FIBER OPTIC SYSTEM DESIGN EDUCATIONAL TOOL

John Frederick Hulbert
B.S., California State University, Sacramento, 2002

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND ELECTRONIC ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SUMMER
2009
Student: John Frederick Hulbert

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

Signature removed

Dr. Preetham Kumar

Department of Electrical and Electronic Engineering

Date 08/09/09.
Abstract

of

FIBER OPTIC SYSTEM DESIGN EDUCATIONAL TOOL

by

John Frederick Hulbert

Fiber optic communication systems consist of various optical component systems including optical sources, fibers and detectors with unique characteristics that ultimately determine system performance. FOSDET is an educational tool designed for student use to better understand the interaction between the key components of the communication system. The impact of different configurations and device parameters on the overall system performance is studied with the aid of computer simulation.

Signature removed

Dr. S.K. Ramesh, Committee Chair

August 05, 2009
Date
I would first like to thank Dr. S.K. Ramesh and Dr. John Oldenburg for their kindness, patience, and help through the duration of my graduate experience here at Sacramento State. I thank my wife Lora for her love and support. I would also like to thank my good friend Brandon Carroll for all of his coaching as a “C# Expert”. Last, but not least, I want to thank my older brother Jared for helping me get started and sharing with me useful ‘tips of the trade’ as an accomplished electronic\software engineer.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>Software Information</td>
<td>viii</td>
</tr>
</tbody>
</table>

## Chapter

1. **INTRODUCTION** .................................................... 1

2. **PROGRAM STRUCTURE** .................................................... 5

3. **THEORETICAL MODELING OF FOSDET COMPONENTS**.................................................... 11
   - Tracking the Pulse Spread .................................................... 12
   - Tracking the SNR .................................................... 13
   - Calculating Results .................................................... 15

4. **MODULE OVERVIEW** .................................................... 17
   - Source Class: Laser1.cs & SourceCustom.cs .................................................... 18
   - Modulator Class: ModulatorCustom.cs .................................................... 18
   - Fiber Class: Fiber1.cs and FiberCustom.cs .................................................... 19
   - Amplifier Class: EDFA1.cs and EDFAcustom.cs .................................................... 24
   - DispersionCompensator Class: DCM1.cs and DCMCustom.cs .................................................... 27
   - Coupler Class: Coupler1.cs and CouplerCustom.cs .................................................... 27
   - Detector Class: APDCustom.cs and PINCustom.cs .................................................... 27

5. **USING FOSDET** .................................................... 31
   - System Overview Form .................................................... 31
   - Add\Edit Component Form .................................................... 33

6. **CONCLUSION** .................................................... 36

References ............................................................... 39
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Fig. 1. FOSDET class hierarchy diagram</td>
<td>8</td>
</tr>
<tr>
<td>2.</td>
<td>Fig. 2. Attenuation vs. wavelength arrays used in Fiber I module.</td>
<td>21</td>
</tr>
<tr>
<td>3.</td>
<td>Fig. 3. Screenshot of main 'System Overview' GUI</td>
<td>32</td>
</tr>
<tr>
<td>4.</td>
<td>Fig. 4. Screenshot of 'Add/Edit Component' GUI</td>
<td>34</td>
</tr>
</tbody>
</table>
SOFTWARE INFORMATION

The submitted software is a completed copy of this project. The program is called FOSDET. FOSDET is an educational program designed to help students learn fiber optic communication system design techniques. The software requirements are as follows:

- Microsoft Windows XP
- PC with 300 megahertz or higher processor clock speed
- 128 megabytes (MB) of RAM or higher
- 5 megabytes (MB) of available hard disk space
- Super VGA (800 x 600) or higher-resolution video adapter and monitor
- CD-ROM or DVD drive
- Keyboard and compatible pointing device
In the field of education, there are many theories as to what constitutes the most effective method of learning. As technology advances, new methods for learning are created and with them new theories. It seems that most educators today end up using a blend of many different teaching methods. This seems particularly true in technical fields, such as engineering. Though technology may alter the format of how information is presented to us, it can never replace the need for diligent study of the material in question. However, most would agree, and I have found this to be true in my own educational career, that some form of 'hands-on' experience with the concepts learned through lecture and study can help to clear up misconceptions and solidify the material.

There are many different forms of hands-on experience and the medium through which this experience is delivered depends greatly upon the material being taught. For example, a student learning the different concepts of effective coding techniques can easily design his own programs. In the case of an engineering student learning effective design techniques for a fiber optic communications system, it would be impossible to actually build a large-scale system as an educational exercise. In this case, a suitable substitute would be to design the system on a simulation tool. Simulation tools may not always provide all of the detail and experience that a real world exercise might, but they offer an effective medium through which essential concepts can be taught in preparation for real world exposure.
Over the years there have been many simulation models developed of different fiber optic communication system components. These models help researchers better understand limiting factors in fiber optic systems and how to overcome them. A model of the non-uniform FM response in semiconductor lasers was developed using recursive digital filters models [1]. The frequency and impulse responses for α-profile multimode optical fibers were characterized by using a linear systems approach. This method was developed to help specifically in the design of computer communication networks, but could be applied to other systems using multimode optical fibers [2]. In another case, a mathematical model was designed to help analyze and develop a proposed solution to clipping noise generated in AM/QAM hybrid light-wave systems [3]. By comparing a link model with an equivalent-circuit model of an optical link, some researchers at MIT discovered that an optical link 3-dB limit was caused by the ohmic nature of the modulator impedance and not from the electro-optic conversion process as previously thought. With that information and previously mentioned models it was shown the system 3dB limit could be lowered by employing impedance mismatching techniques [4]. Another researcher built upon a numerical split-step operator splitting modeling technique by using a cubic B-spline method. This numerical method used to model optical fibers allows for the nonlinear simulation and observation of optical communication links. In other words, smaller sampling intervals can be collected in locations of interest while larger sampling intervals can be used in areas of little concern [5]. In another paper, two models, string length model and modified quadratic model, are used and compared to investigate system penalties caused by polarization-mode
dispersion [6]. Many of these models focus on only single aspects of a fiber optic link and in some cases, of a single component. These have proven useful to researchers trying to overcome specific problems, but they have not been organized into a comprehensive tool capable of modeling an entire customizable optical link from source to detector with an array of customizable components to choose from for educational use.

In one paper, it was shown that a program simulating an entire communications system project could be developed using HDL-A, a hardware description language based on VHDL. This method utilized both analytical equations and numerical methods to describe the component operation. This approach allowed for optical communication links to be integrated into a circuit simulator environment allowing for system designs in a top-down approach [7]. This paper, however, only developed models for a semiconductor laser and single-mode fiber.

For my project, I have designed a program that will allow students to build a simple single line fiber optic communications system with the intent to allow students to experiment with the basic concepts of fiber optic communications system design. In designing an optical communications system, engineers have to understand that most system parameters (i.e. bandwidth, noise levels, data rates, BER, signal power, pulse spread, etc.) have a sort of ‘give-and-take’ relationship with each other. They are all connected. This project was designed to help engineering students better comprehend this relationship through the ability to easily design and experiment with different system configurations. The program is called Fiber Optic System Design Educational Tool (FOSDET).
FOSDET was designed with two potential exercises in mind: the primary function is to allow a student to design a fiber optic communications link; a secondary exercise option is to allow a student to design and code an individual component model and add it to FOSDET. This secondary option would allow a student to gain more experience with fiber optic communication systems on a fundamental component level. However, it does require a certain level of programming ability from the student, which will limit the number of potential applications of this exercise.
When approaching a program like this, one must first create a general layout of the structure and how the different parts will interact with each other. I determined that the best way to structure the program was to mimic the structure of a real fiber optic communications system. The program will have an array of available component modules. Each module is designed to simulate an individual component, i.e. detector, source, fiber, amplifier, etc. These modules are interchangeable and can be strung together in a linear fashion in any order the user desires. A special module designed to act as an optical signal exists to contain and track critical signal information such as pulse spread, signal power, and noise. The signal module is passed down the linear chain of component modules to simulate signal propagation through the system. As a component module receives the signal, it manipulates the signal parameters trying to mimic the real world effects the component would have on an optical signal. The changed signal then goes on to the next component on so on until it reaches the detector module. A main system module builds the system based on the design parameters defined by user input. The main module also starts the signal module propagation, performs final system parameter calculations using information contained in the signal module, and displays the results to the user in a useful manner.

The actual implementation of the previously described modules depends on the programming language. After considering a variety of options, I finally decided to use
C#. I made this decision for a number of reasons. C# is a widely used, well developed programming language that has excellent support and can be coded using the impressive Visual Studio platform. Since C# is a well known industry standard, most universities have access to Visual Studio, potentially allowing for additions and upgrades by any such minded students in the future.

During the entire process of creating FOSDET, special effort was put forth to ensure that the program remained highly customizable. In my opinion, the more flexible and customizable the program, the more opportunities the student has to learn, as the program can be altered to accommodate more situations. However, there are always limitations and some parameters need to be set in stone. In FOSDET, the communication links designed are single linear paths from point A to point B. There are no networks or branching. The most basic system possible in FOSDET is a source, modulator, and detector. The source, modulator, and detector are referred to as the ‘essential’ components. Every system must always include one of those three components, and only one, in order for the system to be valid. To stop any errors associated with this concept, FOSDET does not allow the essential components to be added, removed, or moved within a system. Each system comes preloaded with them already in place. However, the user is allowed to edit the parameters within the essential components and change the models as desired. The non-essential components (i.e. fiber, amplifiers, connectors, dispersion compensators, etc) can only be placed in between the modulator and detector.

Special care was taken while programming FOSDET to structure it in such a way to make adding component modules as easy as possible. All components, both the
essential and non-essential, inherit properties from interface module \textit{IComponent.cs} in FOSDET. \textit{IComponent} groups all components together and defines common properties that all components need to have. All the essential component type classes, \textit{Source.cs}, \textit{Modulator.cs}, and \textit{Detector.cs} inherit the properties directly from \textit{IComponent}. All non-essential component type classes, \textit{Fiber.cs}, \textit{Connector.cs}, \textit{DispersionCompensator.cs}, and \textit{Amplifier.cs} inherit from a parent class called \textit{OptionalComponent.cs} which in turn inherits from \textit{IComponent}. Individual component model classes must inherit from their respective component type class. For example, the fiber models \textit{Fiber1.cs} and \textit{FiberCustom.cs} inherit from \textit{Fiber} which inherits from \textit{OptionalComponent} which inherits from \textit{IComponent}. There are many reasons for this organization. The quick and simple explanation is that it helps the program automatically recognize and implement any new modules that may be added.

For the program to function correctly, each module should have the properties FOSDET expects it to have based on its type. The designed class hierarchy ensures that. For example, since essential classes cannot be added, removed, or moved, they need to be distinguished from the non-essential classes. This is why the non-essential classes inherit from \textit{OptionalComponent} while the essential classes do not. Another example of the need for this hierarchy is the case of the detector. This will be discussed in more detail further on, but FOSDET requires that all detector models have a responsivity variable accessible to the rest of the program. The \textit{Detector} parent class ensures this. The hierarchy also ensures that the models are properly organized to help users navigate and find the desired component easily.
Each component model, both essential and non-essential, have three basic needs to function properly in FOSDET. These are; a name property, a description property, and a propagate function. The name and description properties provide useful information about the module to the user. The propagate function takes a reference to the signal class as an input and manipulates the signal class according to the component function. If a component has user defined variables, said variables need to be properties of the component module to allow other modules easy access to them. A GUI called a user option pane also needs to be designed, providing a way for the user to input the needed variables. The option pane then needs to be linked to the component module variable properties.
It should be mentioned that the form called FOSDETForm.cs is the main system GUI viewed when FOSDET is first opened. The AddComponentForm.cs form is the GUI seen when adding or editing a module. The FOCSystem.cs module is the main module that binds everything together in FOSDET. It creates the system the user defines, calls propagation functions, and calculates the final system parameters depending on the values the signal class returns after propagation through the system.

Since FOSDET was created with future additions in mind, each component module must be as self-sufficient and self-contained as possible. All problems and needs must be met within the confines of the individual modules where possible. As an example, the Fiber1 model uses a table of values to provide an attenuation constant for a given wavelength. If the signal used a wavelength outside the bounds of that table the system would encounter a ‘parameter out of range exception’ and crash. Since other fiber modules might not have this limitation it is better to catch the error in the individual module itself, instead of limiting the entirety of FOSDET.

FOSDET only catches errors that would cause the system to crash. The user must monitor all other system failures. For example, the parameters used in a component module may be optimized for a certain wavelength, BER, and data rate. If the signal does not match these optimized conditions, FOSDET will not stop the simulation. This is done to allow students more freedom to experiment with ‘what if’ type situations. However, the user should always be aware of these limitations. Another common error is unit conversion. To try to limit unit errors all units are converted to their basic form. All
user input GUI's should be clearly labeled with units as well the code defining the modules.
Chapter 3

THEORETICAL MODELING OF FOSDET COMPONENTS

In general, there are two factors that limit the distance an optical signal can propagate down an optical fiber communications link. The first is signal power attenuation and noise generation. The second is dispersion and component rise time causing the signal pulse to widen. There are many books that teach these concepts and though the basic concepts are the same, each book teaches students to track these limiting factors in different ways. Some books just track the attenuation the signal encounters and then compare the power that is left with the minimum power the detector can see [8]. This approach is a simple and effective, but only usable on simple systems where the noise is assumed to be zero. Since FOSDET allows for the effects of noise in the system, a more involved method had to be used. Using the Signal to Noise Ratio (SNR) is what I found to be the most effective way to monitor the signal’s power requirements with the inclusion of system noise. To add the effects of noise to the systems power requirement, one adds the squared noise term of each contributing noise factor to the denominator of the system SNR. This method also allows for the simple addition of new noise factors in future FOSDET add-ons.

The method for tracking dispersion and rise time effects was a lot more straightforward. Each effect correlates directly with an actual amount of pulse spread. All that really had to be done was to keep track of that spread. One of the harder parts of
the program was to decide how the signal was going to carry this information from component to component, or in other words, in what form.

A. Tracking the Pulse Spread

The signal pulse spread is caused by the dispersion experienced while it propagates through the various system components. The amount the signal pulse spreads is calculated by taking the square root of the added squares of each contributor to the overall signal pulse width increase [8:328]. See (1).

\[ \Delta \tau = \sqrt{\tau_A^2 + \tau_B^2 + \tau_C^2} \] (1)

The fact that each contributor to pulse broadening is squared before being summed, necessitated that an individual variable be created in the signal class for each contributor. That way, as the signal propagates, each component can add pulse spreading effects to the variable specific to that type of effect. In the case of FOSDET, four different variables were created in the signal class to track pulse width degradation: inter-modal dispersion \((\text{interDispersion})\), intra-modal dispersion \((\text{intraDispersion})\), polarization mode dispersion \((\text{pmDispersion})\), and component rise times \((\text{riseTimes})\). The rise time variable was created to store the rise time effects of non-fiber components such as the source, detector, amplifiers, etc. It has units of seconds squared since each
component is a separate and individual contributor to the total pulse spreading. The variables for inter, intra, and polarization mode dispersion are kept in units of seconds. The three dispersion variables are kept in units of seconds since, in any give system, there may be any number of fiber links or dispersion compensation units that each contribute to the total of the individual dispersion effect. After the signal finishes propagating, the three dispersion variables are squared and added together with the rise time variable. Equation (2) shows the exact equation used in FOSDET to calculate the pulse width increase. The units are in seconds.

\[
\Delta \tau_{sys} = \sqrt{\text{interDispersion}^2 + \text{intraDispersion}^2 + \text{pmDisperion}^2 + \text{riseTime}} \tag{2}
\]

B. Tracking SNR

For the SNR, two things need to be tracked: signal power and system noise. Keeping track of the power is simple. The source defines the beginning power level and from there on out the different devices inflict loss or gain on the power. In this project, I chose to keep it in units of dBW allowing us to subtract and add to the signal power variable. However, in coding the individual component modules one must keep in mind that the power is in dBW and make the necessary conversions.
The system noise was not quite as easy as the system power. The system noise is needed to calculate the SNR after the signal has propagated. The equation used to calculate the SNR of the system in FOSDET can be seen in (3) [9:516].

\[
\text{SNR}_{\text{sys}} = \frac{i_S^2}{i_N^2} = \frac{i_S^2}{\sigma_A^2 + \sigma_B^2 + \sigma_C^2} = \frac{(R_{\text{in}})^2}{\sigma_{\text{thermal}}^2 + \sigma_{\text{shot}}^2 + \sigma_{\text{EDFA}}^2 + \sigma_{\text{noise}}^2}
\] (3)

The three squared sigma terms are a generalization of the various noise contributions added to a system by different components/effects. In this version of the program, four noise effects are considered in the calculation of the SNR. Two of those are the shot and thermal noise in the detector. We don't need to track those two because they are generated at the detector and therefore are not calculated until the end. However, the third and fourth noise terms do need to be tracked because they are the noise generated by components that can be placed anywhere in the system. The first of these variables (\(\text{sigma2EDFA}\)) is the noise generated by the EDFA amplifiers that are currently available in FOSDET. The second variable (\(\text{noise}\)) exists for any future components that might introduce noise. Each component module that generates noise should add that noise to the appropriate noise variable in the signal class. It is important to note that each component that introduces a signal attenuation or gain must be sure to reflect that same attenuation or gain in the signal noise variables as well. The EDFA noise needs its own variable because of its dependency on system parameters at the detector. This will be discussed in more detail later on in this report.
C. Calculating Results

As the signal class is passed from component to component it collects all of the variables needed to make the final calculation of the system SNR and pulse width increase. By the time the signal class exits the detector module all of the information should be available. The collected noise variables are used to calculate the SNR, see \( (3) \), and the square root of the sum of all the rise time variables squared is taken to calculate the systems pulse width increase, see \( (2) \). We then can compare these values to the minimum SNR and the maximum pulse spread allowed in our communications link.

The minimum SNR allowed is calculated with the bit error rate (BER) of the system as seen in \( (5) \) [9:498].

\[
BER = \frac{1}{2} \text{erfc} \left( \frac{\sqrt{SNR}}{2\sqrt{2}} \right) \tag{4}
\]

\[
\therefore SNR_{\text{min}} = 8 \cdot (\text{erfc}^{-1}(2 \cdot BER))^2 \tag{5}
\]

The maximum pulse spread can be calculated by using the rule of thumb in \( (6) \) [8:328].

\[
\Delta \tau_{\text{max}} = 0.70 \cdot \frac{1}{B} \tag{6}
\]
Note that the electrical bandwidth \( B \) is equal to the data rate for NRZ signals and is double the data rate for RZ signals. If the actual system SNR and pulse width increase calculated in (2) and (3) are greater than and less than the SNR and pulse width increase calculated in (5) and (6) respectively, then your system design works. If not then the system design cannot successfully operate and needs to be changed.

Another limiting factor in designing any system is the cost. Cost is a factor of systems design sometimes overlooked in school projects. Perhaps this is because some view it as a simple, ‘non-technical’ concept that need not be considered in the academic realm. I feel that not including cost as a simulation factor is a huge mistake seeing as how in most real-life applications cost is one of the leading limiting factors. Including cost will help students to focus on the systems critical design specifications and learn to juggle the give and take relationship between different parameters that is integral to any systems design process. FOSDET has the ability to calculate the total cost of the system. It is a fairly linear process. Each module has a cost parameter that is added to a system cost variable in the signal class during propagation. The cost variable will not always be mentioned in the following module overviews, it assumed that each component has the variable included and needs no further explanation.
Chapter 4

MODULE OVERVIEW

In this section, I will give an overview of the different component modules currently available in FOSDET. It should be noted that some of the components have predefined parameters. The parameters used are not identical to, but generally loosely based on (and in some cases combinations of), parameters pulled from various industry device specification sheets and examples from previous college courses. The following module explanations will not necessarily include the parameter values, the focus will be mostly on the theory. Detailed parameter descriptions are available while selecting the components in FOSDET itself.

FOSDET has been designed with two potential exercises in mind that could help strengthen an engineering student's understanding of a fiber optic communications link. The first is to use the program to design and simulate optical communication links and learn the 'ins and outs' of that process. The second is to create an individual component module to help teach the operational details involved at the fundamental component level. To show both of FOSDET's potential applications, I have created two versions of many of the component classes. Generally for those classes with two modules, one version is an unchangeable component with predefined parameters, the type a student could design as a class project. The other version has plenty of customizable parameters to allow the users to add their own component to the system in real time when flexible
system design is the student's need. The following modules are currently available in FOSDET.

A. Source Class: Laser1.cs & SourceCustom.cs

The source class sets three parameters that are critical to the communication system. These parameters are the signal power, signal wavelength, and source spectral width. All other parameters, such as noise, rise time, and cost, depend on the accuracy the user wants in the model. Laser1 requires no user input while SourceCustom is fully user defined and can be used as either a laser or LED.

B. Modulator Class: ModulatorCustom.cs

The modulator must define the BER, electrical bandwidth (B), and signal spectral width. ModulatorCustom asks the user for an array of signal parameters necessary to calculate the aforementioned critical modulator defined parameters. ModulatorCustom asks for the BER, data rate, and signal coding scheme. The signal electrical bandwidth is the same as the data rate if the coding scheme used is non-return to zero (NRZ), but if return to zero (RZ) is used the electrical bandwidth is double the data rate [8:335-338]. Logic code in the modulator class determines and sets the signal electrical bandwidth based on the users input.
The signal spectral width is the larger of the source spectral width and the modulation spectral width. The source spectral width is a parameter already defined by the source. The modulation spectral width ($mod\text{SpecWidth}$) can be calculated from the modulated optical bandwidth ($mod\text{OpticalBandwidth}$). The modulated optical bandwidth is found by taking the Fourier transform of the modulated pulse and extracting the 3dB bandwidth. The current modulator module available in FOSDET, $ModulatorCustom$, assumes a square wave modulation. A square wave signal modulated bandwidth can be approximated as double the electrical bandwidth. Equations (7) and (8) illustrate $ModulatorCustom$’s calculation of the modulated spectral width.

\[
mod\text{OpticalBandwidth} \approx 2 \cdot B 
\]

(7)

\[
mod\text{SpecWidth} = mod\text{OpticalBandwidth} \cdot \frac{\lambda^2}{c} 
\]

(8)

After the calculation of the modulated spectral width, logic code in $ModulatorCustom$ compares the source spectral width with the modulated spectral width and sets the larger of the two as the signal spectral width ($spec\text{Width}$). Insertion loss, cost, and rise time variables can be included depending upon the accuracy that the user desires.

C. Fiber Class: Fiber1.cs and FiberCustom.cs
Typically, a fiber class has two effects on the signal class. The first are dispersion effects and the second is signal power attenuation. The FiberCustom module allows the user to input any value desired for fiber length, inter-modal dispersion, intra-modal dispersion, polarization mode dispersion, and the attenuation constant. A cost per meter can also be associated with the length of fiber. Values for the total dispersion and attenuation encountered are calculated using the fiber length. These calculations are very straightforward. Examples of the three dispersion equations are seen in (9), (10), and (11). Note that all three equations have units of seconds.

\[ \Delta \tau_{\text{inter}} = D_{\text{inter}} \cdot \text{Length} \]  
\[ \Delta \tau_{\text{intra}} = D_{\text{intra}} \cdot \text{specWidth} \cdot \text{Length} \]  
\[ \Delta \tau_{\text{PMD}} = D_{\text{PMD}} \cdot \sqrt{\text{Length}} \]  

The Fiber1 module has the same basic concept as the FiberCustom except there are algorithms that calculate the inter-modal dispersion, intra-modal dispersion, and the attenuation constant based on the system wavelength. The polarization mode dispersion in this model has been simplified to be the same for all wavelengths. It should be noted that the Fiber1 module is a good example of a module design exercise for students.

The intra-modal dispersion is calculated using (12) where the zero-dispersion wavelength \( (\lambda_0) \) and dispersion slope \( (S_0) \) are given from the fiber parameters. Note that the resulting units of (12) are seconds per meter squared [8:128].
The attenuation is taken from an attenuation vs. wavelength array I created mimicking the behavior of fibers commonly used in industry. Fig. 1 shows what the array looks like when plotted. Note that when Fiber1 is being used, the signal wavelength must be within the array boundaries.

\[ D_{\text{wina}} = \frac{S_o}{4} \left( \frac{\lambda - \lambda_o}{\lambda^3} \right) \]  

(12)

Fig. 2. Graph of attenuation vs. wavelength arrays used in Fiber1 module.
The inter-modal dispersion calculation algorithm is a little more involved. In short, it uses the fiber mode equations and numerical methods to solve for the velocity of the fastest traveling mode and the slowest traveling mode and uses them to calculate the inter-modal dispersion. The following is a quick explanation of the theory behind this method.

The first assumption that is made in this method (which is generally true for low dispersion fibers) is that the fiber is weakly guiding or that $\Delta << 1$ where,

$$\Delta = \frac{n_{\text{core}} - n_{\text{clad}}}{n_{\text{core}}}$$  \hspace{1cm} (13)

When this assumption is valid, the fiber mode equation can be simplified into (14) [9:126-134] [8:51-58]. The variable $a$ is the core radius.

$$pa \frac{J_{j+1}(ha)}{J_j(ha)} = qa \frac{K_{j+1}(qa)}{K_j(qa)}$$  \hspace{1cm} (14)

where,

$$p = h = \sqrt{n_1^2 k_0^2 - \beta^2}$$  \hspace{1cm} (15)

$$q = \sqrt{\beta^2 - n_2^2 k_0^2}$$  \hspace{1cm} (16)
This assumption also simplifies the equation used to calculate the normalized propagation constant $b$ as seen in (17) [8:109].

\[
\beta \approx \frac{\beta/k}{n_1 - n_2} 
\]  

(17)

Using (15) and (16) we can then derive the following relationship.

\[
(pa)^2 + (qa)^2 = \left[\left(n_1^2 k_o^2 - \beta^2\right) + \left(\beta^2 - n_2^2 k_o^2\right)\right] \cdot \alpha^2 = \left[k_o a \sqrt{n_1^2 - n_2^2}\right] = V^2
\]  

\[
\therefore qa = \sqrt{V^2 - (pa)^2}
\]  

(18)

(19)

We then plug (19) into (14) and numerically solve for the values of $pa$ that satisfy (14) for each confined mode. The number of modes confined is dependant on the value of $V$. These $pa$ values are then converted to $b$ values using (15) and (17). Solving for $\beta$ in (17) and using approximation shown in (20) gives us (21).

\[
\frac{n_1^2 - n_2^2}{n_2^2} \approx 2\Delta
\]  

(20)

\[
\beta = \frac{\omega n_2}{c} \cdot [1 + b\Delta]
\]  

(21)
The total inter-modal dispersion encountered in a fiber can be described by (22) [8:107].

\[ D_{\text{inter}} = \frac{\Delta \tau_g}{L} = \frac{1}{V_{g_{\text{min}}}} - \frac{1}{V_{g_{\text{max}}}} \]  

where,

\[ \frac{1}{V_g} = \frac{\partial \beta}{\partial \omega} \]  

Plugging (21) into (23),

\[ \frac{1}{V_g} = \frac{1}{c} \left( 1 + \Delta b \right) \cdot \left[ n_2 + \omega \frac{\partial n_2}{\partial \omega} \right] = \frac{1}{c} \left( 1 + \Delta b \right) \cdot n_{2g} \]  

Finally plugging (24) into (22) gives us,

\[ D_{\text{inter}} = \frac{n_{2g}}{c} \Delta (b_{\text{max}} - b_{\text{min}}) \]  

The last step in calculating the inter-modal dispersion is to find \( b_{\text{min}} \) and \( b_{\text{max}} \) in the array of \( b \) values numerically calculated previously, and plug them into (25).

**D. Amplifier Class: EDFA1.cs and EDFACustom.cs**

The EDFA classes are amplifiers that add polarization mode dispersion, gain, and noise to the signal. The dispersion and gain are both defined by fixed values or user input. The method used to calculate the noise added to the system by the EDFA is a
simplified model with 'in-the-ball-park' accuracy. It assumes a large gain which in turn leads to the assumption that the beat noise of the signal with the amplified spontaneous emission (ASE) is the most dominant noise term of the EDFA. With these assumptions in mind, the noise added to the system by the EDFA can be described by (26) [8:438-440] [9:755-766].

\[
\sigma_{\text{sig-ASE}}^2 = 4R^2GP_{\text{in}}S_{\text{ASE}}B
\]

(26)

where,

\[
S_{\text{ASE}} = (G - 1)n_{sp} \frac{hc}{\lambda} \approx G \cdot n_{sp} \frac{hc}{\lambda} \quad \text{(assuming large gain)}
\]

(27)

The only unknown in (26) and (27) is the spontaneous emission factor, \(n_{sp}\). Most specification sheets will provide a noise figure (NF) value for their EDFA. Assuming a large gain and using the definition of the NF term we can derive a relationship between the NF and \(n_{sp}\) [8:440]. This relationship can be seen in (28).

\[
n_{sp} = \frac{NF}{2\eta_{\text{eff}}}
\]

(28)

Something that should be noted is that the EDFA noise depends on some values that aren't known until the system signal reaches the detector: the detector quantum efficiency, responsivity, and the signal power at that detector (\(P_{\text{in}}\)). The \(P_{\text{in}}\) problem was
solved by multiplying the EDFA noise variable by an extra gain or loss term as it propagates, one for the power term $P_{in}$ and one for the noise itself. The quantum efficiency and responsivity problem are one in the same since one can be calculated from the other, see (29).

$$R = \eta_{eff} \frac{q}{h\nu}$$  \hspace{1cm} (29)

The quantum efficiency/responsivity problem was solved by creating a function in the detector parent class that calculates the detector responsivity. When the system propagate function is called the first thing that happens is the responsivity is calculated and stored in the signal class along with the quantum efficiency so any EDFA in the system can access them.

While these solutions to the EDFA’s detector values dependency problem may seem a little messy, there are not a lot of other options to circumvent this problem. Another option I considered for a while was to multiply a stripped down EDFA noise term by the end signal power and the responsivity at the end of the system propagation function. This however, would not have solved the problem of the need for the quantum efficiency, and I wanted to keep the individual component needs confined to their own classes as much as possible to try to make future additions as intuitive as possible. Other variables required are the amplifier noise figure, the signal electrical bandwidth, and the signal wavelength.
E. DispersionCompensator Class: DCM1.cs and DCMCustom.cs

The DCM (Dispersion Compensation Module) module generally adds power loss, PMD dispersion, and a negative material dispersion to the system signal. The material dispersion generated in the DCM is subtracted from the signal material dispersion. All other calculations are straightforward. The only difference between *DCM1* and *DCMCustom* is that *DCM1* has all of its parameters predefined while *DCMCustom* requires user input.

F. Coupler Class: Coupler1.cs and CouplerCustom.cs

The current couplers in FOSDET are very simple models. They only inflict a power loss and a cost to the system. *Coupler1* has set values while *CouplerCustom* requires user input.

G. Detector Class: APDCustom.cs and PINCustom.cs

The detector classes exist primarily to define the detector responsivity, the detector shot noise, the detector thermal noise, and apply a gain (if any) to the signal power and all pre-existing noise variables.
The calculation for the shot noise is pretty straightforward. Assuming surface currents are zero and combining bulk dark current noise with the shot noise [8:252-257] results in (30).

\[ \sigma_{\text{shot}}^2 = 2q(RP_{in} + I_D)M^2 F(M) \cdot B \] (30)

where,

\[ F(M) \approx M^x \] (31)

The noise factor variable \( x \) is always a value between 0 and 1. Common values for \( x \) are 0.3 for Si, 0.7 for InGaAs, and 1 for Ge [8:264]. The user needs to provide the values for the responsivity, dark current \( I_D \), gain \( M \), and \( x \).

Calculating the thermal noise requires a little effort. Equation (32) is the standard equation used to calculate thermal noise [8:256].

\[ \sigma_{\text{thermal}}^2 = \frac{4k_B T}{R_L} B \] (32)

The only unknown in (32) is \( R_L \). In order to calculate the thermal noise in the currently available detector classes, we must assume that the \( R_L \) is the same in our system as it was in the testing setup where the detector parameters were determined. We also assume operating and testing temperatures of 25 degrees Celsius. To make the calculations less cluttered, the constants in (32) are grouped together.
\[ \sigma_{\text{thermal}}^2 = C_{\text{thermal}} B \]  \hspace{1cm} (33)

where,

\[ C_{\text{thermal}} = \frac{4k_B T}{R_L} \]  \hspace{1cm} (34)

Now a value for \( C_{\text{thermal}} \) must be calculated in order to be able to calculate the thermal noise. We can solve for \( C_{\text{thermal}} \) by recreating the SNR equation of the testing setup that provides us with the detector parameters as seen in (35) and (36).

\[ SNR_{\text{specsheet}} = \frac{(R_{\text{specsheet}} M P_{\text{min}})^2}{2q(R_{\text{specsheet}} P_{\text{min}} + I_D) M^2 F(M) \cdot B_{\text{specsheet}} + \frac{4k_B T}{R_L} B_{\text{specsheet}}} \]  \hspace{1cm} (35)

\[ C_{\text{thermal}} = \frac{(R_{\text{specsheet}} M P_{\text{min}})^2}{SNR_{\text{specsheet}} B_{\text{specsheet}}} - 2q(R_{\text{specsheet}} P_{\text{min}} + I_D) M^2 F(M) \]  \hspace{1cm} (36)

where,

\[ SNR_{\text{specsheet}} = 8 \cdot (\text{erfc}^{-1}(2 \cdot BER_{\text{specsheet}}))^2 \]  \hspace{1cm} (37)

The user needs to provide the minimum detectable power, the responsivity of test conditions, the BER of the test conditions, the electrical bandwidth \( B \) of the test conditions, and the gain \( M \).
After calculating the thermal and shot noise, the detector then multiplies the signal power and noise variables by the gain. Then all of these signal and detector noise variables are passed to the main FOSDET module where they will be used in the final system calculations. The only difference that exists between APDCustom and PINCustom is that PINCustom assumes the gain is unity.
A. System Overview Form

The main window of FOSDET is called "System Overview". On the left we see the actual layout of our fiber optic communications system and buttons that help us customize and design our own system. The large text box on the left is called the 'System Layout' text box. Displayed in this text box is the current system's design. Upon starting, the system will automatically include a source, modulator, and detector. These are considered the minimum requirements of a fiber optic communications system. The components are listed in the order that they appear in the communication link, starting with the first component at the top and the last at the bottom.

Clicking on one of the components will result in a description of its functionality and a list of its parameters being displayed in the 'Component Description\System Results' text box. To add a component to your system you must click on the 'Add' button found on the left of the 'System Layout' text box. The new component will be added to the system below the component that was selected when the 'Add' button was clicked. The only exception to this is when the source or detector is selected while the 'Add' button is clicked. In these cases the new component will be added after the modulator or before the detector, respectively. If the components are misplaced, clicking
on the move up or move down buttons will move the selected component up or down one space.

Fig. 3. Screenshot of main 'System Overview' GUI.

Clicking on the 'Remove' button will remove the currently selected component with the exception of the source, modulator, and detector. These components are called the essential components and they cannot be removed. The edit button allows you to edit the parameters of an already existing component. To edit a component, select the desired component in the ‘System Layout’ text box then click on the ‘Edit’ button found on the left. You can also edit a component by double clicking on it in the ‘System Layout’ text box. The edit function is important since it is the only way the initial conditions of the essential components can be changed. Below the ‘System Layout’ text box there are two
buttons, the save button and the load button. These buttons allow the user to save a system design that is wanted for future instances of the FOSDET program. After designing a system, the user can click on the ‘Save’ button and save the system design anywhere in the computer’s file system. The save file can be named whatever the user desires, but the extension should be left unchanged at all times or FOSDET will not recognize the file. A FOSDET system save file has a ‘.fosdet’ extension. After navigating to the desired folder and naming the save file, click the ‘Save’ button to create the saved file. To load a system design, click on the ‘Load’ button and navigate through the computers file system, select the desired FOSDET save file, and click ‘Open’. FOSDET will automatically recognize any FOSDET system files in the currently selected folder.

On the right side there is a large text box that displays a quick overview of the individual component we have selected, or, after running a simulation, it will show the results of the simulation. Next to the ‘Component Description\System Results’ text box is the ‘Run’ button. Clicking on this will simulate the current system displayed in the ‘Current System’ text box. When the program is finished simulating the current system design, the ‘Component Description\System Results’ text box will display the results. This may take several seconds.

B. Add\Edit Component Form
The add/edit component window pops up whenever the ‘Add’ or ‘Edit’ buttons are clicked on the ‘System Overview’ window. The text box on the far left is called the ‘Component Class’ window. This window displays all of the component classes that are available to place in the current location. Selecting one of these classes will result in a list of all the components available of that class type to be displayed in the ‘Component’ text box just to the right. To view a description of an individual component, select a component in the ‘Component’ text box. The component description will appear in the ‘Component Description’ text box. To add the component to the system, click the ‘Add’ button and you will be brought back to the ‘System Overview’ window. To go back to the ‘System Overview’ window without adding a new component click on the ‘Cancel’ button.

![Add/Edit Component GUI](image)

Fig. 4. Screenshot of ‘Add/Edit Component’ GUI.

Some of the components have customizable parameters. When one of these customizable components is selected in the ‘Component’ text box, a field of input text
boxes will appear on the far right side. The user can then fill out the input field and click 'Add' to add the changes to the system. If the user is going to use a component with the same parameters more than once, there is the option to save the current component configuration. After filling out the input field, the user can click on the 'Save' Button. The saving and loading processes for an individual component are identical to those of the system save and load processes, with the only exception being that each component type will have its own unique file extension.
Chapter 6

CONCLUSION

FOSDET has shown a great capability to be an excellent learning tool for students studying fiber optic communication systems. I conducted a small survey on electrical engineering students who had taken a class where fiber optic communication system design was taught. The survey asked students to download and install FOSDET on their computer and run some experimental simulations. They were then asked a series of questions designed to get their opinion on ease of use, flexibility, and thoughts on the usefulness in a classroom setting.

On a scale of 1 to 10 the average for the ease of use was 8.6 with no previous FOSDET experience. One hundred percent of the students felt that FOSDET was a very flexible program capable of handling a large variety of configurations. This same flexibility, however, introduces some limitations. Users will have to understand the limitations and assumptions in each module, both current modules and potential future modules. For example, $EDFA1$ and $EDFACustom$ both use a very simple model for the noise calculation that assumes a large gain and produces ‘ball-park’ results. If more accuracy is desired in the EDFA noise calculations, another module would need to be created.

Another example can be found in $Fiber1$. The model used to calculate the dispersion relies on the assumption that the fiber is weakly guiding. In this instance the assumption will always be true because the index values are set parameters. However, if
a student wanted to modify the model to allow students to design their own fibers by changing parameters such as the index of refraction values and the core diameter, that assumption would have to be revisited. The custom input modules also provide a lot of flexibility in the students learning experience, but it falls upon the student to make sure that values entered are valid for the system parameters like wavelength, BER, power level, etc.

FOSDET was also designed as a platform on which students can develop their own modules to add to the system allowing them to familiarize themselves better with individual component operation. Based on my survey results, all of the students felt that creating their own module would be a beneficial assignment, but approximately eighty percent of the students expressed reservations about their ability to actually write and implement their modules in FOSDET. The weakness of my idea for future student module creation is that electrical engineering students have a wide range of programming proficiency and not all would be up to the task. If a teacher wanted to assign such an exercise, it would be recommended to give an example with detailed instructions on the implementation.

Though FOSDET has potential for add-on modules, many of the students wrote extra comments expressing the current build of FOSDET as an excellent stand-alone program due to the real time component parameter manipulation capability via the custom modules. All of the students surveyed were of the opinion that class exercises using FOSDET would have enhanced their learning experience. FOSDET was designed to be straightforward, usable, flexible, and capable of easy additions without the need for
much explanation. In my opinion, and based on the survey results, it has successfully achieved those goals within its limitations.
REFERENCES


