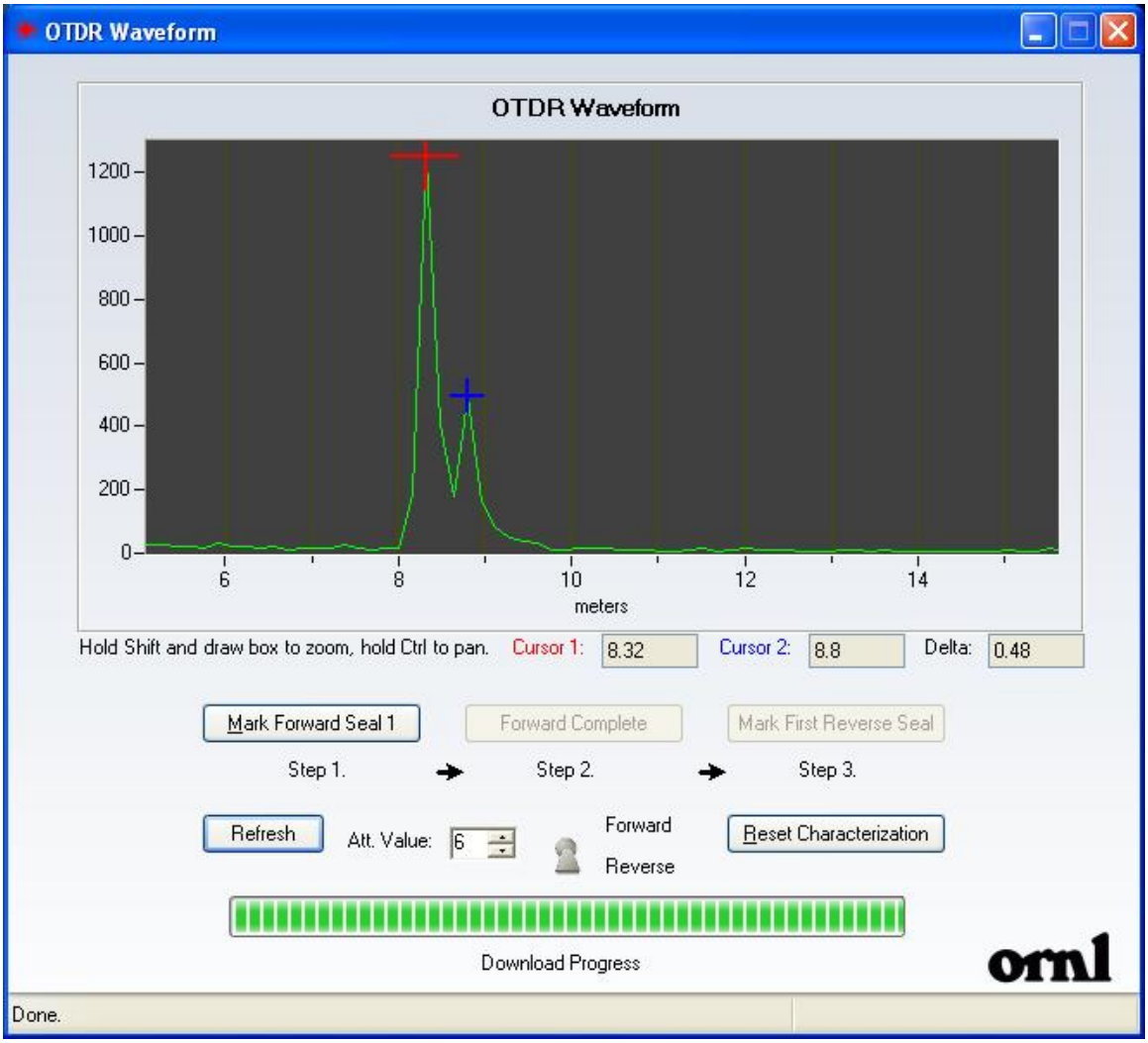

Figure 37: ν-OTDR Spatial Resolution Limit Test

OTDR to have no dead zone it is required that the system be calibrated such that on average much less than one photon is reflected for each output pulse sent. The EDL is designed to receive multiple photons per output pulse yet has no predefined knowledge of gating regions. This means that virtual event dead zones can exist. If a strong reflection event is not properly attenuated such that a reflection is generated at that event for each output pulse, then the recharge time of the SPAD creates a blind spot that never gets analyzed. Proper attenuation can usually eliminate this problem.

The performance of the breach locator can be broken further into three categories; time to resolve a breach, location accuracy and measurement repeatability. The first is easily measured while the second two are difficult to quantify because they depend on the quality of the acquired waveform and the user controlled characterization process.

Breach measurement time is represented in Equation 7. τ_{WA} is the duration spent acquiring the waveform into FPGA memory. A typical value for τ_{WA} is 300ms. τ_{WTFF} is the duration required to transfer the waveform from the FPGA to the RCM3360. Using the SPI protocol, τ_{WTFF} is about 5 seconds. τ_{WPR} is the duration required for the RCM3360 to process the data into the proper format. Based on data conversions in the RCM3360, τ_{WPR} is about 400ms. τ_{WTFR} is the duration required to transfer the waveform from the RCM3360 to the host computer. τ_{WTFR} is highly variable due to network variances, but a typical 100 Mbps Ethernet network with no extraneous traffic would transfer the waveform in 10ms. τ_{WP} is the time required to process the waveform on the host. τ_{WP} is also highly variable based on processor speed and memory configuration.

Typical values are on the order of 200ms. Based on these values, τ_{BM} typically equals about 6.0 seconds.

$$\tau_{BM} = \tau_{WA} + \tau_{WTFF} + \tau_{WPR} + \tau_{WTFR} + \tau_{WP} . \qquad \text{Equation 7}$$

Breach location measurement accuracy depends on the success of the host algorithm. The inputs to this algorithm include the OTDR waveform and the previously defined characterization data. It is impossible to present absolute results, but in testing with 14 seals separated by 15m, 100% location accuracy was obtained for 20 successive measurements. If the system was improperly characterized or the waveform signal is severely degraded, the location accuracy will suffer.

Breach location measurement repeatability also depends of the success of the host algorithm. The measurement repeatability is dependent primarily on changes incurred during repeated waveform acquisitions, but could be negatively impacted by a poor characterization. A test was conducted in which the fiber was opened and subsequently not handled. 20 repetitions led to 100% measurement repeatability. It should be noted that handling the fiber (i.e. contaminating the open end or altering the fiber in another way) could change this result.

As mentioned, a maximum of 14 seals were used for these test. However, fiber loops with more connectors were connected for a short time. It was observed that the reflection strength of events degrade as more connectors are added. Once the connector count exceeds 20, it is usually very difficult to pick out a few of the seals from the noise floor. Methods for future modifications to overcome this problem will be discussed in Section 7.2.

7.0 Conclusions

7.1 Contributions

This paper has demonstrated the successful design of a fiber optic loop continuity detector with integrated multi-hit photon counting OTDR. It was shown how low level ν-OTDR functions including memory and sampling can be implemented in a single FPGA. Firmware was developed on an 8-bit microprocessor core module that provides system control functions and Ethernet based communication. Support electronics were designed and built to interface various optics and electronics components. Host software was developed to illustrate breach detection, breach management and the implementation of location algorithms. The end product is the first known device that can solely detect fiber optic loop discontinuity and resolve the location of the discontinuity in both the forward and reverse directions.

The design was a success and marks a major milestone in the development of the ReflectoActive™ Seals system. Reliance on third party OTDR vendors has been removed, which helps ensure instrument availability and reduces cost. The developed ν-OTDR provides customizable options and a level of control never before available in an RAS system. The tight integration of the event detector and the event locator result in a smaller, cheaper package. The simplified and robust communication scheme eases host and GUI development and ensures compatibility with a wide range of computer equipment. The networked architecture will help meet future installation demands across multiple rooms, facilities and sites.

7.2 Future Work and Research

The design is not without room for improvement. RAS systems have continually evolved since the mid 1990s. Each iteration has relied on new technology to add features and improve performance. There is no reason to assume that this trend will end with this version. A few areas have already been identified that could be improved to make the system faster, more user friendly and more capable.

The ν-OTDR waveform transfer speed is limited by the use of the SPI protocol used to pass data between the FPGA and the RCM3360. The RCM3360's slow processor speed combined with the large size of the waveform result in a transfer that takes 5 seconds for each ν-OTDR scan. A redesigned communication bus, most likely parallel, could be implemented to improve the transfer speed. Waveform compression implemented on the FPGA would reduce the amount of data to transmit (and therefore improve the transfer speed). Recall that during a breach, waveform scans are continually requested by the host to track the breach location. The time it takes to report a breach location change is dictated by how fast the waveform can be acquired and analyzed. It is highly desirable from a security and accountability standpoint that breach location changes be reported as quickly as possible.

Three issues that negatively impact breach location performance have been observed. These issues are not caused or enhanced by this particular implementation of a ν-OTDR, but are issues present with all tested ν-OTDRs and OTDRs, including commercial units. The first performance issue is weak Fresnel reflection strength for some seals. The second is weak Fresnel reflections due to a large seal count. The third is high level ghosting that corrupts the waveform.

Weak Fresnel reflections are probably caused by an ST-ST connector being too good. In some cases, the fiber is coupled near perfectly, which reduces the amount of backscattered light. In approximately 10 percent of seals, very small Fresnel reflections are observed. If the reflections are just barely above the noise floor, it is difficult for the user to characterize the system and even more difficult for the software to perform a breach location algorithm. A method needs to be developed to increase the Fresnel reflections of these connections. This method must be repeatable, must not affect the opening or closing of seals and must be permanent. Any solution will be somewhat of a double edged sword. As Fresnel reflection strength is increased, transmitted light is decreased. In effect, improving the performance of the ν-OTDR via physical modifications will negatively impact the performance of the event detector.

Weak Fresnel reflection strengths also result when the seal count increases over approximately 20 seals. At this point it becomes difficult to isolate some seals from the noise floor. Options for fighting this include increasing the scan time or taking multiple scans to pull the small signals from the noise floor. The downside of either approach is a further decrease in breach location speed. However, if the waveform transfer speed is improved as discussed above, increased scan time or multiple waveform acquisition would become a more feasible option.

Another method that could be used to improve the dynamic range is to change the ν-OTDR design to include gating. Scholder, Wegmüller and Gisin showed that the minimum detectable power of a ν-OTDR diminishes with the square root of the gate duration and the total measurement time [8]. The addition of gating with an

appropriately small duration would trade waveform acquisition speed and design complexity for improved dynamic range.

The ν-OTDR waveform can become corrupted with copies of reflections when severe ghosting occurs. A waveform plot complicated with ghosts can be very difficult or nearly impossible to characterize. In extreme cases the ghost signals can have amplitudes similar to the real reflections. Extreme ghosting presents a major problem for the breach detection algorithm when a ghost appears at the same location as a real signal. There is virtually no way the algorithm can conclude whether the signal is real or a ghost. A proposed solution is to use varying fiber lengths between seals. It is expected that careful length selection can help avoid compounding ghosts that multiply to form large signals. It has also been observed that ghosting may be more prominent with certain brands of fiber connectors. To optimize ν-OTDR performance, future study should be devoted to ghosting minimization techniques.

Continued research will ensure that the performance and security of the RAS system is maximized. Enhancements leading to an improved SNR will provide more seals per EDL, which will help lower the cost per seal and simplify installation. Enhancements leading to improved waveform acquisition speed will further increase the security of the system by reducing the amount of time spent out of the event detection mode. In any case, today's world necessitates economical safeguarding technology; not only for assets valuable to our national security, but also for assets that could potentially do harm in the wrong hands. Continued research will ensure that ReflectoActive™ Seals systems will meet the safeguarding needs of the future.

REFERENCES

[1] M. Earley, *National Electric Code Handbook* Quincy, Massachusetts: National Fire Protection Association, 2002.

[2] D. Smith, J. Muhs, C. Pickett and D. Earl, "Method and Apparatus for Active Tamper Indicating Device Using Optical Time Domain Reflectometery," U.S. Patent No. 6,002,501, December 14, 1999.

[3] M.K.Barnoski and S.M. Jensen, "Fiber waveguides: a novel technique for investigating attenuation characteristics," *Applied Optics,* vol. 15, pp. 2112-2115, September 1976.

[4] M. Wegmuller, F. Scholder, and N. Gisin, "Photon-Counting OTDR for Local Birefringence and Fault Analysis in Metro Environment," *Journal of Lightwave Technology*, vol. 22, pp. 390-400, February 2004.

[5] B. Huttner and J. Bredel, "Photon-Counting techniques for fiber measurements," *Journal of Lightwave Technology*, vol. , pp., August 2000.

[6] P. Healey, "Optical Time Domain Reflectometery – A performance comparison of the analogue and photon counting techniques," *Opt. Quant. Electron.,* vol 16, pp. 267-276, 1984.

[7] E. Pearson, *How to Avoid Cursing at Cursers: An Introduction To Interpretation of OTDR Traces,* Pearson Technologies Inc.

[8] F. Scholder, J.D. Gautier, M. Wegmuller, and N. Gisin, "Long-distance OTDR using photon counting and large detection gates at telecom wavelength," *Opt. Comm.,* vol. 213, pp. 57-61, 2002.

[9] A. Lacaita, P. Francese and S. Cova, "Single-photon optical-time-domain reflectometer at 1.2 um with 5-cm resolution and high sensitivity," *Opt. Letters,* vol 18, pp. 1110-1112, 1993.

[10] F. Scholder, A. Fougères, J.D. Gautier, C. Barreiro, A. Haldimann, H. de Reidmatten, M. Wegmuller, and N. Gisin, "Photon-counting OTDR at telecom wavelength: High-resolution and long-distance measurements," in *Proc. Symposium on Optical Fiber Measurements 2002, NIST Spec. Publ. 988,* 2002, pp. 157-160.

[11] G. Ribordy, J-D. Gautier, H. Zbinden, and N. Gisin, "Performance of InGaAs/InP avalance photodiodes as gate-mode photon counters," *Applied Optics*, vol. 37, pp. 2272-2277, 1998.

[12] Astrodyne Technical Staff, *MCA-30 Series Datasheet*, Astrodyne Corporation, Taunton, MA., 2005.

[13] Altera Technical Staff, *Altera Stratix Device Handbook Volume 1,* Altera Corporation, 2006.

[14] Altera Technical Staff, *Altera Stratix Device Handbook Volume 2,* Altera Corporation, 2006.

[15] Altera Technical Staff, *Configuration Handbook Volume 2,* Altera Corporation, 2006.

[16] Corning Technical Staff, *Corning InfiniCor 300 Optical Fiber Product Information*, Corning Incorporated, 2002.

[17] D. Marcuse, Dietrich, *Priniciples of Optical Fiber Measurement*, New York: Academic Press, 1981.

[18] R. Jaeger, *Microelectronic Circuit Design,* Boston: McGraw-Hill, 1997.

[19] Analog Devices Technical Staff, *Single-Supply, High Speed PECL/LVPECL Comparators ADCMP551/ADCMP552/ADCMP553*, Analog Devices, Inc., 2004.

[20] Rabbit Semiconductor Technical Staff, *Rabbit RCM3360 RabbitCore Microprocessor Core Module*, Rabbit Semiconductor, Inc., 2005.

[21] Rabbit Semiconductor Technical Staff, *Dynamic C For Rabbit Semiconductor Microprocessors Integrated C Development System User's Manual*, 2006.

[22] Fairchild Semiconductor Technical Staff, *LM555/NE555/SA555 Single Timer*, Fairchild Semiconductor, 2002.

[23] EXFO America Inc., Appl. Note 043, pp. 1-3.

[24] Dicon FiberOptics Technical Staff, *TF500 Tunable Filter FA500 Variable Attenuator Operation Manual*, DiCon Fiberoptics, Inc., 2001.

[25] Perkin Elmer Techical Staff, *Single Photon Counting Module SPCM-AQR Serier*sm Perkin Elmer, 2001.

[26] Matrix Orbital Technical Staff, *GLK12232-25-WBL User Manual,* Matrix Orbital.

[27] D. Kalinsky and R. Kalinsky, "Introduction to Serial Peripheral Interface," February 2002, http://www.embedded.com/story/OEG20020124S0116.

[28] M. Abzug, "MD5 Homepage (unofficial)," http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html

[29] J. Franks, P. Hallam-Baker, J. Hostetler, P. Leach, A. Luotonen, E. Sink, and L. Stewart. "An Extension to HTTP : Digest Access Authentication," January 1997, http://www.w3.org/Protocols/rfc2069/rfc2069

[30] "ASCII Table and Description", http://www.lookuptables.com.