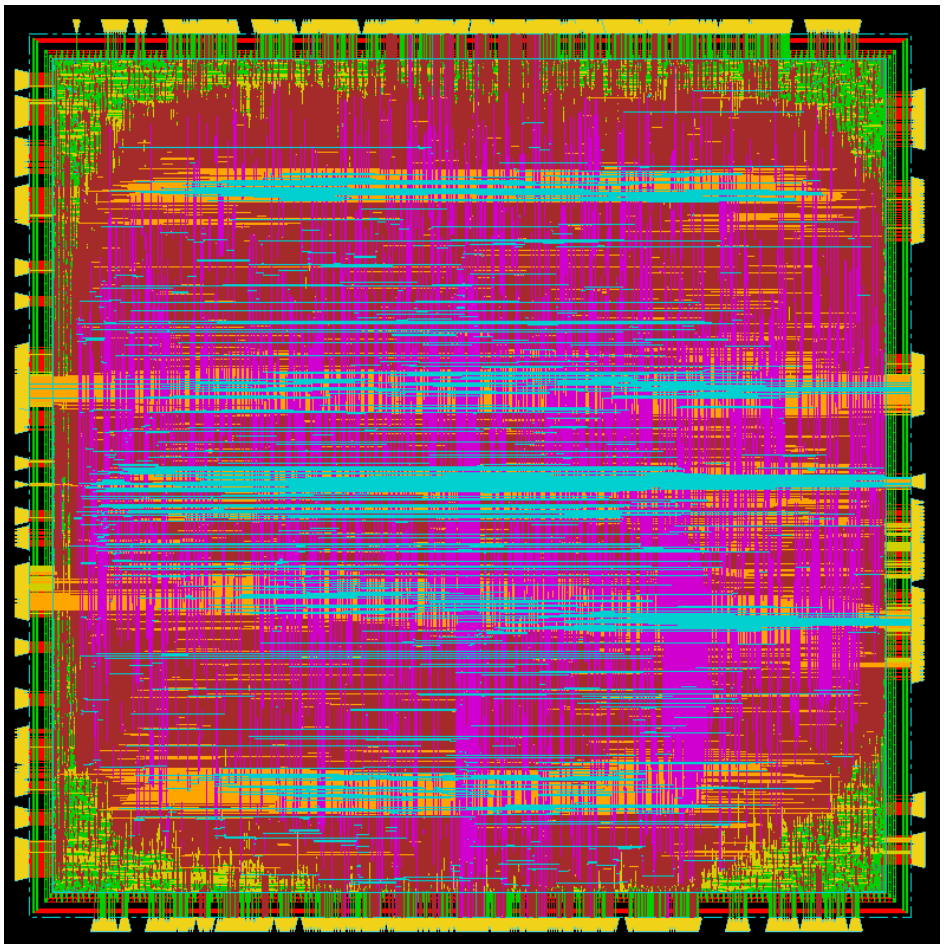
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Time-Domain Digital Back Propagation for Optical Communication in 28 nm FD-SOI

Master's thesis in Embedded Electronic System Design

ANNELI KALANDER
RUI WANG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY AND UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

Time-Domain Digital Back Propagation for Optical Communication in 28 nm FD-SOI

ANNELI KALANDER
RUI WANG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden 2017

Time-Domain Digital Back Propagation for Optical Communication in 28 nm FD-SOI
ANNELI KALANDER
RUI WANG

Supervisor: Per Larsson-Edefors, Department of Computer Science and Engineering
Examiner: Lena Peterson, Department of Computer Science and Engineering

Master's Thesis 2017
Department of Computer Sience and Engineering
Computer Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Placement and routing of a time domain digital back propagating component.

Gothenburg, Sweden 2017

Time-Domain Digital Back Propagation for Optical Communication in 28 nm FD-SOI
ANNELI KALANDER, RUI WANG
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Compensation for signal impairments caused by optical fibers are essential for the communication system when transfering data. Two major impairments are chromatic dispersion and nonlinearities and today these are often compensated for with two separate systems. Digital back propagation is a method used for compensating for both kinds of impairments and, if used in the time domain, it can be implemented in hardware. We have investigated the possibilities of implementing this method in an application specific integrated circuit (ASIC) fabricated in a 28 nm silicon-on-insulator process. Simulations have been done in MATLAB and the design has been implemented with a hardware description language. The conclusion reached is that it is achievable to compensate for both dispersion and nonlinearities with an ASIC of approximately 1 mm^2 at a lower symbol rate and it can be run at a full speed with a larger ASIC area. The achieved bit error rate is 0.007 when compensating for impairments accumulated over 4000 km which is an acceptable result.

Acknowledgements

We want to thank our supervisor Per Larsson-Edefors for his support and help throughout the project. We want to thank Christoffer Fougstedt for the initial VHDL and MATLAB code we started this project with and for the help he has given us to solve problems during the way. We want to thank Kevin Cushon for his help with the designing of the layout. Last, but not least, we want to thank Lars Svensson for providing his support, giving us the optimized filter coefficient we needed in our design and for his help with ordering memories.

Anneli Kalander, Rui Wang, Gothenburg, June 2017

Acronyms

ASIC	Application specific integrated circuit
BER	Bit error rate
CD	Chromatic dispersion
CMOS	Complementary metal oxide semiconductor
DBP	Digital back propagation
DCF	Dispersion compensation fiber
DSP	Digital signal processing
DRC	Design rule check
EDFA	Erbium-doped fiber amplifier
ERC	Electrical rule check
FFT	Fast Fourier transform
FD	Frequency domain
FD-SOI	Fully depleted silicon on insulator
FPGA	Field-programmable gate array
IFFT	Inverse fast Fourier transform
LS-CO	Least-squares constrained optimization
LSV	Layout versus schematic
LVT	Low threshold voltage
NLSE	Nonlinear Schrödinger equation
P&R	Place and route
QAM	Quadrature amplitude modulation
RRC	Root raised cosine
SBS	Simulated Brillouin scattering
SDF	Standard delay format
SNR	Signal-to-noise ratio
SPM	Self-phase modulation
SRS	Simulated Raman scattering
SS	Slow-slow
SSFM	Split step Fourier method
ST	STMicroelectronics
STD	Standard
TD	Time domain
VHDL	(Very high speed integrated circuit) hardware description language
XPM	Cross-phase modulation

Contents

1
Introduction

Impairments such as loss and deformation of the signal are a big problem for signal propagation through fibers today. As the fiber communication becomes faster and operates over longer distances this problem becomes bigger. The impairments increase and the signal can become so distorted that the information cannot be transferred correctly.

To solve this problem the impairments have to be compensated for in the receiver end of the system. This can be done by multiple different approaches, for example, an additional fiber part with the opposite impairment characteristics of the main fiber can be added or DSP can be performed on the received signal. Common for all solutions is that the signal compensation has to be fast as the data-rate is high.

An ASIC is a customized circuit intended for one particular use which makes it fast and advantageous for usage for fiber impairment compensation. In order to compensate for impairments with an ASIC, the impairments and thus the propagation of the signal in the fiber has to be known and modeled. Once the impairments are known these can be compensated for by implementing digital back propagation (DBP). DBP is a method that compensates for impairments by adding the same impairments generated by the fiber but with the inverse values. It has been shown that it is feasible to implement DBP in DSP hardware using an algorithm called TD-DBP [1]. This thesis explores implementing a TD-DBP test-circuit, emulating a full system without the constraint of real-time operation.

1.1 Goals

The overall aim with the project is to design an ASIC that can compensate for impairments generated in a fiber with the TD-DBP method. Because of area limitation the ASIC will not be able to compensate for impairments in real-time but we can show that it is possible to do this. To evaluate the project, performance, area, power consumption, speed and functionality of the ASIC will be simulated and evaluated.

The project can be divided into three separate goals; the first is to extend Christoffer Fougstedts design of the TD-DBP algorithm in hardware so it can compensate for multiple spans. This goal includes MATLAB simulations of the algorithm to find optimal compensating parameters. The second goal is to design a test circuit on the ASIC for the TD-DBP method in order to verify the functionality on-chip. The last goal is to design the layout of the ASIC so it can be fabricated.

1.2 Limitations

For this project there are some limitations we have to adapt our ASIC design to.

The ASIC was initially intended to be designed with a 22 nm process which is new to Chalmers and never used before. The setting up of this process took some time and it was not finished when needed in this project. The alternative was to use a 28 nm process to design the ASIC in instead.

As the ASIC is going to be fabricated and the cost increases for every mm^2 the size of the ASIC is limited to 1 mm^2. This size should be enough for a workable TD-DBP component and the testing circuit around it. However, when the size is limited the speed of the calculations in the ASIC is also limited as fewer parallelization paths can be implemented. This is of no great importance as our goal is only to show that it is possible to design an ASIC compensating for impairments and the throughput is secondary to area.

For the test circuit on the ASIC memories are needed for storing the input and output data and the capacity is limited. There are predefined sizes of the memories that can be used and we have to adapt our design to these. These memories have to be generated (both the hardware and the behavioral description) by the company fabricating the ASIC and we do not know how long it can take to generate these or when we have access to them. For our design we need relatively big memories but Chalmers have smaller ones we can use for simulations before we get access to the bigger ones.

Time is also a limiting factor which means that we do not have time to fabricate the ASIC and test it. We will evaluate the function and performance of the ASIC with simulations.

1.3 Approach

From the goals mentioned in section 1.1 the project can be divided into four separate parts.

The first part is to understand how fiber propagation works and what impairments are generated in the fibers and how these can be compensated for. This is done by doing a small literature study on the subject.

The second part is to investigate how the TD-DBP method works and what is needed to implement it in hardware. The parameters needed for the optimal performance of the impairment compensation also have to be obtained. This is done by simulating the TD-DBP in MATLAB with different parameters.

The third part is to design the hardware for the ASIC. This includes both the DBP part and the testing part which verifies the performance and functionality of the DBP. The hardware is designed in VHDL and simulated with QuestaSim [2]. Area, power and timing evaluations of the design are done with Cadence RTL Compiler [3].

The last part is to design the layout of the ASIC and this is done with Cadence Encounter [4]. The layout is done with the 28 nm process as the 22 nm process is not yet accessible.

1.4 Ethical aspects

We have considered the ethic aspects of this project and reached a conclusion that there are no relevant ethics aspects we can consider.

The project contributes to making the optical fiber connections better by reducing the impairments effects of the signal and this leads to better Internet connections in the future. There are a number of different arguments to if the Internet is advantageous or disadvantageous. One is that we have access to a lot of information, which is advantageous, but all of this information might not be true or it could be private, which is disadvantageous. Our project does not change the connections so much so that we can say we make a big difference.

1.5 Overview

In this report we first describe the background of how fibers work in general and how a signal propagates through them. We are also describing the impairments that occur on the signal during the propagation and how to compensate for these. In chapter 3 we describe the necessary steps needed for designing an ASIC and also the ASIC fabrication process that is going to be used in this project. In the same chapter we also discuss how the testing circuit of the algorithm can look like. After this, we describe how the ASIC is designed, first how we decide parameter values with MATLAB and then how we design the hardware. Then we present the results from simulations and evaluations of the design and conclude the report with a conclusion and what can be done in the future regarding this project.

2
Fiber optics background

Today's society uses fiber optic communication extensively and relies heavily on it being fast and reliable. The communication network is used for transportation of important data all across the globe. To transport all this data, optical fibers are used exclusively today as they are the only known media that can handle the big data rate and at the same time have good reliability and be power efficient [5].

The usage of the Internet is growing and will probably continue to grow in the future. This means that the communication network has to be improved in order to be able to handle the increased data rate and still have the same performance. Moreover, the society expects that the performance of the network is going to be even better in the future. This means that the fiber optic communication system has to be improved constantly.

There are, however, a couple of problems with the increasing data rate in optical fibers. The major one is signal impairments caused by the fiber. When a signal is transmitted through an optical fiber the signal is distorted and the intensity reduced. The receiver has to compensate for these impairments in order to receive the correct data sent from the transmitter.

In this chapter fiber impairments are described after a short history of the optical fiber communication network. We continue with describing how the propagation of signals can be described and how the impairments can be compensated for in the receiver.

2.1 History of optical fibers

The idea that information can be transported by guiding light inside optical fibers was first thought of in the early 19th century. However, it was first a couple of decades later, with the beginning of using cladding around the fibers, which improves the characteristics considerable, that fiber optic communication began to be used in earnest [6]. The first fibers used had a very large loss (over 1000 dB/km compared to 0.2 dB/km which we have today) and was therefore not functional for longer propagations. With new and better fabrication techniques, close to the same loss that exists in the fibers today was achieved.

The first fibers used were so called multi-mode fibers. In these, the signal consists of multiple modes which travel simultaneously inside the fiber. It was then discovered that if the core of the fiber was made smaller the signal in the fiber only propagated in one mode [7]. This eliminated impairments caused by interference between the modes and the throughput and fiber distance could be increased significantly [7].

The first single-mode fiber were constructed for a wavelength of 1.3 µm and the loss was then the greatest limitation of propagation length. Silica-based fibers, which were used then and are still used today, have different loss depending on the wavelength of the signal and the lowest loss occurs for a wavelength of 1.5 µm. In order to get lower loss in the fiber the operating wavelength was switched from 1.3 to 1.5 µm enabling longer fiber spans. However, with longer spans another effect, chromatic dispersion (CD), became significant and also the limiting factor of the possible fiber length [7]. CD depends on the wavelength of the fiber and the silica-based fibers do not have any dispersion in the 1.3 µm band but it is significant in the 1.5 µm band [7]. Even with the new dispersion impairment the fibers could be made longer with the low-loss fibers with the same error rate on the output signal than when 1.3 µm was used.

The new low-loss fibers led to the development of a new field: nonlinear optics. The fibers were used to demonstrate a number of new different nonlinear impairments such as simulated Raman scattering, Brillouin scattering and self-phase modulation [8]; these effects are described in more detail in section 2.3.2.

2.2 Fiber geometry and fabrication

There are different types of optical fibers and they are all constructed in slightly different ways. Common for all fiber types is that they consist of a core surrounded by a cladding, as shown in fig. 2.1. The refractive index of the core has to be higher than the refractive index of the cladding and the difference causes the signal to be totally reflected at the core-cladding boundary, resulting in little loss of the signal.

There are two important fiber characteristic parameters used for describing the properties of a fiber [6]. One is the core-cladding index described as

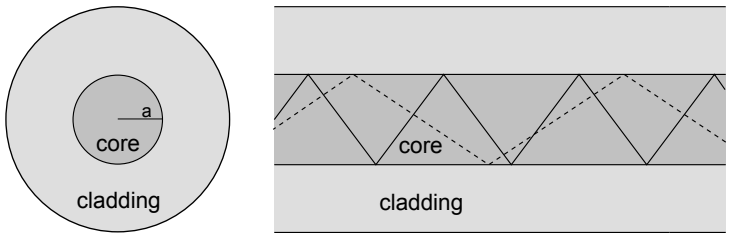
$$\Delta = \frac{n_c - n_{cl}}{n_c}, \tag{2.1}$$

where n_c is the refractive index in the core and n_{cl} is the refractive index in the cladding. This equation is used for determining the loss of the signal when it is reflected at the core-cladding interface and also the loss of the entire fiber. The other parameter is the V parameter defined as

$$V = k_0 a \left(n_c^2 - n_{cl}^2\right)^{1/2}, \tag{2.2}$$

where a is the radius of the core, $k_0 = 2\pi/\lambda$, and λ is the optical wavelength. Depending on the value of V the fiber supports different amounts of modes. If $V < 2.405$ [6] the fiber is a single-mode fiber and only supports one mode, otherwise it is a multi-mode fiber.

From (2.2) we can see that the core area determines if the fiber supports one or multiple modes. In multi-mode fibers the signal consists of multiple different light rays which take different paths inside the fiber due to a difference in reflectance angle to the cladding as shown in fig. 2.1(b). The length of each path is different and results in that different modes travel at slightly different speed and will arrive at slightly different times to the receiver. Single-mode fibers only support one mode as the



(a) Cross section (b) Longitudinal view

Figure 2.1: Geometrical model of a fiber and how different modes propagate in it. The propagation of two different modes is shown in (b), the dashed line is mode 1 and the solid line mode 2.

core is smaller than for multi-mode fibers and $V < 2.4$. For optical communication systems single mode fibers are used exclusively because the lack of mode interference [9].

2.3 Optical fiber impairments

Optical fibers used are not perfect and generate impairments and loss on the transmitted signal. The most important impairments limiting the transmission length of a signal in fibers are loss, chromatic dispersion and nonlinear effects.

2.3.1 Fiber loss

When a signal propagates in an optical fiber the intensity of the pulse is decreased due to loss in the fiber. How much the power is reduced depends on the length of the fiber, the attenuation constant, α (which depends on (2.1)), and the wavelength of the signal. The silica-fibers used in today's communication networks have a loss spectrum as shown in fig. 2.2 [6]. The lowest loss occurs at a wavelength of 1.55 µm and is considerably higher for other wavelengths. This is why a wavelength of 1.55 µm is used for the communication system today.

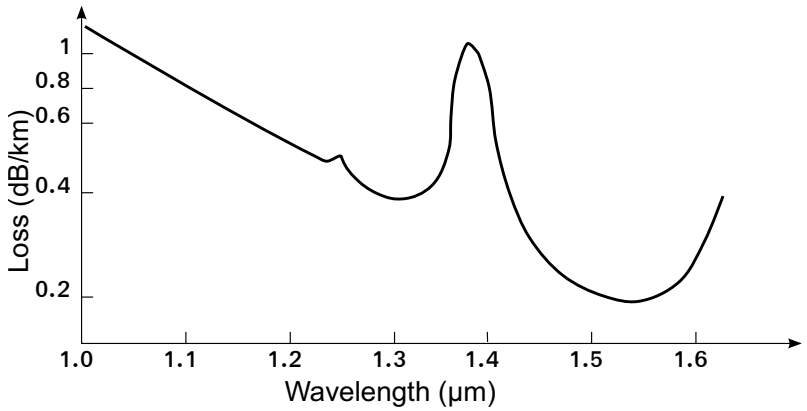


Figure 2.2: The loss spectrum of a single mode silica-fiber.

The loss in a fiber depends on several different factors but material absorption and Rayleigh scattering are most important [6]. Silica-glass has very little absorption in the wavelength region 0.5 to 2 µm but a small amount of impurities can lead to significant increase of the absorption. The peaks around 1.4 µm and 1.25 µm in fig. 2.2 are the result of OH-ions inside the silica-glass [6]. In order to minimize the impurities special precautions are taken during fabrication of the fibers to ensure that the OH-ions are less than one in one hundred millions [10].

Rayleigh scattering is caused by density fluctuations inside the fiber and cause the light to scatter in different directions in the fiber [6]. The scattering is biggest for small wavelengths and have a dependency of λ^{-4}, where λ is the wavelength.

There are also other factors contributing to the loss in the fiber. Some of these are bends in the fiber and scattering of light in the interface between core and cladding. The loss in the communication network used today is a bit larger than 0.2 dB/km for a wavelength of around 1.55 µm [6].

2.3.2 Nonlinearities

There are a number of nonlinear effects in optical fibers that are due to the intensity dependence of the refractive index or inelastic scattering of phonons [6]. The intensity dependence of the refractive index causes, for example, self-phase modulation (SPM) and cross-phase modulation (XPM) effects. Common for SPM and XPM is that both effects affect only the phase of the signal while other nonlinear effects transfer power between the channels in the optical fibers. SPM is a phase-shift effect of the signal due to that the part of the pulse with high intensity encounters a higher refractive index than the low-intensity parts of the pulse. XPM also induces a phase shift on the signal, but the shift is induced from the optical field generated by another pulse also propagating in the fiber.

At high intensity, inelastic scattering can induce effects such as stimulated Raman scattering (SRS) or stimulated Brillouin scattering (SBS) [11]. All stimulated inelastic scattering effects have in common that energy is transferred from the optical field to the nonlinear medium. A photon is annihilated into another photon with lower frequency and a phonon with the energy and momentum difference of the two photons. The difference between SRS and SBS is that Raman scattering generates incoherent optical phonons and Brillouin generates coherent acoustic phonons which induces an acoustic wave in the fiber [11].

2.3.3 Chromatic dispersion

Chromatic dispersion (CD) is a significant cause of distortion in long-distance optic communications. CD occurs because the medium response depends on the optical frequency when an electromagnetic wave interacts with the electrons in the medium. As the refractive index in the fiber depends on the optical frequency the CD effect is amplified inside the fiber.

Every pulse in a fiber has multiple different wavelength components and each component has slightly different refractive indexes and travels with slightly different speeds. The pulse will therefore be broadened and lose amplitude when travelling

inside the fiber as illustrated in fig. 2.3. This is a problem for short optical pulses close to each other as they will merge together under long distances.

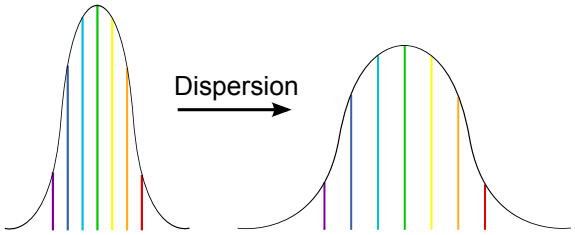


Figure 2.3: The effect of pulse broadening from chromatic dispersion. The pulse to the left is the input pulses to the fiber and the broadened pulse to the right is after propagation through the fiber.

To describe the dispersion mathematically a dispersion coefficient, D, is used and can be calculated from:

$$D = \frac{\partial \beta_1}{\partial \lambda} = \frac{-2\pi c}{\lambda^2}\beta_2 \tag{2.3}$$

where λ is the wavelength of the signal, β_1 and β_2 are the first and second deviatives of the mode-propagation constant β. The first derivative is [6]

$$\beta_1 = \frac{\partial \beta}{\partial \omega} = \frac{1}{v_g} = \frac{n_g}{c} = \frac{1}{c}\left(n + \omega\frac{\partial n}{\partial \omega}\right) \tag{2.4}$$

and the second is

$$\beta_2 = \frac{\partial^2 \beta}{\partial \omega^2} = \frac{1}{c}\left(2\frac{\partial n}{\partial \omega} + \omega\frac{\partial^2 n}{\partial \omega^2}\right). \tag{2.5}$$

In these two equations n_g is the group index and v_g is the group velocity with which the envelope of a pulse moves forward. β_2 is the group-velocity dispersion parameter and describes the dispersion of the group velocity.

As is shown in (2.3) the dispersion parameter depends on the wavelength λ. What is interesting is that, as seen in fig. 2.4, at a specific wavelength the dispersion parameter is zero. This wavelength is called the zero-dispersion wavelength λ_D [6] and at this wavelength the dispersive effect is minimal but not completely gone. D depends on the design of the fiber and by choosing the core size and core-cladding index difference, λ_D can be increased to 1.55 µm where the loss in the fiber is minimum. However, if λ is close to λ_D, there are additional higher order dispersion effects that need to be taken into account, for example the third order dispersion parameter β_3.

2.4 Fiber propagation

The propagation of light in an optical fiber and the impairments on the propagating signal can be described with the nonlinear Schrödinger equation (NLSE)

$$\frac{\partial A}{\partial z} = \left(-\frac{j\beta_2}{2}\frac{\partial^2}{\partial t^2} - \frac{\alpha}{2}\right)A + j\gamma|A|^2 A, \tag{2.6}$$

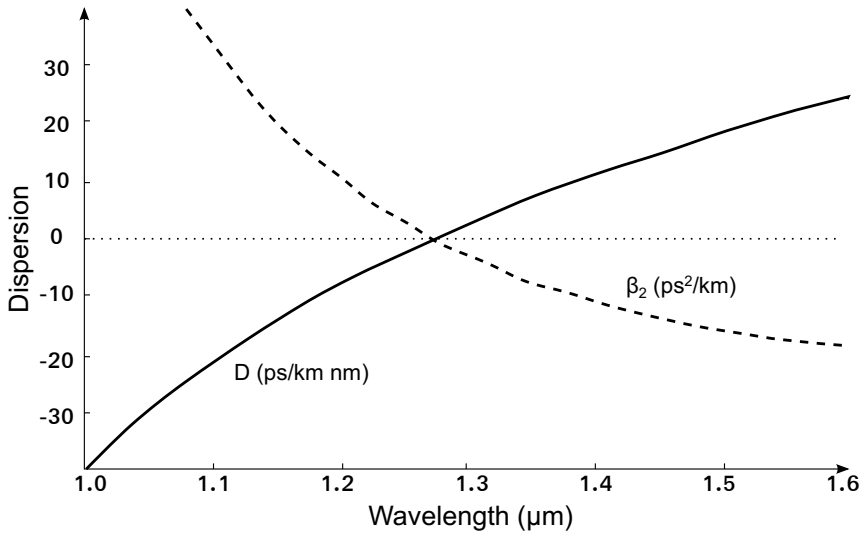


Figure 2.4: Variations of D and β_2 with wavelength for a single-mode silica fiber.

where β_2 is the dispersion parameter and can be obtained from the dispersion coefficient with (2.3). γ is the nonlinear parameter, α describes how much loss there is in the fiber and A is the amplitude envelope of the signal. The NLSE has both a linear and a nonlinear part and cannot be solved with an analytical solution except for special cases. A numerical approach is therefore necessary and there are a large number of these that have been developed during the years [12]. One of the most used is the split-step Fourier method (SSFM) that divides the linear and nonlinear parts of the NLSE and calculates them separately.

2.4.1 Split-step Fourier method

In reality, dispersion and nonlinearities act together in the fiber but the SSFM assumes that the dispersive and the nonlinear effects act independently of each other over a very small distance, Δz. This simplifies the solving of the NLSE and gives a good approximate solution to the equation. Equation (2.6) is separated into one dispersion part and one nonlinear part [12] as

$$\frac{\partial A}{\partial z} = (\hat{D} + \hat{N})A \tag{2.7}$$

where

$$\hat{D} = -\frac{j\beta_2}{2}\frac{\partial^2}{\partial t^2} - \frac{\alpha}{2} \tag{2.8}$$

is linear part including the dispersion and loss and

$$\hat{N} = j\gamma|A|^2 \tag{2.9}$$

accounts for the nonlinearities. The propagation through one step of length Δz is then carried out in two parts; the first part with only dispersion and the second

with only nonlinearities. This can be mathematically written as

$$A(z + \Delta z, t) \approx exp(\Delta z \hat{D}) \cdot exp(\Delta z \hat{N}) \cdot A(z, t) \tag{2.10}$$

where $\hat{D}$ can be evaluated with Fourier transformation.

The nonlinearities are here calculated in the end of each step but the accuracy can be improved by including them in the middle of each step. Equation (2.10) is then written as [12]

$$A(z + \Delta z, t) \approx exp\left(\frac{\Delta z}{2}\hat{D}\right) \cdot exp(\Delta z \hat{N}) \cdot exp\left(\frac{\Delta z}{2}\hat{D}\right) \cdot A(z, t) \tag{2.11}$$

and is illustrated by fig. 2.5. This method is good if the step sizes are small which means that the fiber is divided into many steps. This is not always preferable as more steps increases the number of calculations that have to be made. Often the fiber is divided into separate spans followed by an amplifier and every span is divided into a number of steps compensating for dispersion and nonlinearities separately.

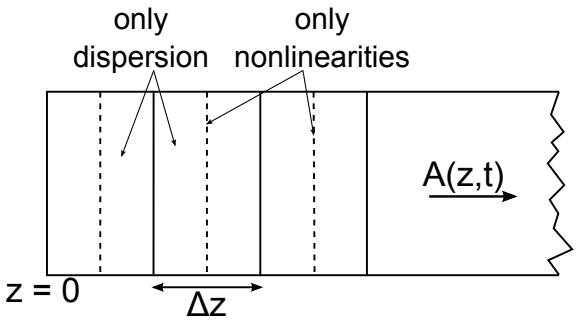


Figure 2.5: Illustration of how the SSFM divides the fiber into dispersion and nonlinear parts. The fiber is divided into steps of size Δz and the nonlinear effects are included in the middle of each step.

The accuracy of SSFM dependents on the step size Δz as well as the number of steps the fiber is divided into. The sizes used for the steps are often equally distributed but a more accurate propagation description can be achieved by varying the step size for the nonlinear part. The fiber is then divided into separate spans and each span is divided into a number of steps which have different sizes. Equation (2.9) and (2.10) shows that the nonlinearities in a fiber depends on γ, the power of the optical signal A, and the step size Δz. Since γ does not change in a certain fiber, the design of Δz and A are important regarding compensation for the nonlinearities. Since power drops down exponentially in the fiber the size of Δz can either change logarithmically in every span or be constant. By using a varying step size the same performance as for when using smaller steps with constant step size can be achieved [13]. The step size is determined by

$$z(n) = -\frac{1}{k\alpha} \cdot ln\left[\frac{1 - (N - n + 1) \cdot \delta}{1 - (N - n) \cdot \delta}\right], \quad n = 1, ..., N \tag{2.12}$$

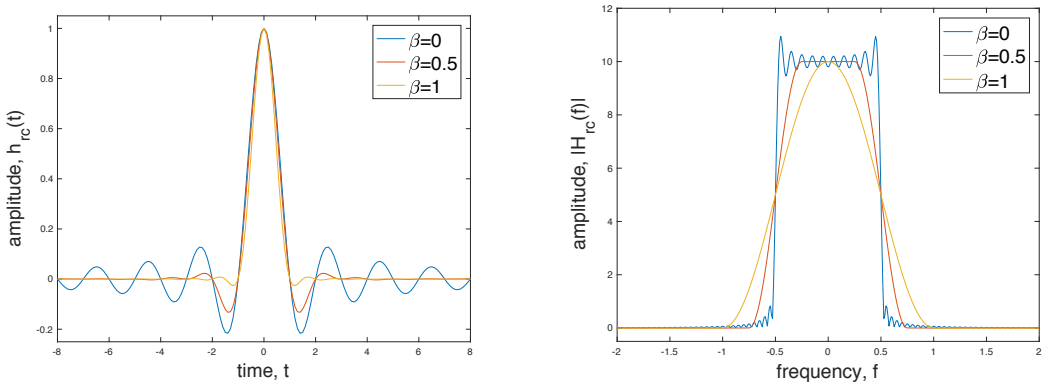
where $z(n)$ is the nth step size, k is an adjusting factor, N is number of steps in each span and n is the step index. $\delta = (1 - e^{-k\alpha L})$ where L is the length of each span and α is the fiber loss. As can be seen, when n increases, the step size, $z(n)$, also increases.

2.4.2 Root raised cosine filter

When the signal is transmitted through a bandwidth limited channel, inter-symbol interference (ISI) may occur. An ideal low pass filter can suppress the distortion but it is not possible to implement an ideal filter for a real application. Raised cosine filters are a good alternative for attenuating the distortion. The frequency response can be written as

$$H_{rc}(f) = \begin{cases} T & 0 \leq |f| \leq \frac{(1-\beta)}{2T} \\ \frac{T}{2}[1 + \cos\frac{\pi T}{\beta}(|f| - \frac{1-\beta}{2T})] & \frac{(1-\beta)}{2T} < |f| < \frac{(1+\beta)}{2T} \\ 0 & |f| \geq \frac{(1+\beta)}{2T} \end{cases} \tag{2.13}$$

where β is the roll-off factor and T is the sampling time. As can be seen in fig. 2.6(a) larger β provides more damping in the tails and therefore reduce ISI more compared to small values of β. However, large β also gives a wider bandwidth of the filter as shown in fig. 2.6(b) [14]. This means that there is a trade-off between reducing ISI and having small bandwidth.



(a) Time domain.

(b) Frequency domain.

Figure 2.6: Raised cosine pulse shaping filters.

In order to implement the raised cosine filter a root raised cosine (RRC) filter and a matching filter are often used together. This gives the same same performance of one raised cosine filter and can be simply achieved by taking the square root of the raised cosine filter function

$$H_{rrc}(f) = H_{matching}(f) = \sqrt{H_{rc}(f)}. \tag{2.14}$$

The $H_{rrc}(f)$ is then implemented as the transmitting filter and $H_{matching}(f)$ as the receiving filter [15].

2.4.3 Quadrature amplitude modulation

Quadrature amplitude modulation (QAM) is commonly used to modulate signals.For digital systems QAM combines phase-shift keying (PSK) and amplitude-shift keying (ASK). PSK and ASK modulates the phase respectively amplitude of a carrier

wave into a finite number of phases and amplitudes. The data signal are represented with a real part and an imaginary part of equal word-lengths and modulated with the carrier waves. The number of phases and amplitudes the carrier consists of is determined by the order of QAM. The most common orders used are 16 or 64 and which is used depends on how many bits per symbol the data consist of. Figure 2.7 is a constellation of a 16 order QAM, each symbol of the transmitted data consists of four bits, two bits represent the real part and the other two bits represent the imaginary part. How close the received symbols are around each digital value represents how good the data is. If the points are far away or even overlap the area belonging to other symbols the data suffer from noise.

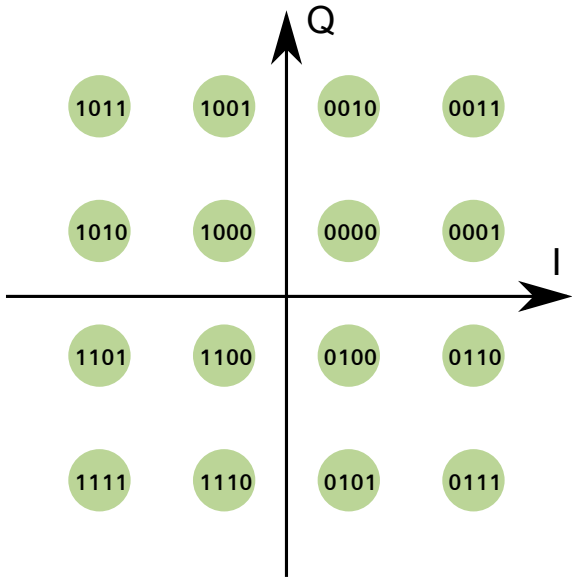


Figure 2.7: 16-QAM with example points, the green area contains the points which belong to the value in the circle.

2.4.4 Bit error rate

When a signal is transmitted through an optical fiber the signal suffers from noise, both from the fiber and from the impairments compensation in the receiver. This means that the symbol sent might not be the same as the received symbol and a measurement of how many faulty symbols transmitted are needed.

Bit error rate (BER) is used to evaluate the performance of the system. BER is defined as the ratio of the number of bits in the received signal different from the number of bits in the sent signal to the total number of bits sent [16]. BER can be used to make better choices in communication system designs by optimizing parameters to get the lowest BER. To detect or correct errors, extra (redundant) bits with data have been sent. Usually a forward error correction (FEC) unit is attached to the output of the DSP to reduce BER. A good system often has a pre-FEC BER at a 10^{-3} level.

2.5 Methods for impairments compensation

Optic fiber communication systems are limited by different impairments, such as fiber loss, nonlinearities and chromatic dispersion as mentioned above. Most methods compensating for impairments separate nonlinearities and dispersion and compensate for them with two different systems [17]. However, there are also some methods that combine nonlinearities and dispersion and compensate for them in one system, for example DBP.

2.5.1 Fiber loss compensation

As discussed in section 2.3.1 the loss of optical fibers used today is around 0.2 dB/km. This results in a loss of more than 20 dB for a 100 km long fiber. Optical signals with this attenuation are weak and it is therefore necessary to compensate for the loss generated in the fiber. To compensate for the loss, periodic optical amplifiers are often used, the fiber is divided into spans of 100 km and an amplifier is added after every span [18].

A common amplifier is the erbium-doped fiber amplifier (EDFA) which uses short erbium-doped fibers (EDF) of around 10 m [18], for compensating for fiber losses for around 100 km. As explained in section 2.3.1, signals with 1.55 µm wavelength have the lowest fiber losses and light of this wavelength is therefore used in communication systems. In EDFs the silica in the core is doped with erbium ions (Er^3+) which have an excited state at 1.53 µm [19]. The signal to be amplified is mixed with a laser beam in the EDFA and the signal is attenuated by exciting Er^{3+} ions in the fiber. EDFAs require very small pump power for the lasers and with only 20 m long fibers they can reach a gain of up to 40 dB [18].

Amplifier noise is a key factor that affects the performance of the amplifier and is caused by spontaneous emission during the amplification. The performance of the amplifier is evaluated by using an amplifier noise figure, F_n, [18] and can be written as

$$F_n = \frac{(SNR)_{in}}{(SNR)_{out}}, \tag{2.15}$$

where F_n is measured in decibel. SNR_{in} and SNR_{out} are the signal-to-noise ratios for the input and output of the EDFA.

2.5.2 Chromatic dispersion compensation

As mentioned in section 2.3.3, chromatic dispersion (CD) is a limiting impairment for optical communication systems. When periodic amplifiers are implemented together with longer propagation distances the dispersive impairments are increased. There are two main ways of compensating for the dispersion, either optically or digitally.

Optical compensation

The optical solution uses dispersion compensation fibers (DCF) and the idea is to modify the fiber design to obtain negative dispersion [20]. Often one meter of DCF can compensate for several meters of transmission fiber because the dispersion

effect in DCF is high. The drawback with optical dispersion compensation is that the compensation cannot be tuned easily without changing the entire compensation fiber [21]. Another drawback is that the DCF introduces latency of the signal which is unnecessary and can be avoided by using other compensation techniques.

Digital compensation

CD can be modeled using DSP with an all-pass FIR filter and by reversing the signs of the filter it can be used to compensate for dispersion. One filter to use for dispersion compensation is the Savory filter. The transfer function of this filter is [22]

$$H_{CD} = e^{jM(\omega T)^2} \tag{2.16}$$

where $\omega T \in [-\pi, \pi]$ is the digital frequency, T is the sampling interval and

$$M = \frac{D\lambda^2 L}{4\pi c T^2}. \tag{2.17}$$

Here D is the dispersion coefficient, λ is the wavelength of the light, L is the length of the fiber and c is the speed of light. The impulse response can be derived from the frequency response by using Fourier transform and is

$$h_n^{DS} = \sqrt{\frac{j}{4M\pi}} e^{-\frac{n^2}{4M}} \tag{2.18}$$

where n denotes the filter tap index. To avoid aliasing this index has to be lower than half the number of maximal filter taps as

$$-\frac{N_{max}-1}{2} \leq n \leq \frac{N_{max}-1}{2}. \tag{2.19}$$

The maximal number of filter taps is

$$N_{max} = 2\lfloor 2\pi |M| \rfloor + 1 \tag{2.20}$$

and ensures that the angular frequency of the impulse response never exceeds the Nyquist frequency in order to avoid aliasing [22].

By implementing this filter with the right parameters, CD in the fiber can be compensated for by sending the signal from the fiber through the filter. It is also relatively easy to change the parameters and design a different filter for a different type of fiber. However, this type of filter compensates for CD over the entire bandwidth and requires a large number of filter taps. A better filter to use for hardware implementation is a least-square constrained optimization (LS-CO) filter where the filter taps are derived in closed form. This type of filter is described by Sheikh et.al. [22].

2.5.3 Digital back propagation (DBP)

As mentioned above there are some compensation techniques that can compensate for both dispersion and nonlinearities at the same time. Digital back propagation

(DBP) is one of those techniques. The impairments in the fiber are compensated for by sending the received signal through a virtual fiber with the opposite sign of the fiber parameters (β and γ) to the transmission fiber. This method compensates for both dispersion and nonlinearities effectively. The virtual fiber is modeled by NLSE for calculating the signal propagation. This means that the NLSE has to be solved in order to be able to compensate for the impairments with this method.

Mitigation of fiber nonlinearities in general requires heavy computing and, therefore, is not straight forward to implement in hardware [5]. The same holds for using the NLSE to solve the fiber propagation, it is not very suitable for hardware implementation.

It has, however, been shown that DBP can be implemented in time-domain (TD) with close to the same performance as that of frequency-domain DBP [1]. The equation for TD-DBP can be written as [1]

$$A(z + \Delta z, t) = [A(z, t) * h_{CDC}(\Delta z)] \cdot exp\left(\frac{\alpha \Delta z}{2}\right) \cdot exp(-j \Delta z \gamma |A|^2) \tag{2.21}$$

for a single step where $h_{CDC}(\Delta z)$ is the impulse response of a filter capable of compensating for CD. The second term compensates for the loss and the last term for the nonlinearities in the fiber.

3

ASIC technical background

The design and fabrication of an ASIC is a complex process and involves a number of different steps. First the ASIC has to be designed and the functionality verified. Then the layout of it has to be designed and also verified to meet all constraints so that the ASIC can be fabricated with a relatively high yield.

In this chapter we will go through how an ASIC is designed and how it can be fabricated. We are also discussing how memories in the ASIC can be designed and how the design can be evaluated and tested.

3.1 Design flow

The design of an ASIC is done in hardware description languages such as Verilog or VHDL. Often a big design is composed of separate components that are then connected together in a top design.

In order to verify the functional behavior of the hardware description a testbench is used. The testbench generates signals for the input ports on the design and the output can then be verified with different simulation programs like QuestaSim [2].

In order to create a functioning design without physical or logical errors, certain steps in the implementation process have to be done. The main steps of the process are shown in fig. 3.1.

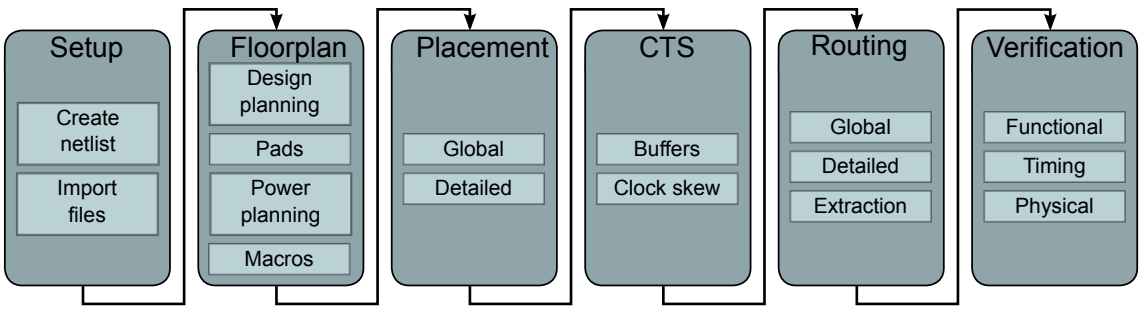


Figure 3.1: Design flow of an ASIC design.

3.1.1 Setup

The first step is, from the hardware description of the design, to create a netlist. A netlist describes the cells used in a design and the connections between them. In

order to decide what type of cells that are going to be used in the netlist libraries describing these cells have to be set up.

Every standard (STD) cell in the design consists of a number of different gates which, in turn, consists of a number of different transistors. The huge amount of transistors in ASIC designs today makes the power consumption and speed for each single transistor important and there is a trade-off between the two factors. High threshold voltage (V_t) gives low leakage power but also high delay for the switching. Similarly, low V_t causes high leakage power and fast switching. The area of a design is today not limited by the STD cell size but by the routing area [23]. It is therefore important to design the STD cells so that they are well suited to the place and route tool instead of concentrating on minimizing the size of the cells in order to minimize the area.

3.1.2 Floorplanning

Floorplanning is the first step in the physical design flow. In this step the size of the ASIC is determined and major blocks (macros) of the design are assigned a location. If the design requires pads, these are placed on the edge of the chip in this step. Floorplanning is significant for the timing of the design. If the macros are placed in a bad way the timing of the design might not be met. Also, if the blocks are placed badly the design might not be routable.

The first step in floorplanning the design is to import the technology files, library files, the netlist, and design constraints into the design tool. After importing and verifying the files the core area, standard cell rows, and I/O pads are created.

Design planning

There are two different ways an ASIC design can be implemented in; hierarchical or flat style [23]. The hierarchical style is mostly used for large designs and the flat style for small or medium sized designs.

The hierarchical style divides the design into several sub-blocks and is therefore suited for very large designs. This style is normally used when the ASIC is designed with subcircuits which are individually placed and power routed. However, if the components forming the critical path is placed in different locations this can extend the critical path and degrade the performance which is a drawback for this method. The flat style utilizes the area better than the hierarchical style as the design is not divided into separate blocks. Drawbacks with this style are that it requires a big design tool memory and the run time increase rapidly with design size [23].

Pad placement

The I/O pads are the external connections to the ASIC and consist of three types; power, ground and signal. The pads need to have sufficient power and ground connections and be placed properly in order to avoid electromigration and current-switching noise [23]. Electromigration is the transfer of metal due to an electric field or a current [24] and can cause short circuits or brakes in the channels. Switching noise occurs when the output of the ASIC shifts between two states can cause

internal power fluctuations in the ASIC and can lead to logic errors. These effects can be eliminated by placing the pads properly and have the right number of power and ground connections.

Power planning

The next step of designing the ASIC is to create the power and ground structures. Around the core of the ASIC two or more rings are placed, one for power and one for ground. The power and ground to the STD cells in the core are constructed by horizontal and/or vertical metal stripes as can be seen in fig. 3.2. The number of power stripes is dependent on the total power consumption of the ASIC.

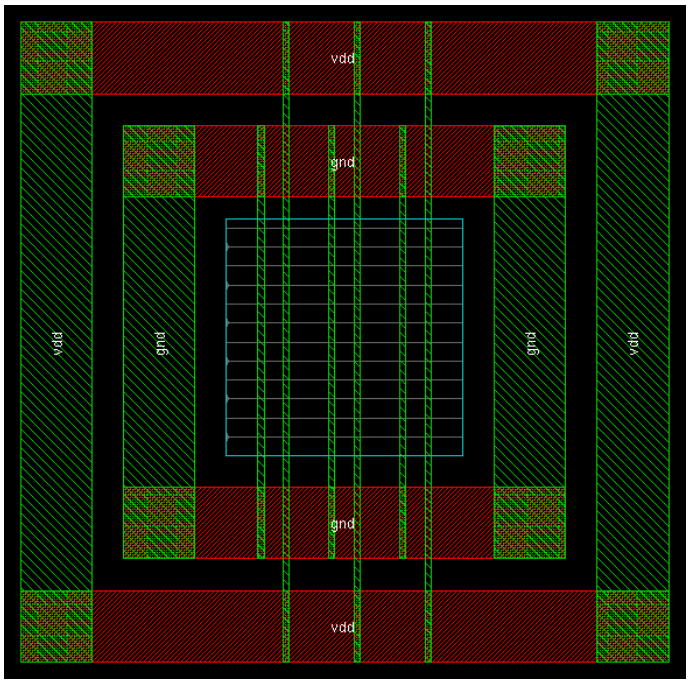


Figure 3.2: Ring and core power and ground planning. The power and ground ring is located outside the core of the ASIC and vertical stripes distribute power and ground to the STD cells inside the core.

Macro placement

The last step in the floorplanning is the placement of macros. In case a hierarchically implementation style is used macros could be subcircuits or memories. The macros can be placed either manually, which is good if the connections between the blocks are known and the number of macros are small, or automatically.

3.1.3 Placement

After floorplanning the STD cells are placed in the core of the ASIC as shown in fig. 3.3. The cells are placed in defined rows and should be placed so that the design can be routed efficiently. The cell placement is important as it decides the area

usage, routing congestion and timing behavior of the design. First global placement is done which minimizes the wire lengths and then detailed placement is performed which ensures that all design constraints are met [23].

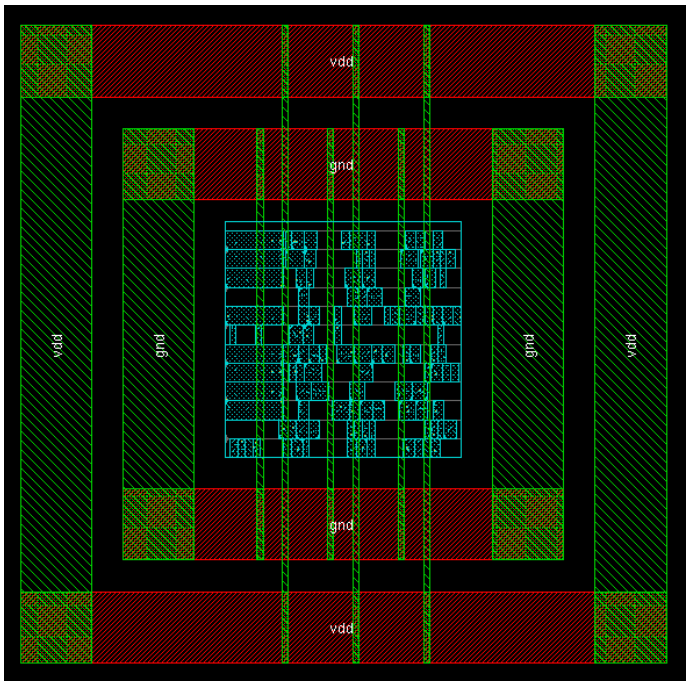


Figure 3.3: Placement of cells inside the core of an ASIC.

The STD cells are partitioned in the design which means that certain cells that should be placed close are grouped together in regions. The regions can be either soft or hard. Soft regions specifies an area where the cells can be placed outside while hard regions specifies an area where the cells must be placed inside. After partitioning of the design is done the global placement algorithm distributes STD cells uniformly in the ASIC core and the cells are placed in such a way as to minimize the wire length for the connections.

Once the STD cells are placed globally a detailed placing algorithm is executed. This algorithm refine the cell placement based on congestion, timing or power requirements depending on teh settings from the user.

3.1.4 Clock tree synthesis

During the clock tree synthesis step buffers are inserted along the clock path in order to have the same delay for all clock inputs to avoid clock skew. Skew is the difference in propagation delay of two leafs of the clock tree (a leaf is the last level of the clock tree). If the circuit has a positive clock skew the transmitting registers receives the clock pulse earlier than the receiving registers and this could increase the performance of the design. Negative clock skew on the other hand decrease the performance as the receiving registers get the clock pulse before the transmitting registers and this increase the hold time in the registers [23].

Another important aspect of designing the clock tree is to consider the power consumption. As the clock operates at the highest frequency in the design and often

have high capacitive load the clock network uses 30 % [23] of the total dynamic power in the circuit. The clock tree design is therefore, in addition to performance, important for power consumption of the design.

3.1.5 Routing of nets

The routing step includes routing the nets and extracting the parasitic parameters of the wires in order to be able to do a realistic timing analysis of the circuit. Similar to the placement, the routing is divided in a global and a detailed step. An example of a routed design is shown in fig. 3.4.

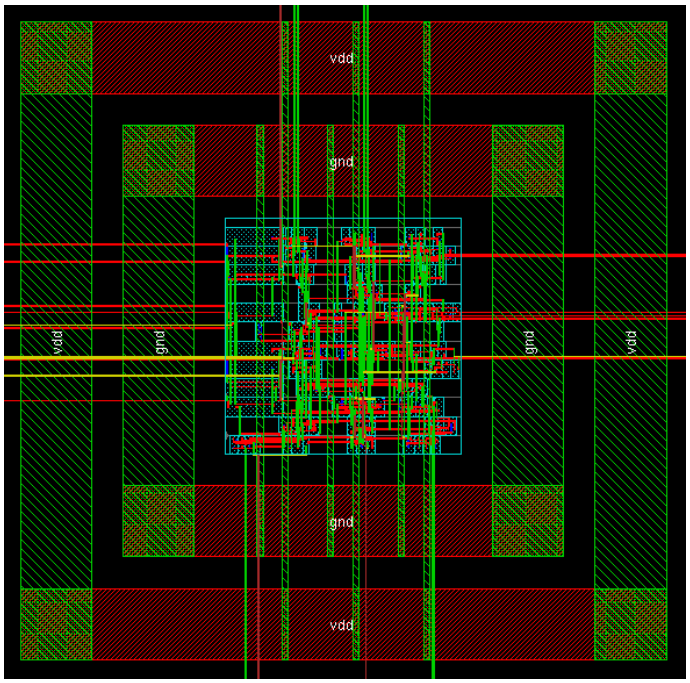


Figure 3.4: An example of a routed ASIC design. The connections going outside the ASIC is the designs external inputs and outputs and should be connected to I/O pads.

The global routing algorithm generate a non-restricted route for all the wires in the design. The lengths of the wires are also calculated and the parasitics extracted. The global routing is done by first dividing the design into separate regions and calculate the routing density for each region. Then the routing is done with the help of a graph that models the connection network.

The detailed routing places the wires inside the regions defined by the global routing and creates the physical connections between the ports. The detailed routing can be grid-based, gridless-based or subgrid-based. The grid-based routing uses a grid composed of evenly spaced routing tracks that the wires must follow. This method is efficient but all the ports of the cells need to be placed on the grid or design rule violations occurs. Gridless routing is not restricted by any grid but uses the entire routing area. This method is very slow and complicated but can use different widths and spacing between the wires. Subgrid routing is a combination

of grid-based and grid-less routing. The method sees the grid as a guideline and do not strictly follows it.

3.1.6 Verification

The verification of the design is the final step of the design flow. This step verifies the functionality of the design, ensures that timing constraints are met and checks that the design follows all design rules.

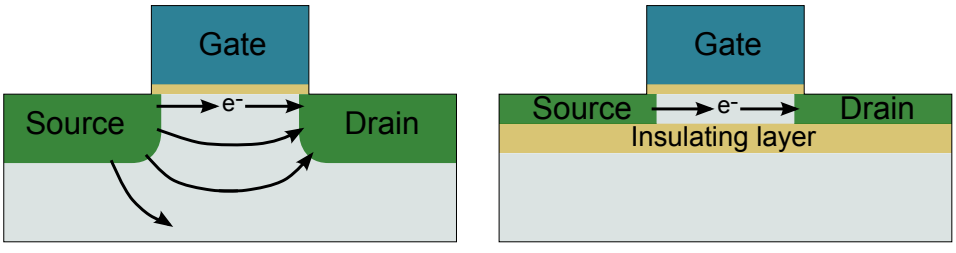
Functional verification is performed on the netlist and this can be done by comparing the gate level description or the netlist with the behavioral description of the design.

In order to confirm that the design meets the timing constraints timing verification is needed. This is done by calculating the worst case delay from the gate and path delays. The gate delays for each gate is provided by the technology files but the path delays needs to be calculated from the parasitic effects in the design. Extraction of the parasitic parameters means to calculate the routed net resistances and capacitances so these can be used for delay calculations. The parasitic extraction is done on each net and takes both the net's own structure and the proximity to other nets into consideration.

Physical verification ensures that all process rules are followed by the design. The process rules are provided by the semiconductor foundries and are provided so that the design can be manufactured without errors and with high yield. The rules become more complex as density of the transistors increases. As the physical verification is so complex it is almost fully automated and includes layout verses schematic (LVS), design rule check (DRC) and electrical rule check (ERC).

3.2 Fabrication technologies

Today Chalmers uses STMicroelectronics (ST) to fabricate their 28 nm ASICs. ST uses a fully depleted silicon on insulator (FD-SOI) technique in order to be able to fabricate small transistors and at the same time have low leakage power and a not too complex manufacturing technique.



(a) Bulk (b) FD-SOI

Figure 3.5: Comparison between a bulk transistor and one in FD-SOI.

Compared to bulk transistors, FD-SOI transistors are constructed with a thin buried insulating oxide layer on the base silicon as seen in fig. 3.5, with the transistor

channel formed on top of the insulating layer. This gives much better electrostatic characteristics compared to bulk technology [25] and the parasitic capacitance and the leakage power are reduced.

By controlling the voltage of the base silicon beneath the channel the behavior of the transistor can be controlled. When the polarization is positive this gives the transistor a lower switching time.

3.3 Verification circuit

In order to verify the behavior of the TD-DBP block a testing circuit has to be developed. The goal with the testing circuit is to generate test vectors that simulate the fiber propagation as realistically as possible. There are a number of different approaches one can take in order to do this.

Simulate a realistic input signal

The best approach, from a verification point of view, is to design a block simulating the fiber propagation from an input signal. This is, however, very complex and the best method here is to use a DBP block with inverted parameters. Then no new block has to be designed as the same block can be used twice. On the other hand, if there is something wrong with the DBP block this will not be detected as the same fault exist when generating the simulated signal through the fiber. This method is therefore not very good as it will not find any errors in the DBP block. The block simulating the fiber should also have a large number of steps in order to create a good simulation of the fiber. This will however, contribute to the block simulating the fiber will be occupy more area than the DBP block.

Instead of using an inverse DBP function there are other functions that could be used. These are, however, not very well suited for hardware implementation which make them much more complex to implement than the DBP function.

One symbol

One straightforward method to verify the performance is to store one symbol in a memory on the ASIC and send this to the DBP block in a loop. The result of the DBP could then be compared in a block calculating the BER. In order to calculate the BER the expected output from the DBP also has to be stored in the memory. The drawbacks with this method is that it only tests one case for the input signal and if there are any cases in which the DBP does not work very well those will not be found. The positive thing is that the memory block of the ASIC can be very small as it only needs to store one or two symbols depending on if the BER calculations are performed on- or off-chip..

Store test vectors

Another method is, instead of only storing only one symbol in the memory, to store a number of different symbols. By using multiple symbols more cases are tested and errors and corner cases can be found compared to when only one symbol is used.

In this case, similar to when one symbol is used, the expected output can also be stored and a block which verifies the output from the DBP can be used. Even if this method is better than using one symbol the number of symbols that can be stored in the ASIC is still limited because of size limitations, and not all different cases can be tested.

Input vectors from test equipment

The last approach is to use external test equipment to generate the input signal to the DBP block. By using this method several different test vectors can be tested without using a memory in the ASIC. This method also ensures that the input signal is a realistic output from the fiber as a real fiber can be used for generating the input signal. However, as the fibers the DBP can compensate for are several kilometers long it is impractical to test which such a long fiber. Another problem is that the test equipment is slower than the ASIC and by using this method the ASIC cannot be tested at full speed.

4
Design

The designing of the ASIC is divided into two parts. First a MATLAB simulation has to be done where the best trade-off setup for the TD-DBP method is investigated and different values on certain parameters are analyzed. The second part is to design the VHDL code for the ASIC, this includes implementing a test-circuit which extends the TD-DBP implementation to multiple span operation. We want to reach a BER level of 10^{-3} when 40 spans and have an area of no more than $1\,\mathrm{mm}^2$ for the ASIC.

In this chapter we first describe how the MATLAB simulations are done and how the DBP method can be constructed. Then we describe how the hardware for the DBP and testing circuit are designed.

4.1 MATLAB simulation of TD-DBP

The MATLAB simulation consists of both simulation of the light propagation through optical fibers and the simulation of the TD-DBP algorithm. This is because in order to verify that the simulation of the DBP works correctly the output signal from a fiber is needed. The output signal is therefore produced by sending random signals through a emulated fiber designed as described below.

4.1.1 Light propagation simulation

As described in section 2.4 light propagation can be emulated by the NLSE. By using SSFM (2.10) to solve the NLSE the propagation of a signal through a fiber is known. The SSFM assumes that dispersion and nonlinearity do not affect each other and can be calculated separately when light propagates over a very small distance, Δz, in the fiber. This approach is used when simulating the fiber propagation. The fiber is divided into spans of 100 km and each span is divided into 200 equally distributed steps. By dividing each span into a large number of steps the simulation of the fiber propagation becomes closer to the real behavior of the fiber. For each step the dispersion, which is calculated using FFT, and the nonlinearities are calculated separately. The dispersion coefficient, D, is set to $17\,\mathrm{ps/km\,nm}$, the wavelength of the signal is $1.55\,\mu\mathrm{m}$ and the nonlinear factor λ is $1.3\,(\mathrm{W\,km})^{-1}$. These values, together with a loss factor of $0.2\,\mathrm{dB/km}$ are used to simulate the fiber propagation.

Random vectors are used for the input to the emulated fiber. The random signals are filtered by an RRC filter with $\beta = 0.1$, and modulated by 16-QAM as shown in fig. 4.1. NLSE is then applied to emulate the fiber and EDFA is used to emulate amplifiers and adds both amplification noise and amplification to the signal for each

span. The amplifier is needed as the NLSE emulates loss of the fiber but the DBP method does not compensate for it. In other words, the EDFA is used to compensate for the losses generated in the fiber.

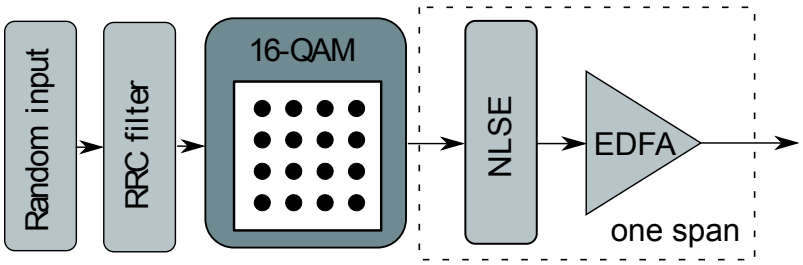


Figure 4.1: Block diagram of MATLAB emulation of light propagation in an optical fiber where each span emulates 100 km fiber propagation.

Parameters used for the propagation and the amplification are listed in table 4.1. The number of samples per symbol (SaPS) of the propagating signal is 16 and the throughput of the system is 20 GBd. The loss in each span is compensated for by an EDFA with a gain of 20 dB and 4.5 dB amplifier noise.

Table 4.1: Parameters used to simulate fiber propagation, the first six parameters are used in NLSE and the last four parameters are used in EDFA.

	Parameter	Value		
NLSE	D	17	ps/(km nm)	
	γ	1.3	1/(W km)	
	λ	1550	nm	
	α	0.2	dB/km	
	SaPS	16	-	**EDFA**
	Symbol rate	20	GBd	
	Gain	20	dB/100 km	
	NF	4.5	dB	

4.1.2 TD-DBP simulation

The transmitted signal from the fiber propagation emulated is used as input to the DBP algorithm to verify the functionality and evaluate the performance of it.

In order to implement the DBP algorithm in hardware without using FFT or IFFT, which are used to calculate the dispersion, the DBP can be used in time domain (TD) [1] and the need for FFT is withdrawn. The same approach is used as for simulating the propagation of the signal. The fiber is divided into a number of 100 km spans and each span is in this case divided into two steps instead of 200 which is used to emulate the fiber. Fewer steps leads to less computations for the cost that some performance is lost. Every step is compensated for by using the TD-DBP in (2.21) which describes the compensation needed for one step.

To compensate for the dispersion the CD compensation filter described in section 2.5.2 is used. Two parameters important for the filter are the number of filter taps and the filter coefficients. Simulations with different number of taps have been done with the conclusion that 17 taps is a good trade-off between area and performance. These simulations are described more thoroughly in section 5.1.2.

As seen in (2.21) the nonlinear impairments are complex exponential which is very complex to implement in hardware. In order to avoid using the exponential function Taylor expansions are considered [1]. What order of Taylor expansion used does not affect the performance much and first order Taylor expansion is therefore used for the design. The nonlinearity impairments are power dependent which means that the nonlinearities are larger for higher power for the signal. In each span the power is decreased logarithmically and therefore the first step should be smaller than the second step. For back propagation, which is the reverse of the propagation, bigger step size is needed in the beginning of the span and smaller in the end. To calculate step sizes (2.12) is used with the optimal value of $k = 0.4$ [26]. This means that the step size of the first step in each span is compensating for 70 km accumulated nonlinearities and the second step compensates for 30 km.

As mentioned in section 2.4.1 the accuracy can be improved by calculating the nonlinearities in the middle of each span. This means that, as seen in fig. 4.2, the first and last step of the whole fiber for the dispersion are different. The first dispersion step is compensating for 35 km and the next two dispersion steps can be combined to 50 km. For the last span, a dispersion step of 15 km is used.

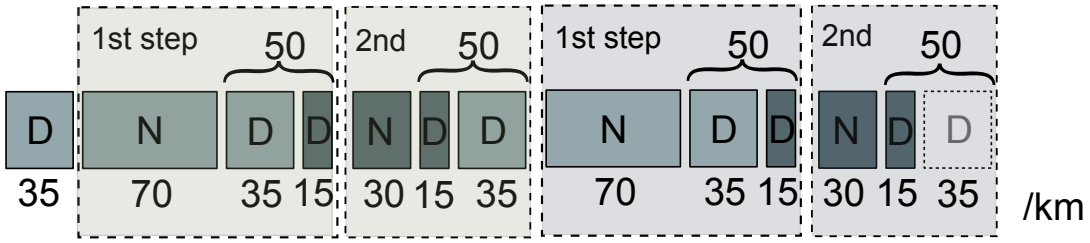


Figure 4.2: Step sizes for dispersion and nonlinearities, the light gray small blocks are for the first step and dark gray blocks are for the second step. Dashed lined blocks are the step blocks used in MATLAB simulation to reduce dispersion computation. The unit of the number under each block is in km and indicates the distance each block compensates for.

The received signal from the light propagation emulation is down-sampled from 16 SaPS to 2 SaPS to decrease the number of calculations for the DBP as seen in fig. 4.3. In order to compensate for the losses in the fiber an gain stage with a gain of 20 dB is implemented for each span. A matched filter is applied after back propagation with the same parameter as the RRC filter used for the propagation simulation, $\beta = 0.1$. After the matched filter the output symbols are compared to the input symbols to the fiber and BER is calculated to evaluate the performance of the system.

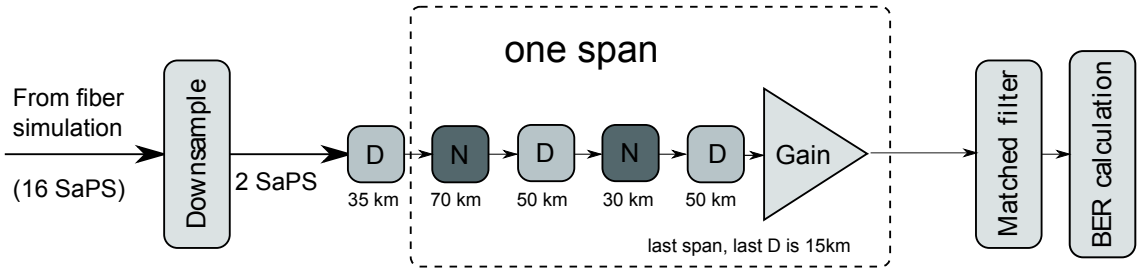


Figure 4.3: Block diagram of simulation of TD-DBP. The D blocks compensate for dispersion and the N blocks for nonlinearities.

4.2 Hardware implementation

For implementation of the TD-DBP in hardware the fiber is, similar to the MATLAB simulation, divided into separate spans where each span is capable of compensating for 100 km of accumulated fiber impairments. Only one span is implemented in hardware and this component is used multiple times on the same symbol in order to compensate for multiple spans. For storing the output signal from the DBP so that it can be used as input for the next span two memories are used. One memory stores the output of the DBP and the other provides input signals. The input symbols are sent through the first span and the output is stored in the memory. The stored output is then read and sent through the next span. These two steps are iterated for as many spans the DBP should compensate for. The output from the DBP block is read from the memory when the input symbols have gone through the span block the right number of times.

The input and output signals to the ASIC are

```
entity top_module is
    port(
                reset       : in std_logic;
                clk         : in std_logic;
                data_in     : in std_logic_vector(1 downto 0);
                span_num    : in integer range 1 to 50;
                data_out    : out std_logic_vector(1 downto 0)
    );
end entity;
```

where *data_in* and *data_out* are the input and output symbols and how these are used are described in section 4.4. The number of spans the DBP compensates for are controlled by the signal *span_num*. If this signal is set to 0 the number of spans are set to infinite for power measurements. The *reset* signal has to be set to 0 in the beginning to reset some buffers in the DBP block. When *reset* is set to 1 the DBP will start calculating.

4.2.1 TD-DBP component

Each span is built up in the same way as the MATLAB simulations with two steps in each span. The first step compensates for 70 km nonlinearities and 50 km dispersion

and the second step for 30 km nonlinearities and 50 km dispersion. In the MATLAB simulations, for the entire fiber, we have different first and last steps of 35 and 15 km for the dispersion compensating part. These steps are not implemented in hardware to make it smaller and simpler. This is acceptable as the first and last steps are small compared to the whole fiber and do not impact the performance much. The number of bits used for the input signal and the dispersion filter coefficients is chosen from the MATLAB simulations to get a good result but at the same time have a small area.

In this section, we first describe how one step is designed and then how two steps are put together. The second part in this section describes how the span implementation is used for compensating for multiple spans. The last part describes how the implementation is verified in regard to functionality.

One step

One step is designed by using two different sub blocks where one compensates for nonlinearities and the other for dispersions.

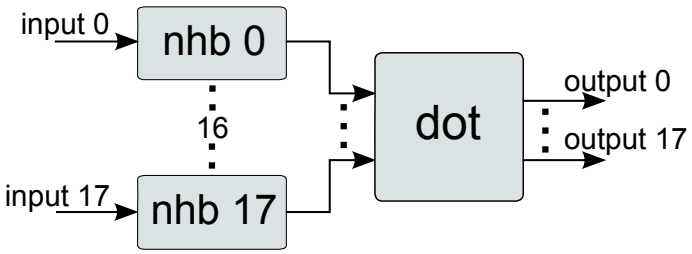


Figure 4.4: 18 `nhb` blocks are used to get 18 input symbols to the `dot` block.

The `nhb` block in fig. 4.4 compensates for the nonlinearities by multiplying the input with a nonlinearity factor. The factor depends on a *gamma* signal on the input to the `nhb` block that depends on how many kilometers nonlinearities should be compensated for and the nonlinear parameter, γ, of the fiber. As each step compensates for either 30 or 70 km the *gamma* signal have two different values. It is calculated as $\gamma \cdot length$ and as $\gamma = 1.3$ the *gamma* signal have the value of either 39 or 91.

The nonlinearity factor is calculated as $gamma{\cdot}x^2$, where x is the input signal describing one symbol consisting of a vector for the real part and a vector for the imaginary part. The factor is then right shifted and rescaled in order to get the right length of it. The output of the `nhb` block is simply the input symbol,x, multiplied by the nonlinearity factor.

The dispersion compensation block, `dot`, takes the output from the `nhb` block and sends it through a FIR filter with filter coefficients calculated as in section 4.1.2. To calculate the output from the filter both the current input signal and the previous input signal are used and these are stored for every step in an array. 17 taps are used for the filter and 18 input symbols are used at the same time in order to speed up the calculations. 18 input symbols are used as the size of the ASIC is limited and the filter is implemented in such a way that the number of input symbols have to be

larger than the number of filter taps. If we want 18 input symbols to the `dot` block we also need 18 `nhb` blocks generating the input signals to `dot` as shown in fig. 4.4. The output from the `dot` block is also 18 symbols. By having higher parallelization the speed of the `DBP` is higher but the area of the layout also increases. In the `dot` block the input is multiplied with the filter coefficients. When multiplying two vectors the word length of the product is the sum of the length of the factors. This means that the output for the `dot` block is the size of the input signals plus the size of the coefficients. If a signal of 8 bits is used and the coefficients are 9 bits the output will be 17 bits. When two steps are used the output from the `dot` block is connected to the input of the `nhb` block which means that the signal has to be requantised to the same size as the input signal.

In order to handle complex signals, signal types are created consisting of one real and one imaginary part for the input signal and the filter coefficients. The real and imaginary parts, in turn, consist of signed vectors so that signed calculations can be performed. The complex calculations needed are only a multiplication between the filter coefficients and the input signal in the `dot` block. A separate function for multiplying two different complex signals is created and used in the `dot` block. The function takes two inputs, `A` and `B`, and calculates the real and imaginary part for `A·B` separately as

$$A \cdot B = (a + bi)(c + di) = ac - bd + i(ad + bc). \tag{4.1}$$

One span

One span consists of two steps and this can be implemented in two different ways; either two step components are used where the output from the first step is connected to the input of the second step component (as seen in fig. 4.5), or one step component is used where the source of the input signals are changed continually between the input signal from the memory and the output from the step component (as seen in fig. 4.6).

For the filter coefficients we use two sets, one set for each step. There are two methods how these coefficients can be stored in the ASIC. The coefficients can be either hardcoded in the ASIC or stored in the memory of the ASIC. By hardcoding the coefficients both area and memory can be saved. Area measurements of when fixed coefficients are used versus when adjustable (stored in memory) coefficients are used are presented in section 5.2. We chose to use fixed coefficients because of the area limitations we have on the ASIC and as this also makes the ASIC slightly faster as the coefficients do not have to be read from the memory. However, with fixed coefficients the compensation parameters are fixed and can not be changed which means that only one type of fiber can be compensated for with this particular ASIC.

The easiest way to design one span is to use two step components where the output of the first is connected to the output of the second as shown in fig. 4.5. As mentioned above, the output from one step is bigger than the input and has to be requantised. This is simply done by having a resizing block between the two step components which takes the least significant bits of the input as output.

In order to be able to have adjustable coefficients and still have small area a

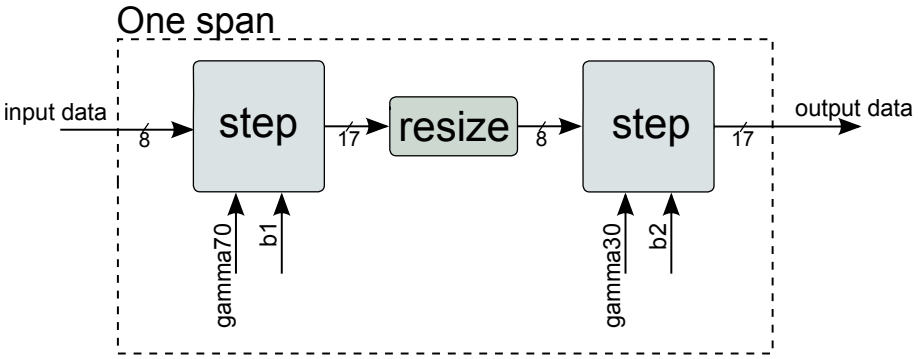


Figure 4.5: Block diagram of howone span is implemented with two step components.

single step component can be used where the input is changed continually. This implementation is twice as slow as when having two step components as every second input is taken from the output of the component. As is shown in fig. 4.6 a variable *stepNum* signal keeps track of if the first or last step is used. If the first step is used the input is taken as the input to the span component and *gamma* is chosen for 70 km and *b1* is used for the coefficients. The output from the first step is then used as the input to the second step and *gamma* is changed to 30 km and the coefficients to *b2*. The output from the span component is updated every second clock cycle and taken from the second step.

As the dot component stores the most recent and second most recent input values these have to be stored in different arrays for the first and the second step. In order to do this, one additional array is used and for the first step the first array is used and for the second step the other array used. The signal *stepNum* as input to the step component ensures that the input is stored in the right array for each step.

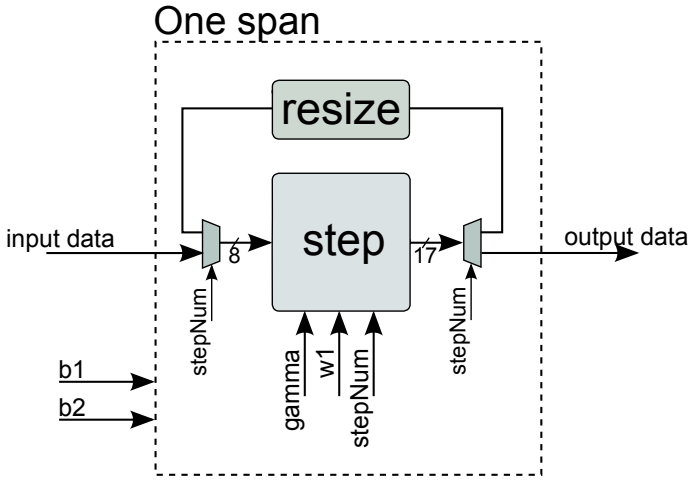


Figure 4.6: Block diagram of how one span is implemented with one step component. The *b1* and *b2* input is used if the coefficients are adjustable and stored in the memory.

The first implementation method is area ineffective as two step components are used where only one is necessary but it is also fast compared to the other method.

For the method with one component the output is updated every second clock cycle due to that every other output is the input to the step component. For our implementation we are using the first method with two step components as it is faster and, from the area measurements in section 5.2, not very much bigger.

Multiple spans

The ASIC should be able to compensate dispersion and nonlinearities for several 100 km spans which means that the input symbols have to be sent through several spans. The input symbols can be sent through the spans by two different methods.

The first method is to read one symbol from the memory and send it through the span component. The output from the span component is then sent through the span component again and this is repeated for as many times as the number of spans we want to compensate for. This implementation method means that the old input of every span needs to be stored for the dispersion compensation filter. If 40 spans are used $40 \cdot 18$ symbols need to be stored, as the parallelization level used is 18 and 18 symbols are operated on at the same time. This method takes up much area which is disadvantageous as area is a limiting factor in our project.

For the other method all the symbols stored in the memory are sent through the first span and the output is stored in the memory. This means that the order of the inputs to the dispersion compensation filter will be correct and only one old value has to be stored. However, with this method the output of the `DBP` block needs to be stored in the memory. This leads to that only half the memory can be used to store input symbols and the other half is used to store the output. This is the method we are using as it saves area and the memory used in the design has such a capacity that enough input symbols can be stored in half of it that a good verification of the functionality can be achieved.

4.2.2 Verification of TD-DBP component

In order to verify that the VHDL code works correctly a MATLAB file generating two files are used. One file is the input symbols and the other, the expected output symbols. The input file is then used in the testbench of the `DBP`. The testbench also generates an output file and the expected output from the MATLAB file can be compared to the real output from the VHDL code.

The input data used are random numbers of fixed type with 8 bits which are stored in a file that the testbench of the `DBP` uses as input. The number of input symbols used is around 100 000 complex numbers in the range -100 to 100 for both the real and imaginary part.

The expected output file is generated by simulating the TD-DBP with MATLAB. The simulation is built up in the same way the VHDL code is constructed with a fixed point implementation. First the signal is sent through one span consisting of two steps and a resizing function. The output from the span is resized again to get the right number of bits on the output signal. Each step consists of first a nonlinear compensation part and then a dispersion compensation part like the VHDL code.

4.3 Memories

The memories used in the ASIC are needed to store both the input and the output of the DBP block. The memories are generated by STMicroelectronics as it is they who will fabricate the ASIC. We have decided to use four memories and as it is not necessary to read and write to the memories at the same time single port memories are used. The BER should be around 10^{-3} and this means that several thousands different input symbols are needed. The memories chosen for our design can store 4096 words with a word-length of 72 bits. This means that each memory has approximately 288 000 bits capacity.

As shown in fig. 4.8, four memories are used for the design. Two memories are used to store the real part of the signal and the other two to store the imaginary part. All the memories have the same address which is increased every clock cycle, from 0 to 4095. In the case when the input signal to the memories is coming from an external source, the address is incremented every 72 clock cycle as the input are two bits for every clock cycle.

In fig. 4.7, the memories named A are the first two memories and B are the second two memories. The external input is first stored in memory A and then DBP reads from A while writing the output to memory B. As there is a delay between reading the first data in the memories and receiving the first valid data from the DBP, the memories will store the output after a certain delay. The output after the first span is stored in B but the first addresses, until a parameter *delay*, are empty. After spanNum of spans, the output of the DBP block is written from $N \cdot delay$ to the end of two memories and from 0 to $N \cdot delay$ of the other two memories.

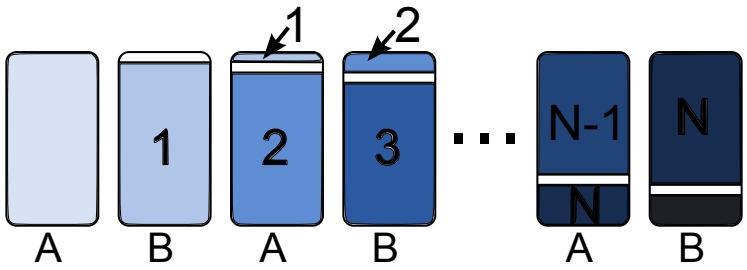


Figure 4.7: How data is stored in the memories and how the delay affects the location of the input and output data. A is the first two memories and B the second two.

4.4 ASIC design

The goal with the test circuit is to verify the functionality of the DBP component in the ASIC. The testing circuit is designed by using one of the methods mentioned in section 3.3, to store a large number of test vectors and use these as inputs. Either the BER calculations are done on chip or off. In our final design we have chosen to not include any BER calculation circuit on chip as this would occupy a significant amount of area and requires a full receiver and a demodulator. The subblocks for the ASIC and how they are connected are shown in fig. 4.8.

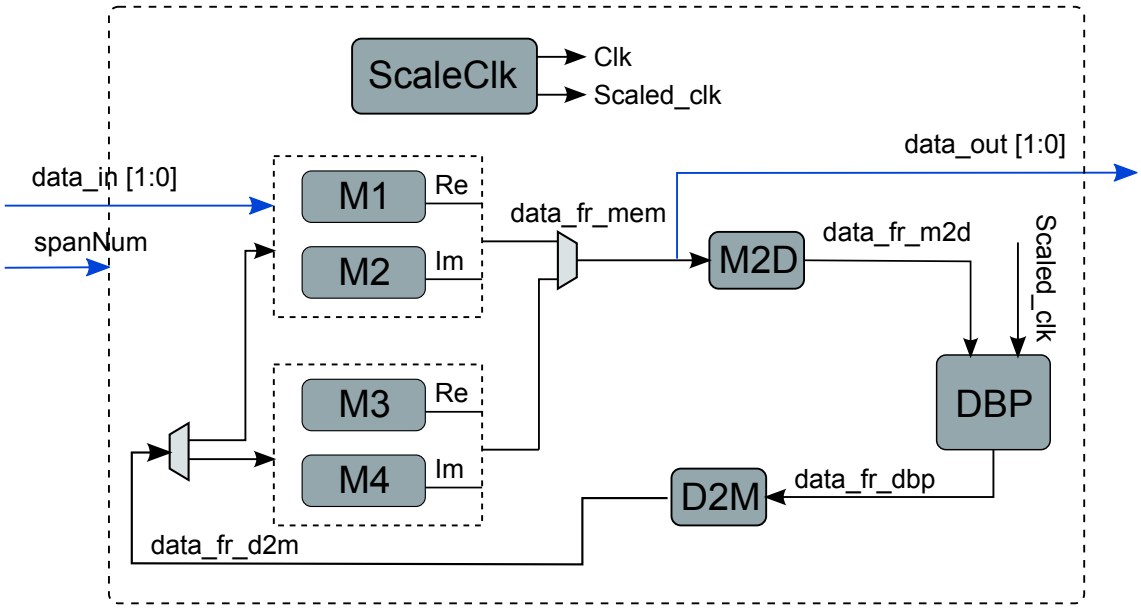


Figure 4.8: Subblocks and connections between these for the ASIC design.

As the input to the DBP comes from the memories, the input data first need to be stored. Each memory has the capacity to store 36 000 random input symbols and these symbols are generated with MATLAB. From the MATLAB simulations we have an approximate value on the BER of 10^{-3} to 10^{-2} which means that 36 000 symbols will give approximately 36 to 360 incorrect symbols. As the size of the memories are limited four separate SRAMs are used as shown in fig. 4.8.

For the first span, data first has to be stored in two memories that provides input to the DBP block. When the memories are filled with 36 000 symbols the data are sent to the DBP block every second clock cycle. The output of the DBP will be stored in the other two memories and when all the input symbols in the first memories have been processed by the DBP the signal has been compensated for one span. When all the data have been read from the first two memories the memories are switched so that the ones sending data are storing data and the signal begins being compensated for the second span. After the signal has been compensated for the right amount of spans, the output is read from the memories as an output signal from the ASIC.

Each address points to 72 bits and as two memories are connected in parallel 144 bits are read or stored every clock cycle (every address). Since DBP has a parallelization level of 18 and a word-length of 8 bits for both the real part and the imaginary part are used, the input to the DBP consists of a signal of 288 bits. The DBP therefore runs on half the speed of the memories all 288 bits can be stored in the memories before the output from the DBP is updated again. The blocks M2D and D2M are used to convert signals between the memories and the DBP block.

To measure the power consumption of the ASIC it should be possible to run it for a longer time. The number of spans the ASIC compensates for are controlled by a *spanNum* signal and if this signal is set to 0 the ASIC will run for an infinite number of spans so that power consumption can be measured.

4.5 Layout of ASIC

The steps for generating the layout for the ASIC design is described thoroughly in section 3.1. For our design we use Cadence Encounter [4] to generate the layout for the design and generate the gds file to be sent to STMicroelectronics for fabrication.

The first step for the layout is to load the netlist and necessary constraint and library files. The library files (lef files) describe the cells and the physical layout of these and are provided by ST. The pads for the design are also placed in this step. After all the files are loaded and the pads placed, the design is floorplanned. First I/O fillers are added between the pads and then the power and ground structures are created. The power rings are added in metal layer two and three and one ring is used for power and one for ground. The placement of the cells is automatic and first pre-placement is done and checked for violations. If there are no violations the placement is refined and optimized for pre-CTS. After the placement of the cells the power net is routed and connected to the power pads already placed in the design. The next step is to route the nets of the design and connect all the cells together. This is done after a final verification of the placement of the cells. First the design is automatically routed globally and then detailed. The design is then timing optimized for post routing. The final step of the design is to add filler cells between the STD cells and do the final verifications for the design. The verifications done are for geometry, connectivity and antenna violations. If there are no violations the design can be extracted to a gds file which can be sent to the factory for fabrication.

5
Results and discussion

Evaluations of the design have been done in order to find the optimal parameters for the DBP method and the results from these tests are presented and discussed in this chapter. First the results from the MATLAB simulations are presented and then the evaluations done on the ASIC design.

5.1 Simulation of TD-DBP

The results of TD-DBP simulations consist of two parts; DBP is first implemented in frequency domain and the dispersion is compensated for with FFT and IFFT functions. The second part is to implement TD-DBP and an LS-CO filter is then implemented for compensation of chromatic dispersion. The parameters considered are the filter coefficients (if non-optimized or optimized coefficients should be used), the number of filter taps, order of Taylor expansion and word-length of signals. The best values for these parameters are investigated in order to get the best performance for the hardware design. The BER aimed at is 10^{-3} when compensating for a 4000 km long fiber divided into 40 spans and a small area is wanted.

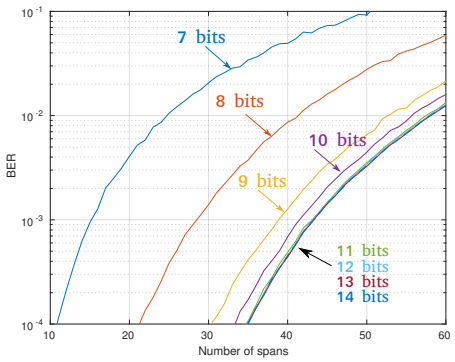
5.1.1 Filter coefficients

The filter coefficients used for the dispersion compensation filter can either be non-optimized or optimized. The MATLAB simulations use non-optimized filter coefficients but the hardware is implemented with optimized coefficients.

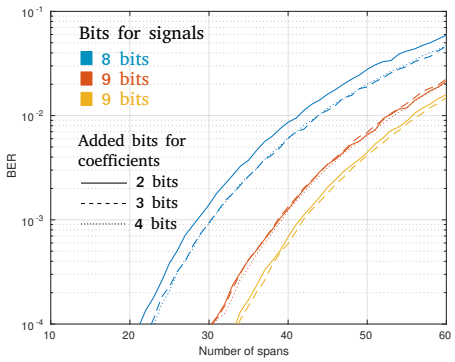
Different combinations of signal and coefficients lengths are simulated when using non-optimized coefficients. The coefficients have longer word-length as these are more sensitive as the error from these accumulates over the spans.

As can be seen in fig. 5.1(a), 9 bits for the signal and 11 bits for the coefficients has a significant performance improvement compared to when 8 bits are used for the signal. For signals with longer word-lengths the performance improvement is negligible. In fig. 5.1(b) the difference in word-length between the length of the signal and coefficient is varied. The performance does not increase noticeably when using a larger difference between the signal and coefficients. The optimal number of bits for the signal are 9 and the coefficient should have two more bits than this i.e. 11 bits for optimal performance when non-optimized filter coefficients are used.

Simulation results from when using optimized filter coefficients shows that the BER is lower than when using non-optimized coefficients and the best word-length for the coefficients are 9 bits.



(a) Coefficients have two more bits than the signal.



(b) The coefficients have 2 or 3 or 4 bits more than the signal.

Figure 5.1: BER results of propagation up to 60 spans with different word-length for the signal and coefficients.

5.1.2 Number of filter taps

LS-CO filters are used in the simulations to compensate for the dispersion accumulated in the signal. As few number of taps as possible is desirable since the filter will be implemented in the ASIC and fewer taps gives a smaller size. However, a filter with too few taps has bad performance and the least number of taps usable for each step has to be found. In order to do this a varying number of filter taps are used in the simulation of the TD-DBP. The result from these simulations, is shown in fig. 5.2.

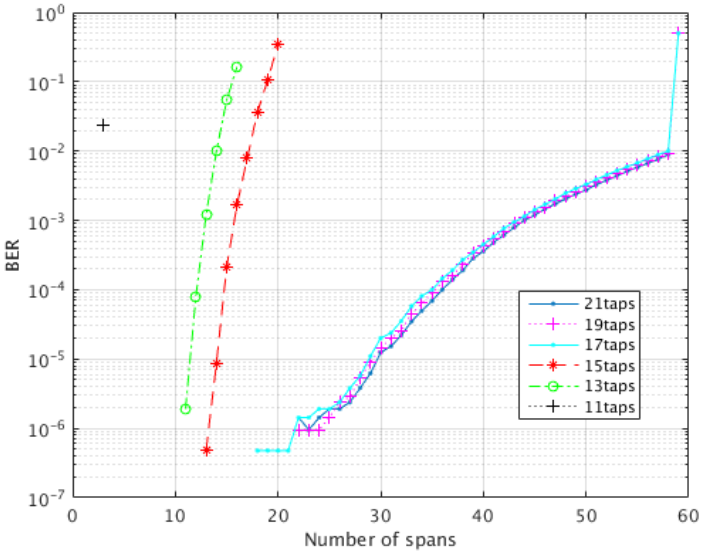


Figure 5.2: Different number of LS-CO dispersion filter taps. Each span is 100 km and divided into two steps. First order of Taylor expansion is used, 9 bits and 11 bits are used for the word-length of the signal and coefficients.

The LS-CO filter is very sensitive when using fewer than 17 taps and it is not possible to use less than 15 spans if the BER should be lower than 10^{-3}. The performance does not change considerably if 17 or more taps are used and 17 is therefore a suitable number of filter taps.

5.1.3 Taylor expansion order

Nonlinear impairments are exponential, (2.21), and in order to implement the TD-DBP algorithm in hardware, Taylor expansion is used instead for an exponential function as described in section 4.1.2.

The order of Taylor expansion does not make a big difference on the performance for fewer than 50 spans. This means that for a lower number of spans, using second order does not affect the accuracy much compared to first order. However, for a higher number of spans, the performance is noticeably affected by the order of Taylor expansion used. As the ASIC mostly will compensate for a lower number of spans Taylor order of one is chosen for our design.

5.1.4 Step size

In section 4.1.2 it is described how the step sizes are determined and how to compensate for the nonlinear impairments in each step. Nonlinearities can either be compensated for in the middle of each step or in the end of each step.

Compensating for nonlinearities in the end of each step introduces more phase rotation than compensating for it in the middle. The accuracy is also much better when compensating in the middle of each span. The BER is extremely high for 15 spans when compensating for nonlinearities in the end of each step. With the same parameters compensating in the middle, the BER is around 10^{-3} when compensating for 40 spans.

As discussed in section 2.4.1 logarithmic step sizes have better performance compared to fixed step sizes when compensating for nonlinearities. Using a factor of $k = 0.4$ we calculated the step sizes to 70 km and 30 km. To see how much logarithmic step sizes affect the performance, we calculated the BER for both fixed and logarithmic step sizes. The result from these calculations is presented in fig. 5.3. For shorter word-lengths, the quantization noise is higher and the improvement of logarithmic step sizes is negligible. However, when using longer word-lengths for the signals, the quantization noise is decreased and the BER is lower when logarithmic step sizes are used.

5.1.5 Final parameters

From the results presented above we have concluded that two pairs of optimized filter coefficients of 9 bits should be used in the final design. The number of filter taps are chosen to 17 and a Taylor expansion order of one is used for the nonlinear compensations. The nonlinearities will be compensated in the middle of each span and logarithmic step sizes are used. The word-length for the signal is chosen to 8 bits, this choice is mainly due to the size of the memories available.

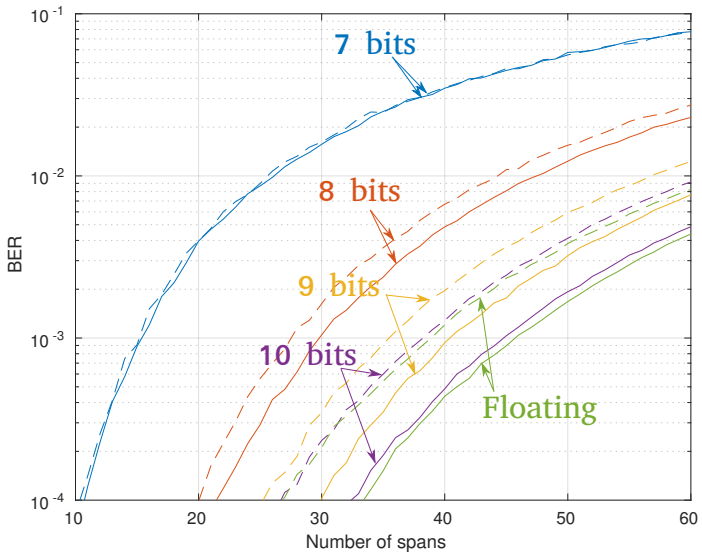


Figure 5.3: BER results of propagation with constant and logarithmic step sizes, dashed lines are constant step sizes and solid lines are logarithmic step sizes. BER with different word-lengths of the signal are shown in the figure.

The constellation for the DBP with these parameters is shown in fig. 5.4 after 40 spans compensation and the BER is 0.007. The groups in the center of the plot have the lowest power and also the best performance. The four corner groups have the highest amplitude and suffer more from more nonlinearities than the other groups which can be seen from that they are not completely circular.

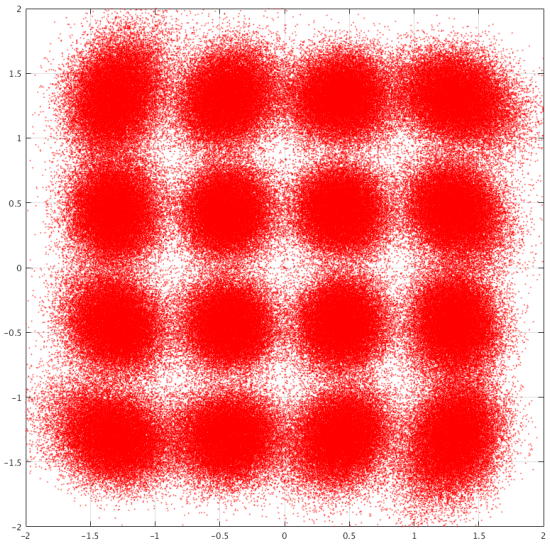


Figure 5.4: Constellation of the output of TD-DBP with the final parameters propagating through 40 spans.

5.2 ASIC evaluation

The parameters relevant for the ASIC design are area, power consumption and clock frequency. In order to estimate these parameters, Cadence RTL Compiler [3] is used for evaluations. Which type of cell libraries used for the layout are shown in table 5.1.

Table 5.1: Information about the cell libraries used.

Parameter	
Tech node	28 nm
Supply voltage	0.8 V
Temperature	125 C
Corner	SS
Threshold voltage	LVT

5.2.1 Timing

In order to estimate the highest possible clock frequency of the ASIC the clock frequency for each main component was evaluated. As shown in table 5.2 the maximum frequency of the `TD-DBP` component depends on if it is implemented with one or two step components. The maximum frequency of the memory is taken from data sheets.

Table 5.2: Highest frequency for each component in the ASIC.

	TD-DBP		Memory
Component	(one step component)	(two step components)	
Frequency [MHz]	500	1000	1848

The `TD-DBP` component, when implemented with one step component should receive data every second clock cycle whilst when implemented with two step components, it should receive data every clock cycle. The memory sends out a new data signal every second clock cycle. This means that if the implementation method with one step component is used the clock for both the memories and the `TD-DBP` component has to be 500 MHz (the lowest frequency for the components) and the output signal is updated every second clock cycle. In case two step components are used the clock frequency can be set to 1800 MHz for the memories and a clock of half this frequency can be used for the `TD-DBP` component. The output data signal would then be updated with a frequency of 900 MHz.

In our design we therefore choose to use the implementation method with two step components as this is faster than the other implementation method. The power consumption and area usage is then slightly higher for the design.

5.2.2 Area and power evaluation for TD-DBP block

The area and power evaluations are done with different signal and coefficient word-lengths and two different frequencies for the clock: 500 MHz and 667 MHz. We use LVT (low threshold voltage) cells for the evaluation because speed is more important than power for the design. The power evaluations are done by setting a toggle rate of 0.2 ns on all input signals with a toggle probability of 50 %.

One step

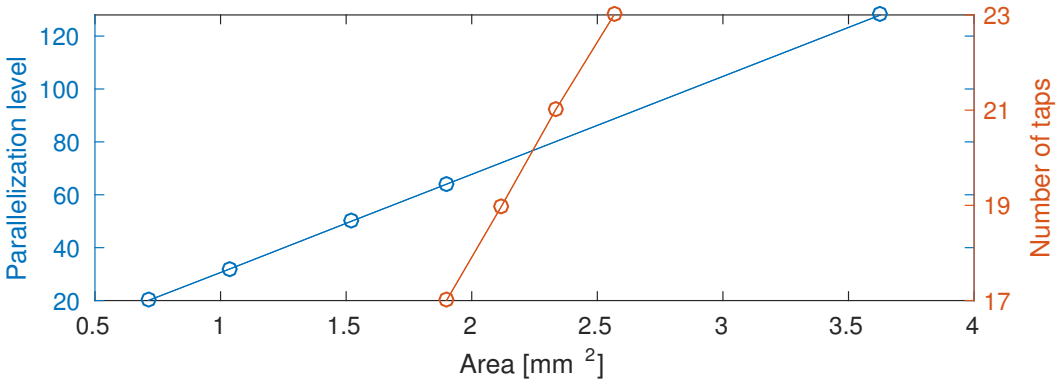
The size of the `TD-DBP` block in the ASIC design depends on different parameters. In order to estimate the size of the `DBP` block for one step of 50 km with different parameter values a number of netlists were generated. The four parameters changed are: the number of bits of the signal, the CD filter coefficients, the parallelization level of the design and the number of taps in the filter. The clock period was kept constant at 1500 ps for these simulations. An area report was generated for each of the different netlists and the result is shown in fig. 5.5. The area report from the netlist gives a summary of the area of each component in the design and also estimates the routing area needed; in fig. 5.5 the total area of the block is shown. The default values are 64 for the parallelization level, 17 for the number of taps, the signal bits and coefficient bits are 8 and 11. The estimated area for these parameter values is 1.899 mm^2. When changing one parameter the others are kept constant at these values.

As expected, when increasing the parallelization level and the number of taps the area of the `TD-DBP` block is increased. By increasing the number of bits used in the signal and the coefficients the area is also increased. The number of signal bits controls the area more than the word-length of the coefficients as they are used in more blocks. As we want the ASIC to be around 1 mm^2 the parallelization level has to be lower than 32 as other blocks also needs to be included in the ASIC. The number of taps and size of signal and coefficients do not affect the area greatly and can therefore be used with the values that gives the optimal performance.

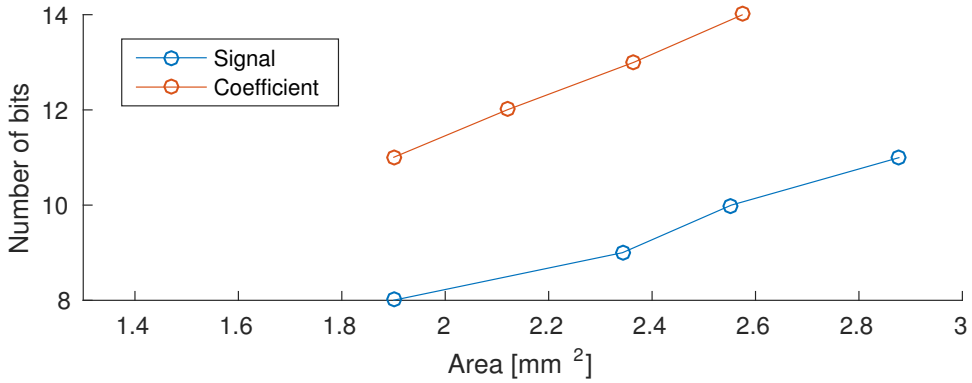
One span

Depending on if two step components or one step component is used for the implementation the power and area for the design are different. Evaluations are therefore performed on one span with both one and two step components. The results from these evaluations are shown in table 5.3. The parallelization level used is 18 and the clock frequency is 500 MHz. The filter coefficients are hardcoded as a signal in the span component. In the table the presented power is the total power, both the leakage power and the dynamic power, measured in Watt. The area is measured in mm^2 and is the area of the cells in the ASIC. In addition to the cell area there is the net area and this area is almost the same size as the cell area. This means that the actual area of the `TD-DBP` block is almost twice as big as the numbers presented in table 5.3.

The area is larger for longer word-length on the signal and the coefficients because the net area is bigger. The cell area, however, remains almost the same in each case



(a) Area with different levels of parallellization and taps.



(b) Area with different number of bits of signal and coefficients.

Figure 5.5: Size of the TD-DBP block estimated from the netlist dependent on different parameter values. The fixed parameters when varying one is 64 for the parallelization level, 17 for the number of taps, 8 bits for the signal and 11 bits for the coefficients.

Table 5.3: Power and area evaluations for one span with one and two step components and fixed filter coefficients. The power is the total power and the area is the cell area.

	Fixed (one component)				Fixed (two components)			
Coeff.	8-bit signal		9-bit signal		8-bit signal		9-bit signal	
word-length	Power [W]	Area [mm^2]	Power [W]	Area [mm^2]	Power [W]	Area [mm^2]	Power [W]	Area [mm^2]
8	0.24	0.20	0.28	0.23	0.42	0.26	0.42	0.29
9	0.29	0.23	0.30	0.24	0.53	0.34	0.55	0.31
10	0.30	0.24	0.36	0.29	0.55	0.36	0.74	0.41

and does not depend much on the number of bits in the signals. The power usage is higher for two components as twice as many gates have to be driven at the same time compared to when one component is used.

One span is built up by either one or two step components and each step component in turn are built up by a `dot` component and a `nhb` component. It is interesting to see how much of the area each separate component needs for the two different span

implementations. In table 5.4 the area for each component in one span is presented for both when one span component is used and for when two span components are used. The input signal word-length is 8 bits and 9 bits are used for the coefficients. The clock frequency is 500 MHz.

Table 5.4: Area of the `TD-DBP` block in the design for 8-bit signal and 9-bit coefficient. The frequency is 500 MHz.

One step component		Two step components	
	[mm^2]		[mm^2]
Step	0.23	**Step 1**	0.17
dot	0.21	dot	0.15
nhb	0.001	nhb	0.001
		Step 2	0.17
		dot	0.15
		nhb	0.001

The `dot` component for the single step implementation is bigger than for the two step implementation as it uses two arrays to store the previous input to the separate steps. The area of the `nhb` blocks are almost the same which is to be expected as the blocks are identical.

In order to see how much using adjustable coefficients that are read from the memory affects the area of the `TD-DBP` block evaluations of this are done. The coefficients are an input signal to the `TD-DBP` block read from a file simulating the memory. The same parallelization level, word-length and frequency are used as for the evaluations for the fixed coefficients i.e. 17 in parallelization level, 8 and 9 bits for the signal and coefficients and 500 MHz for the clock. The result is shown in table 5.5.

Table 5.5: Power and area evaluations for one span with one step component with adjustable coefficients read from memory. The power is the total power and the area is the cell area.

	Adjustable (one component)			
Coeff.	8-bit signal		9-bit signal	
word-length	Power	Area	Power	Area
	[W]	[mm^2]	[W]	[mm^2]
8	0.40	0.31	0.47	0.34
9	0.45	0.34	0.53	0.39
10	0.52	0.38	0.60	0.44

When adjustable coefficients are used every bit of the coefficients has to be connected with a wire to the border of the `TD-DBP` block in order to be able to use these as input signals. Compared to when fixed coefficients are used the coefficient bits has to be tied to either `1` or `0`. This means that the net area is smaller for fixed coefficients and the total area when using adjustable coefficients will be bigger. The cell area is also bigger when using adjustable coefficients as a register for storing

the coefficients are needed. A full multiplier in the `dot` block is also needed when adjustable coefficients are used which also increases the area.

Multiple spans

The area and power of the ASIC should not be affected much when using multiple spans compared to when using one span as the same components are used but a different number of times. For these evaluations a span implementation of two step components are used and the clock frequency is 667 MHz. The coefficients for the dispersion compensating filter are fixed and the number of spans are controlled from an input signal. The power consumption and area usage for two and three spans are evaluated and the result is shown in table 5.6.

Table 5.6: Power and area evaluations for two and three spans with fixed filter coefficients and two step components for each span. The power is the total power and the area is the cell area.

	Two spans				Three spans			
Coeff.	8-bit signal		9-bit signal		8-bit signal		9-bit signal	
word-length	Power [W]	Area [mm^2]	Power [W]	Area [mm^2]	Power [W]	Area [mm^2]	Power [W]	Area [mm^2]
8	0.43	0.28	0.48	0.31	0.42	0.28	0.46	0.32
9	0.55	0.36	0.51	0.33	0.51	0.37	0.49	0.34
10	0.57	0.38	0.66	0.44	0.53	0.39	0.59	0.45

Table 5.6 shows that the area does not vary when multiple spans are used which is what is expected.

5.2.3 Area for finished ASIC

For the design, we want to have the best possible performance and at the same time the smallest possible area. In fig. 5.6 the performance is compared to the area of the `TD-DBP` block. It is clear that the best performance is achieved with longer word-lengths but this also gives a bigger area. It would be optimal to use 9 bits for the signal and 8 bits for the coefficients as this gives the best performance/area combination. However, if 8 bits for the signal is used more symbols can be stored in the memory and as the area of the ASIC is small enough to fit in the limited area of 1 mm^2 we can use the smaller signal length. The best length for the filter coefficients are 9 bits as this gives a relative small area and almost the same performance as for when 10 bits are used.

The final area for the ASIC including the memories cannot be evaluated as we do not have time to design the layout of the ASIC with the memories. However, the estimated area for each memory is 0.057 mm^2 and the area for the `TD-DBP` block is 0.34 mm^2. This means that the cell area for the ASIC is 0.57 mm^2. For the final ASIC the net area is also included into the final area and from the area measurements done we have seen that the net area is almost the same or a bit smaller than the

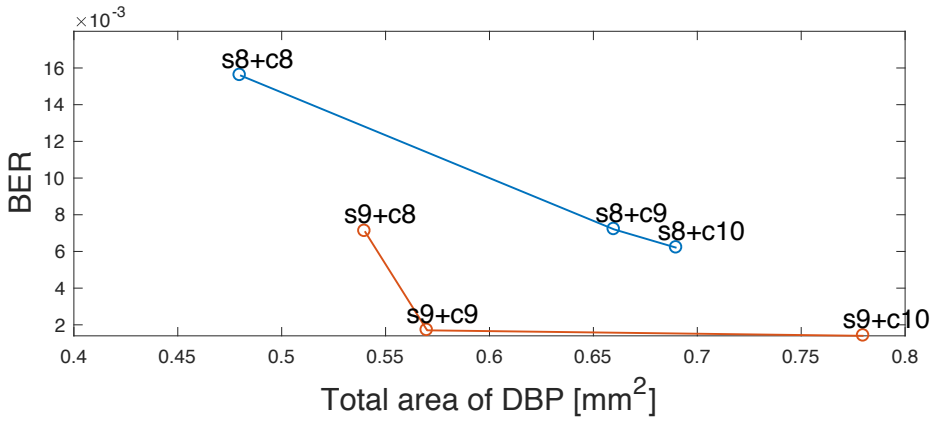


Figure 5.6: BER vs total area of DBP with different word-length of coefficients and signals, the first number represent the signal length and the second is for coefficient length.

cell area. This means that the area of the ASIC can be approximated to 1.1 mm^2 which is a slight overestimation.

6
Conclusion

In the beginning of the project we set up three goals that we were to meet. The first goal was to design the hardware for the test implementation of the TD-DBP method and the second was to design the testing circuit so that the method can be verified on-chip. The last goal was to design the layout of the ASIC on an area no larger than $1\,\mathrm{mm}^2$. Regarding the performance of the DBP method we aimed to reach a BER of 10^{-3} when compensating for accumulated impairments over 40 spans.

In order to meet the first two goals the TD-DBP method was simulated in MATLAB and the optimal parameters for compensating impairments for one filter type was found. The VHDL for both the TD-DBP and the testing circuit have been developed based on these parameters and the functionality verified. The BER reached is $7 \cdot 10^{-3}$ when compensating for 40 spans which is slightly higher than we aimed at but is still acceptable.

The last goal for the project has not been reached as we did not have time to do the layout for the DBP together with the memories. However, evaluations of the area usage for the DBP block has been done and we can estimate the area of the whole ASIC, including memories, to $1.1\,\mathrm{mm}^2$.

7
Future work

As the design is not yet finished there are some future work to be done for the ASIC. Firstly, the bigger memories need to be added to the design and the whole design needs to be verified to work. This should be easy to do when the bigger memories are generated. Secondly, the layout of the ASIC with the new memories needs to be done. To do this a new netlist with the bigger memories needs to be generated and used in the already existing scripts for generating the layout. Then the actual area of the ASIC can also be evaluated.

To be able to test the physical design the input signals from the ASIC need to be created and the output signals need to be taken care of. This can be done by connecting the ASIC to an FPGA designed for generating and taking care of the signals to and from the ASIC. The physical tests on the ASIC that need to be done are to verify that it works correctly and measure the power consumption.

Apart from finishing the design there are also some improvements that can be made on the design. Adjustable filter coefficients can be used instead of fixed which then means that the fiber parameters of the fiber the ASIC compensates for can be changed. This can be favorable as the same ASIC can then be used for different fibers instead for having one ASIC for each fiber. The performance of the DBP can also be increased by using longer word-lengths for the signal and filter coefficients. Right now a signal of 8 bits is used but if choosing a signal length of 9 of even 10 the BER are lower and the performance higher. The memories we have ordered have 72 bits word-length which is perfect for storing 8 bits signals with a parallelization level of 18. So if 9 bits signals are going to be implemented in the ASIC in the future word-length of 81 bits memories can be used, or even longer a word-length for a memory with bit-mask.

Bibliography

[1] C. Fougstedt, M. Mazur, L. Svensson, H. Eliasson, M. Karlsson, and P. Larsson-Edefors, "Time-domain digital back propagation: Algorithm and finite-precision implementation aspects," in *Optical Fiber Communications Conference and Exhibition*, 2017.

[2] *QuestaSim-64, 6.5b*, Mentor Graphics Corporation, 2009.

[3] *Cadence® Encounter® RTL Compiler, v. 14.10-s012_1*, Cadence Design Systems, Inc., 2014.

[4] *Cadence® Encounter®, v14.27-s035_1*, Cadence Design Systems, Inc., 2014.

[5] E. Agrell, M. Karlsson, A. Chraplyvy, D. Richardson, P. Krummrich, P. Winzer, K. Roberts, J. Fischer, S. Savory, B. Eggleton, M. Secondini, F. Kschischang, A. Lord, J. Prat, I. Tomkos, J. Bowers, S. Srinivasan, M. Brandt-Pearce, and N. Gisin, "Roadmap of optical communications," *Journal of optics*, vol. 18, no. 6, 2016.

[6] G. P. Agrawal, *Nonlinear fiber optics*. Boston: Academic Press, 2012, ch. 1, Introduction.

[7] R. Ramaswami, K. N. Sivarajan, and G. H. Sasaki, *Optical networks: a practical perspective*, 3rd ed. Amsterdam: Elsevier, 2010.

[8] G. P. Agrawal, "Nonlinear fiber optics: Its history and recent progress [invited]," *Journal of the Optical Society of America B: Optical Physics*, vol. 28, no. 12, pp. A1–A10, 2011.

[9] S. Mumtaz, R.-J. Essiambre, and G. P. Agrawal, "Nonlinear propagation in multimode and multicore fibers: Generalization of the manakov equations," *Journal of Lightwave Technology*, vol. 31, no. 3, pp. 398–406, 2013;2012;.

[10] T. Li, *Optical fiber communications: Volume 1, Fiber fabrication*. Orlando: Academic Press, Inc, 1985.

[11] S. Singh and N. Singh, "Nonlinear effects in optical fibers: origin, management and applications," *Progress In Electromagnetics Research*, vol. 73, pp. 249–275, 2007.

[12] G. P. Agrawal, *Nonlinear fiber optics*. Boston: Academic Press, 2012, ch. 2, Pulse Propagation in Fibers.

[13] R. Asif, C.-Y. Lin, M. Holtmannspoetter, and B. Schmauss, "Logarithmic step-size based digital backward propagation in n-channel 112Gbit/s/ch DP-QPSK transmission," International Conference on Transparent Optical Networks, Ed., 2011, pp. 1–4.

[14] National Instruments. Pulse shaping to improve spectral efficiency. [Online]. Available: http://www.ni.com/white-paper/14849/en/#toc1

[15] E. Cubukcu. Root raised cosine (RRC) filters and pulse shaping in communication systems. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20120008631.pdf

[16] S. V. Kartalopoulos, *Optical bit error rate: an estimation methodology.* Piscataway, NJ: IEEE Press, 2004.

[17] E. Ip and J. M. Kahn, "Compensation of dispersion and nonlinear impairments using digital backpropagation," *Journal of Lightwave Technology*, vol. 26, no. 20, pp. 3416–3425, 2008.

[18] G. P. Agrawal, *Fiber-optic communication systems*, 4th ed. New York: Wiley, 2010, vol. 222, ch. 7, Loss Management.

[19] D. Kumar, Shiva and M. J. Deen, *Fiber optic communications: fundamentals and applications*, new;1; ed. Chichester, West Sussex, United Kingdom: Wiley, 2014.

[20] J. Riishede and O. Sigmund, "Inverse design of dispersion compensating optical fiber using topology optimization," *Journal of the Optical Society of America B*, vol. 25, no. 1, pp. 88–97, 2008.

[21] C. C. Davis and T. E. Murphy, "Fiber-optic communications [in the spotlight]," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 152–150, 2011.

[22] A. Sheikh, C. Fougstedt, A. G. i. Amat, P. Johannisson, P. Larsson-Edefors, and M. Karlsson, "Dispersion compensation FIR filter with improved robustness to coefficient quantization errors," *Journal of Lightwave Technology*, vol. 34, no. 22, pp. 5110–5117, 2016.

[23] K. Golshan, *Physical design essentials: an ASIC design implementation perspective.* New York: Springer, 2007.

[24] B. Buszewski, E. Dziubakiewicz, and M. Szumski, *Electromigration techniques: theory and practice.* New York: Springer, 2013, vol. 105.

[25] STMicroelectronics. FD-SOI. [Online]. Available: http://www.st.com/content/st_com/en/about/innovation---technology/FD-SOI.html

[26] J. Zhang, X. Li, and Z. Dong, "Digital nonlinear compensation based on the modified logarithmic step size," *Journal of Lightwave Technology*, vol. 31, no. 22, pp. 3546–3555, 2013.