Staggered delay tuning algorithms for ring resonators in optical beam forming networks

by

R.J. Blokpoel

Master thesis
Executed from December 2006 to August 2007

Supervisor: Prof. Dr. Ir. W. van Etten
Advisors: Dr. Ir. C.G.H. Roeloffzen
        Dr. Ir. A. Meijerink
        Ir. L. Zhuang
Summary

The Telecommunication Engineering group at the University of Twente is currently working on two projects about phased array antenna systems with photonic beamformers. One is the SMart Antenna systems for Radio Transceivers (SMART) project. The other is the Broadband Photonic Beamformer project.

Some of the advantages of implementing the beam forming network in the optical domain are immunity to electromagnetic interference, high bandwidth and low loss. The optical delays needed for the beamforming are realized with Optical Ring Resonators (ORRs), which can be tuned by varying their parameters. To get higher delay values multiple rings are cascaded and the rings should be tuned in such a way that the delay ripple is minimal in the frequency band of interest.

The goal of the assignment was to design a subsystem which converts a desired set of delay values into a set of optimal ORR parameters. The first step to do this was to derive optimality criteria. Three different criteria were the result of this: the delay criterion, phase criterion and power criterion. All of them used a metric value which was the sum of the squared deviation from the target at a certain number of points.

Three tuning algorithms followed from those criteria by optimizing for the criteria using Non Linear Programming (NLP) techniques. The phase tuning algorithm was concluded to be best suited to use for the derivation of rules of thumb. This was because negative side effects of improper tuning like the rise of side lobes in the radiation pattern are directly dependent on the phase error and the power tuning algorithm gave only slightly better results at the cost of a significant increase in complexity.

Since these algorithms were too complex to implement directly into a microcontroller, rules of thumb were needed. One option was a lookup table, which had severe memory requirements since the 2 kB for a 1×4 OBFN was approaching the limits of the current system. The other option was to store only coefficients of curve-fitted polynomials, which saves a lot of memory at the cost of a little processing time and accuracy.

The research also gave solutions for adapting the algorithms when a different ORR response formula is found or when the system needs to be expanded to a 1×M OBFN. Another result from the research was the delay tuning range covered by a certain number of rings. From that the conclusion was drawn that only three rings were needed in the first stage of a 1×8 OBFN instead of four.
# Contents

Summary iii

1 Introduction 1
1.1 Background ............................................... 1
1.2 Framework ................................................ 5
1.3 Assignment goal ......................................... 5
1.4 Organization .............................................. 6

2 Properties of the optical ring resonators 7
2.1 Structure of a single optical ring resonators ............ 7
2.2 Manual tuning principles with multiple optical ring resonators .... 11
2.3 Crosstalk between different heaters ................. 12

3 Effects of optical beam forming network parameter errors 15
3.1 Delay ripple and phase errors ............................ 15
3.2 Imperfections in the optical beam forming network chip .......... 20
3.3 Derivation of optimality criteria ......................... 21
  3.3.1 Delay criterion ........................................ 21
  3.3.2 Phase criterion ........................................ 23
  3.3.3 Power criterion ........................................ 23

4 The Optimal Tuning Algorithms 27
4.1 Optimization Method ..................................... 27
4.2 Results for delay response tuning ...................... 29
  4.2.1 One ring ............................................ 29
  4.2.2 Two rings ............................................. 31
  4.2.3 Three rings ........................................... 33
  4.2.4 Tuning a 1×4 optical beam forming network chip at once .... 35
4.3 Results for phase response tuning .................... 36
  4.3.1 Method used for phase tuning .................... 36
  4.3.2 One ring ............................................. 38
Chapter 1

Introduction

1.1 Background

The major benefits of phased-array antenna systems are their high gain and possibility of electronic beam steering and shaping, which is together called beam forming. A phased-array antenna system consists of multiple antenna elements with corresponding tunable phase shifters or delay elements, and some splitting/combining circuitry. Delay elements should be used instead of phase shifters when it is applied to broadband beam forming. The steering resolution for the system then depends on the tuning resolution of the delay elements.

There are several advantages when the beam forming network is implemented in the optical domain. The system will be immune to electromagnetic interference and will have a high bandwidth and low loss. Also it will be compact and have a low weight when it is implemented using integrated optics. More on various types of optical delay elements can be found in [6]. The complete system is shown schematically in Figure 1.1.

![Figure 1.1: The complete antenna system](image)

The first block represents the Antenna Elements (AEs), which are fed to the Electro-optic (E/O) converter, to get the signals into the optical domain. Then the signals from the different AEs are delayed in the Optical Beam Forming Network (OBFN). This is steered by the control block which has the beam angle as input. With this angle the delay values for each AE are calculated so that there is positive interference for radio signals coming from the desired direction at the end of the OBFN. Finally the output of the OBFN is converted back to the electric domain in the O/E block and fed to the
Rx block for detection.

The OBFN block is very important for the system and will be discussed in more detail. It contains splitters, combiners, Mach-Zehnder interferometers (MZIs) and optical ring resonators (ORRs). Especially the ORRs are interesting, since they cause the required delays for positive interference between signals from different AEs. When a straight optical waveguide is coupled to an ORR, it will behave as an all-pass filter with a periodic bell-shaped group delay response. This is illustrated by the dashed lines in the figure below, which are the individual group delays of the ORRs shown in the inset.

![Figure 1.2: Theoretical group delay response of three cascaded ORR sections (taken from [7])](image)

The resonance frequencies of the ORRs, \( f_1 \), \( f_2 \) and \( f_3 \), depend on the round-trip times \( T \) and the extra phase shifts \( \phi_1 \), \( \phi_2 \) and \( \phi_3 \) due to the heater on top of the ring. The group delays at the resonance frequencies depend on the coupling coefficients \( \kappa_1 \), \( \kappa_2 \) and \( \kappa_3 \). Both the \( \phi \)s and \( \kappa \)s can be tuned using chromium heaters, based on the thermo-optical effect. More details on this can be found in Chapter 2.

As can be seen from Figure 1.2 a broadband delay element can be created by cascading multiple ORR sections. The total group delay response (the solid line) simply follows by summing the individual responses (the dotted lines). In this way a flattened response with some ripple on it can be obtained by tuning the ORRs to different resonance frequencies. This is called staggered delay tuning. As can be seen from Figure 1.2 the response of a single ring is much narrower than the cascade of three, so more rings give more bandwidth. When the center frequencies are placed further apart, the bandwidth will be larger but the ripple will also be larger. A solution to keep the ripple at the same level is to use more rings. So more rings means a higher bandwidth or smaller ripple, or a combination of both. However the fabrication costs of the chip increase with the number of rings. Therefore the number of rings is kept minimal.

When all the delay elements and splitting/combining circuitry are realized in the optical domain and integrated onto one chip, an optical beam forming network (OBFN)
is obtained. An example of a $1\times4$ OBFN, based on a binary tree topology, is shown in Figure 1.3.

![Diagram of $1\times4$ binary tree OBFN](image)

**Figure 1.3:** $1\times4$ binary tree OBFN (taken from [1])

This principle was recently demonstrated in [1]. The major advantage of using a binary tree structure is that less rings are required to get four different outputs. Since each output has one ring more in the optical path than the previous one, the delay can be selected in steps. This can be seen in Figure 1.4, where some group delay response measurements of this $1\times4$ OBFN chip are shown. Output 1 does not have any rings and is therefore not present in the graph. Output 2, 3 and 4 respectively have 1, 2 and 3 rings in their path, which results in a linearly increasing delay, when properly tuned. In [7] the same is done for a $1\times8$ OBFN chip.

![Graph of group delay vs. wavelength](image)

**Figure 1.4:** Output measurements of $1\times4$ binary tree OBFN (taken from [1])

The heater elements in the OBFN chip were manually tuned such that flat group delay responses over a bandwidth of 1.5 GHz were obtained. As mentioned earlier, the
manual heater tuning is a laborious procedure, which becomes increasingly complicated for increasing number of cascaded ORR sections. Obviously manual tuning is not desirable in a phased-array antenna system using optical beam forming.

The next block in Figure 1.1 to discuss is the control block. Its main function is to calculate the voltages which should be applied to the heaters of the ORRs. This is not a straightforward operation and therefore it can be divided into several functions. This is shown in Figure 1.5

![Figure 1.5: The project layers](image)

The first layer is the transmit or receive angle of the antenna system. This angle is determined from the position of the other end of the communications link. When either end of the link is moving, a tracking system should also be included in this layer. In Figure 1.1 it is shown as input for the control element. When the angle is known, the group delays and amplitudes for the array elements can be calculated. This is done in the second layer, which passes information about delays to the next layer. Here the optical parameters for the ORRs are calculated to get an optimal delay or phase response. The layer should also compensate for additional optical phases due to some problems which are specific for the chosen OBFN structure. More on those problems can be found in Chapter 2 and 3. The last layer calculates and applies the voltages to the heaters of the ORRs using information of the layers above.

So using the ORRs described in [1] and [2], the implementation of the layers remain as a problem to complete the system. In [3] a solution to the problem of crosstalk between the heaters is presented. It also considers the problem in the last layer of voltage calculation. A solution to apply the voltages to the heaters is presented in [4]. It makes use of a microprocessor which controls a multichannel DAC chip with an analog amplifier at the output. The distribution of control signals and the voltages is done by means of a bus using IDE connectors.

The current and temporal solution for the third layer is manual tuning of the OBFN chip. This is an undesirable activity, which is increasingly complicated with the number of elements in the antenna system and the number of ORRs in the chip.
1.2 Framework

This assignment is part of two research projects carried out by the Telecommunication Engineering group at the University of Twente. The aim of these projects is to develop an antenna array system with photonic beamformers. One is the SMart Antenna systems for Radio Transceivers (SMART) project, which should provide mobile wireless broadband communication access. The pilot should be a system suited to place in an aircraft and uses novel broadband (2 GHz) antenna concepts to receive satellite television. The delays should be tunable between 0 and 5 ns. The project is carried out in cooperation with LioniX, National Aerospace Laboratory NLR and Cyner Substrates. The other project is the Broadband Photonic Beamformer project, which is carried out with LioniX and Astron. The application for the phased array is astronomy in this case. The array will be over one squared kilometer and signals should be processed very accurately over a very broad frequency range. So both projects use the same kind of system but have different bandwidths and frequencies and delay values.

1.3 Assignment goal

With such a large aimed antenna array as in the Broadband Photonic Beamformer project or with a moving object like in the SMART project, the tuning cannot be done manually since it would take too much time for the application. Therefore the tuning should be done automatically. A subsystem should convert a desired delay value to a set of $\phi$s and $\kappa$s to pass on to the next layer. As explained before this will belong to the optical parameters conversion function from Figure 1.5.

The goal of this assignment is to design such a subsystem for the phased-array antenna system. First an algorithm should be made which just converts the input delay value into the required optical parameters resulting in an optimal delay curve. The algorithm should not have too high memory of processing power requirements. When that is done it should be linked to the next layer and take crosstalk of the heaters into consideration too. When time permits it the assignment could also include a calibration procedure to tackle the problem of fabrication offsets in the ORRs. Another possibility is to go up in the layer structure and look at the problem of a tracking system to keep the array aimed at the other end of the communications link. Which way is chosen depends on the prior results and the total project state at that moment. Of course other ways can come up during the research too.
1.4 Organization

A deeper understanding of the working principle of the ORRs is needed to carry out this assignment. Therefore Chapter 2 treats the ORRs in more detail, with formulae for phase and delay responses and the basics of manual tuning. It also contains more details about the crosstalk problem. The next chapter describes the effect of OBFN parameter errors (such as delay ripple) to the performance of the system, resulting in criteria to evaluate tuning algorithms. In Chapter 4 of this report the theoretical algorithm is explained. The next chapter will treat a solution based on a rule of thumb, which is easier to implement and saves a lot calculation time but will be less accurate. Also the alternative of a lookup table will be considered. Finally in Chapter 6 conclusions will be drawn.
Chapter 2

Properties of the optical ring resonators

2.1 Structure of a single optical ring resonators

To be able to solve the problem of tuning the delay elements, it is of course necessary to have a deep understanding of the working of a single ORR. The structure of the ORR is shown in Figure 2.1. In the figure $E$ is the optical field, with $E_1$ the input of the ORR and $E_2$ the output. Two parameters in the structure are tunable: the first one is the coupling constant $\kappa$ of the coupler and the second is the additional phase $\phi$. The roundtrip time of the ring is $T$. Using these parameters the group delay response for a single lossless ORR is, according to [1], given by

$$\tau(f) = \frac{\kappa T}{2 - \kappa - 2\sqrt{1 - \kappa} \cos(2\pi f T + \phi)} \tag{2.1}$$

In [2] a more accurate formula for this response is presented. This one takes losses in the optical ring into account

$$\tau(f) = \frac{T}{2} \cdot \frac{1 - r^2(1 - \kappa)}{1 + r^2(1 - \kappa) - 2r\sqrt{1 - \kappa} \cos(2\pi f T + \phi)} + \frac{T}{2} \cdot \frac{r^2 - (1 - \kappa)}{1 - \kappa + r^2 - 2r\sqrt{1 - \kappa} \cos(2\pi f T + \phi)} \tag{2.2}$$

with $r = 10^{-\alpha/20}$ and $\alpha$ is the loss of the ring in dBs. More about these losses can be found in Chapter 3.

The formulae (2.1) and (2.2) are derived for an ORR with a single fixed coupler, but the one used in this research is a tunable coupler, consisting of two fixed couplers with a phase shifter in between them. To get more insight in the differences between those, the structure of the fixed type is presented schematically in Figure 2.1 and the tunable one in Figure 2.2.
Figure 2.1: Detailed structure of the fixed ORR with only one directional coupler

Figure 2.2: Detailed structure of the tunable ORR with two directional couplers.

Direct derivation of formulae for the group delay in the tunable structure gives quite large expressions. Therefore it is better to keep working with (2.1) and (2.2) while doing the staggered delay tuning and apply a correction afterwards. This correction can be derived by writing the structure of the tunable ORR in the same way as the old one with some corrections. The relation between the fields at the inputs and outputs in the fixed one is given by

\[
\begin{bmatrix}
E_4 \\
E_2
\end{bmatrix}
= \begin{bmatrix}
\sqrt{1 - \kappa} & j \sqrt{\kappa} \\
j \sqrt{\kappa} & \sqrt{1 - \kappa}
\end{bmatrix}
\begin{bmatrix}
E_3 \\
E_1
\end{bmatrix}
\]

(2.3)

The tunable structure consists of two of such couplers and an extra phase shift in the upper branch between those two couplers. It is easy to produce equal values for the \(\kappa_i\)'s of both couplers in the fabrication process and therefore they will be assumed identical.
When the extra phase shift is included into the second matrix the following expression is obtained for the new structure

\[
\begin{bmatrix}
E_4 \\
E_2
\end{bmatrix} = \begin{bmatrix}
\sqrt{1 - \kappa_i} & j\sqrt{\kappa_i} \\
j\sqrt{\kappa_i} & \sqrt{1 - \kappa_i}
\end{bmatrix} \begin{bmatrix}
e^{j\theta} & 0 \\
0 & 1
\end{bmatrix} \begin{bmatrix}
\sqrt{1 - \kappa_i} & j\sqrt{\kappa_i} \\
j\sqrt{\kappa_i} & \sqrt{1 - \kappa_i}
\end{bmatrix} \begin{bmatrix}
E_3 \\
E_1
\end{bmatrix}
\]

(2.4)

When those three matrices are multiplied it results into the transmission matrix \( \mathbf{H} \) of the complete system with the following entries

\[
H_{11} = e^{j\theta/2} \left[ (1 - \kappa_i)e^{j\theta/2} - \kappa_i e^{-j\theta/2} \right]
\]

\[
H_{12} = e^{j\theta/2} \left[ j\sqrt{\kappa_i (1 - \kappa_i)} e^{j\theta/2} + j\sqrt{\kappa_i (1 - \kappa_i)} e^{-j\theta/2} \right]
\]

\[
H_{21} = e^{j\theta/2} \left[ j\sqrt{\kappa_i (1 - \kappa_i)} e^{j\theta/2} + j\sqrt{\kappa_i (1 - \kappa_i)} e^{-j\theta/2} \right]
\]

\[
H_{22} = e^{j\theta/2} \left[ (1 - \kappa_i) e^{-j\theta/2} - \kappa_i e^{j\theta/2} \right]
\]

(2.5)

Note that this matrix could also be obtained by analyzing all possible paths for each matrix entry, which of course gives the same result. The factor \( e^{j\theta/2} \) is pulled out of the matrix. This is done to make the rewriting of the entries \( H_{12} \) and \( H_{21} \) (which are equal) easier and is done as follows

\[
H_{12} = H_{21} = 2j e^{j\theta/2} \sqrt{\kappa_i (1 - \kappa_i)} \cos (\theta/2)
\]

(2.6)

From this the total \( \kappa \) for the ORR (like the \( \kappa \) in Figure 2.1) can be derived which is expressed in \( \kappa_i \) and \( \theta \) as

\[
\kappa = 4\kappa_i (1 - \kappa_i) \cos^2 (\theta/2)
\]

(2.7)

Using this definition for \( \kappa \) it can be seen that \( H_{12} \) and \( H_{21} \) can now be written as \( j\sqrt{\kappa_i} e^{j\theta/2} \), which makes those entries equal to those of (2.3). For the entry of \( H_{11} \) a more advanced trick has to be applied to get to an expression which fits into the original structure. First the Euler formula is applied to get

\[
(1 - \kappa_i) e^{j\theta/2} - \kappa_i e^{-j\theta/2} = (1 - 2\kappa_i) \cos (\theta/2) + j \sin (\theta/2)
\]

(2.8)

The next step is to write this in the polar representation:

\[
H_{11} = \sqrt{1 - 4\kappa_i (1 - \kappa_i) \cos^2 (\theta/2)} \cdot \exp (j\Phi + j\theta/2)
\]

(2.9)

The square root in the first part of the right hand side of the equation is equal to \( \sqrt{1 - \kappa} \) when compared to (2.7). To keep the formulae small and comprehensible an argument \( \Phi \) is defined by

\[
\tan \Phi = \frac{\tan (\theta/2)}{1 - 2\kappa_i}
\]

(2.10)
The derivation of $H_{22}$ is analogous to the one of $H_{11}$ and results into the same modulus but this time with a phase of $-\Phi + \theta/2$. Combining these results the transmission matrix $H$ now becomes

$$H = e^{j\theta/2} \begin{bmatrix} e^{j\Phi} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{1-\kappa} & j\sqrt{\kappa} \\ j\sqrt{\kappa} & \sqrt{1-\kappa} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\Phi} \end{bmatrix}$$

With this result for $H$, the ORR of Figure 2.2 is expressed with almost the same transmission matrix as in (2.3). The only differences are some phases. With those phases pulled out of the matrix the use of formulae (2.1) and (2.2) is justified again. The phase of $e^{j\theta/2}$ was already out of the matrix and can be seen as two phase shift blocks between the coupler and the outputs $E_2$ and $E_4$. For the phases $\Phi$ and $-\Phi$ the matrix of (2.3) should have its upper row multiplied with $e^{j\Phi}$ and its left column with $e^{-j\Phi}$. The same effect can be achieved by adding a phase shift of $\Phi$ between the coupler and output $E_4$ and a phase shift of $-\Phi$ between input $E_1$ and the coupler. However the input of the ORR is $E_1$ and the output is $E_2$, so from a system point of view it does not matter whether the shift of $-\Phi$ is placed after $E_1$ or before $E_2$. Using these derivations a new equivalent structure for the one of Figure 2.2 can be made in which the formulae of (2.1) and (2.2) can be used. This structure is presented in Figure 2.3.

![Figure 2.3: Detailed structure of the corrections applied to the fixed ORR to be equal to the tunable one.](image)

The phase shift $\Psi_1$ is equal to $\Phi + \frac{\theta}{2}$ and $\Psi_2$ is equal to $-\Phi + \frac{\theta}{2}$. When the staggered delay tuning using (2.1) and (2.2) is finished $\phi$ can be corrected by subtracting $\Psi_1$ and $\Psi_2$ has to be taken into account when the optical phases in the different ports of the OBFN chip have to be matched.

In an ideal ORR, $\kappa$ can be varied between 0 and 1. However when taking a closer look at (2.7) and the structure of the ORRs in Figure 2.2 (which is used in [2]), it
can be seen that the coupling constants, $\kappa_i$, of the directional couplers determine the maximum value of $\kappa$. As mentioned earlier it is not difficult to get equal values for the $\kappa_i$s in the fabrication process, but it is hard to get them exactly at 0.5. As can be seen from (2.7), the maximum value of the total $\kappa$ is reached when the phase shift $\theta$ is 0 degrees, since the cosine term then becomes one, and is given by

$$\kappa_{\text{max}} = 4\kappa_i(1 - \kappa_i)$$

(2.12)

The best value for the fabricated $\kappa_i$s currently achieved was 0.465 and using (2.12) this results in 0.9951 as maximum for $\kappa$. The minimum value of $\kappa$ is always zero, since the cosine term is zero when $\theta$ is set to $\pi/2$.

2.2 Manual tuning principles with multiple optical ring resonators

To get more insight in the tuning procedures it is useful to look at some basic principles of manual tuning. Using this, one can immediately compare the results of the automatic tuning algorithm with it and see whether the algorithm gives meaningful results. In Figure 2.4 an example of manually tuning a 1×4 OBFN chip is given. Note that the axes are both normalized with respect to the roundtrip time $T$.

Tuning the first output (one ring) is quite easy: the phase should be tuned so that the bell-shaped response is exactly in the middle of the tuning bandwidth. The $\kappa$ should be set to a value which gives a slightly too high delay value in the center of the tuning range and slightly too low on the edges. That way the average error is the smallest.

The second output with two ORRs is more difficult. One ring should have its peak delay value a little to the left of the tuning range and the other to the right. That can be seen from the dotted responses which are the two separate rings which together form the second output. That way a response is created which is higher at the edges and lower in the middle. It can be tuned more flat by pulling both responses to the middle, but that would not benefit the last output. Also important is the symmetry for the tuning: both ORRs should have the same $\kappa$ and should lie at equal distance from the center.

The sum of the first and second output gives the last output. Since the first one is always too high in the middle it should be compensated for by the second one to get the last as flat as possible. However the optimization algorithm should give the answer to what is exactly the best way of tuning. Now the difference in delay between the first and second output is too small in the center and too big at the edges, with a relatively flat response for the third one. One could imagine that there is a certain
Figure 2.4: Results of manually tuning a 1×4 OBFN chip. The outputs have increasing delay values with the number of rings. The first output is with only one ring (dash-dotted), the second with two rings (dashed) and the last (solid) is the sum of both and thus has the highest delay value. The output with two rings is composed of the two dotted responses. The dashed grid represent the target bandwidth and delay values for the tuning.

optimum between tuning the third one flat or tuning the second one flat. This is also where the relations between the different outputs in the chip become important and that is described in Section 3.3.

2.3 Thermal crosstalk between different heaters

The problem of crosstalk between the heater elements on the OBFN chip was already mentioned in the introduction and will be further explained in this section. In [3] the problem for two heaters which should cause certain phase shifts is treated. The relation between phases and voltages is given by:

\[
\begin{bmatrix}
\phi_1 \\
\phi_2
\end{bmatrix} = \begin{bmatrix}
a & b \\
c & d
\end{bmatrix} \begin{bmatrix}
V_1^2 \\
V_2^2
\end{bmatrix}
\]  

(2.13)

in which the \(\phi\)s are the phase shifts, the \(V^2\)s the squared input voltages to the heaters and \(a, b, c\) and \(d\) are the transfer coefficients between the voltages and phases. So not only \(V_1\) determines \(\phi_1\), but also \(V_2\) as can be seen from the formula. This is due to the
crosstalk between the heaters. The heat from one heater will spread out over the chip and causes some heating in other regions too. When for example the heater on top of Figure 2.2 is turned on, there will be a rise in temperature in the environment of the lower heater too. When $a$ and $d$ are taken 9 and $b$ and $c$ are 1 for example, it results in Figure 2.4, which is taken from [3].

![Figure 2.5](image)

**Figure 2.5:** Possible values for the phase shifts, $\phi_1$ and $\phi_2$, while tuning with crosstalk. The allowed values are in the white area (taken from [3]).

As can be seen from the figure it is not possible to have a shift of $\pi$ at $\phi_1$ and 0 at $\phi_2$. The solution to reach those values is to turn on the other heater too such that $\phi_2$ will become $2\pi$, which of course has the same effect as 0. In [3] an iterative scheme is presented to calculate the voltages while taking the crosstalk into account. First it determines the value as if there were no problem with negative phase shifts. This is done by inverting the matrix. However some desired values, the ones in the gray areas to the left and below the white parallelogram, will result in negative phase shifts when the matrix is inverted. This is when the iterative part of the scheme comes in since it starts to add $2\pi$ to each heater outside the tuning range until a valid solution is reached. That way a solution is reached without negative phase shifts. It can be
used to any number of heaters. The parallelogram in Figure 2.4 would then become an $n$-dimensional one, but the principle remains the same.
Chapter 3

Effects of optical beam forming network parameter errors

3.1 Delay ripple and phase errors

The goal of this assignment is to design a subsystem which converts a desired delay value to a set of $\phi$s and $\kappa$s in such a way that an optimal response results. This section describes the effects of errors in the phase response. Using that theory it is possible to evaluate whether the subsystem reaches an optimal solution or not.

The delays are required to aim the phased array antenna in a certain direction. So only with correct delays the signals will have optimal constructive interference resulting in the highest output power. To get more insight in the effects of delay or phase errors to the output power, it is good to take a look at the direct effects of delay ripple to the output of a single DE. In Figure 3.1 the group delay response of a DE with much ripple and one with little ripple is presented. Details about those simulations can be found in [9]. It can be clearly seen that the input pulse is severely distorted when the DE has a large ripple. Also the transient time will be much longer which could be a source for Inter Symbol Interference (ISI).

The next step to get the total output power is to take a look at the expression for the desired signal

$$s(t) = \sum_n r_n(t) \cos(2\pi f_{IF,n} t + \psi_n(t))$$  \hspace{1cm} (3.1)

The set $n$ represents the different subcarriers in the spectrum considered, so $f_{IF,n}$ are the carrier frequencies. The amplitude $r_n(t)$ and phase $\psi_n(t)$ depend on the modulation type of the signal. When all carriers are summed the desired signal results. In fact this can be seen as the input for the system.

It is also important to look at the kind of modulation and detection used in the system to get to an expression for the output power. In Figure 3.2 a block diagram of
the system concentrating on the modulation is presented. There is a Single SideBand-Suppressed Carrier SSB-SC modulation block for each output signal of an AE, but first each signal is amplified by a Low Noise Amplifier (LNA). The index $m$ represents the different AEs and goes from 1 to $M$. More details about the SSB-SC block can be found in [16].

Now the basic principle behind the delays in the OBFN is presented. This is the last step toward an expression for the output power. In a phased-array antenna system the desired signal is received in each AE with a certain time delay due to the physical location of the antenna. So each AE will receive $s(t)$ with an additional time delay ($T_m$). When extra time delays are added in a correct manner all signals from the different AEs will have constructive interference for radio waves coming from a certain direction. These extra time delays are caused by the DEs in the OBFN chip. So $s(t)$ should arrive at the detector for all AEs with a delay of $T_{\text{max}}$, which is an arbitrary common delay.
Figure 3.2: OBF system with optical SSB-SC modulation and balanced coherent detection. (taken from [12])

and should be larger than all delay values $T_m$. To achieve this, the OBFN should add a delay of $T_{max} - T_m$ to every signal. In [5] the theory of [8] with formulae for the output power in the ideal case is used together with a few assumptions to get the relation between output power and phase error. So when the magnitude response and group delay response within a particular channel are assumed to be flat, the output field is equal to

$$\begin{align*}
E_{\text{out}}(t) &= \frac{1}{2} \sqrt{\frac{P_o}{2L_s L_c}} \sum_n \sum_{m=1}^M a_m |H_m(f_o + f_{IF,n})| \exp \left(j \Delta \phi_m(f_o + f_{IF,n})\right) \\
&\cdot r_n(t - T_{\text{max}} - \Delta \tau_m(f_o + f_{IF,n})) \exp \left(j 2\pi t(f_o + f_{IF,n})\right) \\
&+ j 2\pi (f_{IF} - f_{IF,n}) T_{\text{max}} + j \psi_n(t - T_{\text{max}} - \tau_m(f_o + f_{IF,n}))
\end{align*}$$  \(3.2\)

with $P_o$ the optical power, and $L_s$ and $L_c$ the splitting and combining loss, respectively. The index $m$ represents the different AEs, so $a_m$ is a weighting factor for each different AE to the output, the absolute value of $H_m$ is the magnitude response through the OBFN chip for such a signal.

The group delay ripple $\Delta \tau$ will in general be much smaller than the symbol time, which is 25-50 ns for Digital Video Broadcasting for Satellite (DVB-S). Therefore the envelope shifts can be neglected in (3.2). The detector is a balanced coherent detector with photodiodes as can be seen from Figure 3.2. The following expression results for the detector output current

$$\begin{align*}
I_{\text{out}}(t) &= \frac{R_{pd} P_o}{2 L_s \sqrt{L_c}} \sum_n \sum_{m=1}^M a_m |H_m(f_o + f_{IF,n})| r_n(t - T_{\text{max}}) \\
&\cdot \sin (2\pi f_{IF}(t - T_{\text{max}}) + \psi_n(t - T_{\text{max}}) + 2\pi f_{IF} T_{\text{max}} + \Delta \phi_m(f_o + f_{IF,n}))
\end{align*}$$  \(3.3\)

with $R_{pd}$ the responsivity of the photodiode. Ideally the sine terms should all be in phase for the different $m$s, but this is not the case when the phase errors $\Delta \phi_m$ in a
certain carrier frequency \( f_{\text{IF},n} \) are not all the same. In the scope of this assignment only the proportionality of the phase error for each subcarrier is important and therefore only this part will be considered. So the losses of \( L_s \) and \( L_c \) will not be considered. Also the optical power and responsivity of the photodiode are not in the scope of the assignment. So when those parts are removed, the following proportionality constant remains

\[
\left| \sum_{m=1}^{M} a_m |H_m(f_o + f_{\text{IF},n})| \exp(j\Delta \phi_m(f_o + f_{\text{IF},n})) \right| \tag{3.4}
\]

To get the strongest field and thus output current the proportionality constant should be maximized with respect to the phase error.

Errors in the phase response cause a lower output power but that could also be seen as a gain reduction. Using the proportionality from (3.3) it can be seen that the gain reduction in dB is equal to

\[
P_{\text{gain}} = 20 \log \left| \sum_{m=1}^{M} a_m |H_m(f_o + f_{\text{IF},n})| \exp(j\Delta \phi_m(f_o + f_{\text{IF},n})) \right| \tag{3.5}
\]

When all \( a_m \) are assumed to be equal and the magnitude response of \( H_m \) is assumed to be one for every \( m \), the effect of the phase error can be isolated. When the phase errors are bounded by \( \pm \Delta \phi_{\text{max}} \), the worst case occurs when half of the errors are \( +\Delta \phi_{\text{max}} \) and the other half are \( -\Delta \phi_{\text{max}} \). The gain reduction in dB then becomes [5]

\[
P_{\text{gain}} \leq -20 \log \left[ \cos \Delta \phi_{\text{max}} \right] \tag{3.6}
\]

In Figure 3.3 a plot of this function is shown. It can be seen that the gain reduction caused by phase errors is not very severe since the penalty is still below 0.7 dB with a phase error of \( \pi/8 \).

Worse interference suppression is another effect of errors in the phase response. Small errors in there cause some residual signal from unwanted directions since the destructive interference is not completely destructive anymore then. A measure for the consequences of this is the Normalized Residual Null Level (NRNL), which is the output signal level from an unwanted direction (which should be zero) normalized with respect to the signal level from a wanted direction. Just as for the gain reduction case all \( a_m \) and \( H_m \) are assumed to be equal and worst case is considered, so half of the errors is \( +\Delta \phi_{\text{max}} \) and the other half \( -\Delta \phi_{\text{max}} \). Using those assumptions the NRNL, is derived in [5] to be approximately equal to

\[
NRNL \approx 20 \log \left[ \frac{2}{\pi} \sin \left( \Delta \phi_{\text{max}} \right) \right] \tag{3.7}
\]

The figure below shows a plot of the function.
The NRNL is still below -12 dB when the phase error is $\pi/8$ and below -18 dB when the phase error is $\pi/16$. However it depends on the demands of the system whether those errors are severe or not.

However when the complete radiation pattern is considered, the nulls are not that important. It is more useful to look at the sidelobes. This has been done by NLR, one of the partners in the project, in [17] for an array consisting of 8 AEs. When there are phase errors present, the radiation pattern is not fixed anymore and the signal level lies between a maximum and minimum value. The range between those values becomes larger with higher errors. Since only the worst case is interesting, the minimum bound for the signal level will be dropped from now on. Note that the minimum value is important for the main lobe, but that theory was described with (3.5) and (3.6). The first sidelobe is the strongest one and rises from -13 dB (the ideal case) to -9 dB at maximum when the phase error is $\pi/8$. For $\pi/16$ the rise is reduced to 2 dB and the null is then at -11 dB. The complete patterns for many different phase errors can be found in [17].

From this theory it can be concluded that the sidelobes are most important to look at when it is evaluated whether a certain phase error is still acceptable. However the patterns may differ when an array with more elements is used. The signal levels of the sidelobes are of course higher than the nulls and therefore the nulls are less important. However generally speaking there might be applications in which a specific angle has

\textbf{Figure 3.3:} Gain reduction as a function of the maximum phase error ($\Delta \phi_{\text{max}}$). (taken from [5])
to be suppressed, because a much more powerful transmitter is present in there for example. So all effects have to be kept in mind but the most important for the rest of this research will be the one of the sidelobes, since other transmitters (like other satellites) may be present in there, and according to that one an error of $\pi/16$ should still be acceptable. For other applications this could be different however. It is also important to note that this is just a parameter which should be kept in mind during the research and not of fundamental importance for the optimization procedure used.

3.2 Imperfections in the optical beam forming network chip

As mentioned in Chapter 2 there are losses in the OBFN chip. Especially in the rings the losses are quite high with 0.8 dB/cm as is measured in [10], but it will be improved with better technology which is presented in [11]. In [12] a better technology with 0.55 dB/cm is presented and the technology of [13] should even be able to reach 0.1 dB/cm. In Figure 3.5 the same delay response as in Figure 3.1b is presented but now the losses in the ring are taken into account too.

As can be seen there is quite a lot of loss because of the ring. Especially for the higher delay tunings this is the case since the ring of 1.2 cm circumference has to be
Figure 3.5: The pulse response with the same delay ripple as in Figure 3.1a but now with attenuation effects taken into account. (taken from [9])

traveled many times then. This effect will cause a high difference in output powers between branches with high delay and low delay. However when the technology of 0.1 dB/cm is used the differences between the OBFN outputs are much smaller and are expected to be small enough to get compensated by tuning the couplers, which divide the power between the branches of the OBFN chip (κ₅, κ₆ and κ₇ in Figure 1.3)

Another issue is the roundtrip time of the ring which can vary due to tolerances in the fabrication process. This introduces a variable φ value in the equations such as (2.1) and (2.2), which of course makes it harder to tune the system. This problem was already mentioned in the introduction, where the need for a calibration procedure was expressed. Note that the effect of those errors on T is negligible so that only a phase difference can be seen.

3.3 Derivation of optimality criteria

3.3.1 Delay criterion

The goal of this assignment is to generate an optimal group delay curve of the cascaded ORRs. However when optimizing something, one should first know how optimality is defined. Therefore this section will derive criteria to optimize for together with associated constraints.

The first criterion is based on the delay spectrum formula presented in the first section. The first step is to generalize this formula by adding subscripts l to the parameters which could be different between ORRs

\[ \tau_l(f) = \frac{\kappa_lT}{2 - \kappa_l - 2\sqrt{1 - \kappa_l \cos (2\pi f T + \phi_l)}} \]  

(3.8)
Chapter 3. Effects of optical beam forming network parameter errors

in which the $\phi$s and $\kappa$s are tunable with heaters. The delay responses, $\tau_l(f)$, should be summed to get the total delay response of the cascaded ORRs

$$\tau_{\text{total}}(f) = \sum_l \tau_l(f) \quad (3.9)$$

This delay spectrum can now be compared with the target delay ($D$) and a criterion for optimality can be defined. The value of the delay is only important in that part of the spectrum where the modulated optical signal is located. Therefore the comparison should be carried out in a certain band, defined by a start ($f_{\text{min}}$) and end frequency ($f_{\text{max}}$). It can be compared by means of the Minimum Mean Squared Error (MMSE) for example

$$\mu = \int_{f_{\text{min}}}^{f_{\text{max}}} (\tau_{\text{total}}(f) - D)^2 df \quad (3.10)$$

The integral of this squared error function results into a metric ($\mu$), which should be minimized in order to get an optimal result. $D$ is assumed to be constant over the interval considered and therefore has no frequency dependence. The method of minimizing is MMSE as can be seen from the square of the error in the formula. This criterion is chosen because there are many channels in the spectrum which should all have an acceptable signal level. So to prevent a result with most channels perfect, but a few so bad that using them is impossible, the MMSE punishes a larger error harder. The nature of the ORR responses with relatively flat graphs in the considered regions and without discontinuities does not require a higher order error function than a quadratic one. The maximum error in the considered set of frequencies could also be taken as criterion, but that criterion would result into mathematical issues since the second order derivate won’t be a continuous function anymore.

The reason why this structure with a metric is chosen is because it is suited for Non-Linear Programming (NLP) solvers. The idea behind this general class of solvers is that they minimize a certain expression, the $\mu$ in this case, subject to a couple of constraints. For this problem the values of the $\kappa$s must be larger than 0 and smaller than 0.9951, as explained in Chapter 1. The bounds for the $\theta$s are not really important but it could make the solver more stable when they are bounded in such a manner that the center of the peaks do not lie outside the frequency range between $f_{\text{min}}$ and $f_{\text{max}}$. More on the solving method can be found in Chapter 4.

The evaluation of the integral in (3.10), however, produces an expression which is quite involved and large since it would take over 1 page to display. This would annoy the NLP solver badly, resulting in enormous calculation times. Therefore it would be better to approximate the integral with a Riemann sum.
\[ \mu = \sum_k (\tau_{\text{total}}(f_k) - D)^2 \]  

(3.11)

in which \( f_k \) is the set of frequencies over which the summation has to be carried out. Note that multiplication with the interval length is left out since that is an unimportant scaling factor in the minimization problem. The number of points in the set \( f_k \) determines the accuracy of the approximation. In case Frequency Division Multiplexing (FDM) is used all points in the set of the sum could be chosen equal to the FDM carrier frequencies. Also note that the index \( n \), which was introduced in Section 3.1, represents the channel number in there. When the frequencies in \( f_k \) are chosen equal to the carrier frequencies of the channels as mentioned in Section 3.1, \( k \) can be replaced by \( n \) and \( f_k \) by \( f_0 + f_{\text{IF},n} \). The previous derivations of the Riemann sum can be linked easily with the result of [5]. In there the assumption was made that the magnitude response and group delay response are constant within one channel to justify the replacement of an integral over \( f \) by a sum over the channels. That is in fact the same as done in here when replacing the integral of (3.10) with a Riemann sum in (3.11). Therefore \( k \) will be replaced by \( n \) from now on.

3.3.2 Phase criterion

Another way to define a criterion is by looking at the phase response in stead of the delay. Note that the delay response was derived from this phase response by taking the derivative and multiplication by \(-1/2\pi\), as is explained in [2]

\[
\psi(f) = \arctan \left( \frac{\sin \left(2\pi f T + \phi_l\right)}{\sqrt{1 - \kappa_l} - \cos \left(2\pi f T + \phi_l\right)} \right) \\
- \arctan \left( \frac{\sqrt{1 - \kappa_l} \sin \left(2\pi f T + \phi_l\right)}{1 - \sqrt{1 - \kappa_l} \cos \left(2\pi f T + \phi_l\right)} \right) 
\]

(3.12)

For this equation the same reasoning holds as for (3.8) to (3.11) when MMSE is applied and this results into

\[
\mu = \sum_n \left( \psi_{\text{total}}(f_0 + f_{\text{IF},n}) + 2\pi D (f_0 + f_{\text{IF},n}) \right)^2 
\]

(3.13)

3.3.3 Power criterion

These criteria are suited for one set of cascaded ORRs. However the criteria should be expanded when a binary tree structure like in Figure 1.4 is used. This is to take the mutual relationships between the paths in the binary tree into account. One could imagine that when output 4 is tuned optimal, the performance of the other outputs
could get worse. To get more insight into the tuning of a complete OBFN chip a simple scenario of tuning two outputs is presented in Figure 3.6. Two possible tunings are presented in there. The first one (dotted arrow) is optimized to have the largest power in combination with the first output and the second one (dashed) is optimized on a stand-alone basis to have minimal phase error.

![Figure 3.6:](image)

**Figure 3.6:** Representation of output powers in the complex plane. The phase error can be seen from the angle with the real axis, and the power from the absolute length of the vectors. The output of the first ring is represented by the thin solid arrow, and two possible tunings for the second output are displayed (dotted and dashed arrow). The bold line represents the sum of the solid and dotted output vectors. It is rotated to the real axis to make the loss of power with respect to the sum of the solid and dashed vectors clear.

From this figure it becomes clear that tuning all outputs individually is not always the best option. The phase error of the combination of the first and second output is better for the option using the dotted arrow than the dashed one, but the combination of both has a smaller modulus and thus less output power. Therefore the theory of the relation between phase errors and system output power from Section 3.1 is needed. For convenience the proportionality of the output power of one subcarrier with the phase is repeated here (the different AEs are denoted by the index $m$)

$$\sum_{m=1}^{M} a_m |H_m(f_o + f_{IF,n})| \exp(j\Delta\phi_m(f_o + f_{IF,n}))$$

(3.14)

Since the usage of $n$ was already implemented in the previous derivations, (3.13) and (3.14) can easily be combined into a new metric, which also takes the relations between the ORRs in a tree structure into account
\[ \mu = \sum_n \left[ \sum_{m=1}^{M} a_m |H_m(f_0 + f_{IF,n})| \right. \]

\[ - \left| \sum_{m=1}^{M} a_m |H_m(f_0 + f_{IF,n})| \exp \left( j(\psi_{\text{total},m}((f_0 + f_{IF,n}) + D_m (f_0 + f_{IF,n}))) \right) \right|^2 \] (3.15)

The first step was to substitute the expression for \( \Delta \phi_m(f_0 + f_{IF,n}) \) with \( \psi_{\text{total}}((f_0 + f_{IF,n}) + D (f_0 + f_{IF,n})) \), which was also used in (3.13). Then the ideal power would be reached when all the phases are equal, so the loss of power could be expressed by subtracting the actual output power from the ideal one. In the expression the part to the left of the minus sign between the brackets is the ideal power and to the right is the actual power. After this new way of expressing the error, the result is squared and summed over all frequencies.

Three different metrics are now derived to evaluate the optimality of the tuning. One is suited to minimize the delay ripple (3.11), the second one is to minimize the phase error (3.13). Both metrics are suited for a structure of \( n \) cascaded ORRs. The last criterion also takes the mutual relations between the phase errors of paths inside a \( m \times 1 \) tree structure into account and minimizes the squared deviation of the optimal power. Now that the metrics are ready, the next step is to implement them into an NLP solver. This will be done in Chapter 4.
Chapter 4

The Optimal Tuning Algorithms

4.1 Optimization Method

In this section the method for optimizing the equations (3.11), (3.13) and (3.15) will be treated. As mentioned in Chapter 3 an NLP solver will be used for this.

The algorithm which will be used in this research is implemented in the \textit{fmincon} function of MATLAB. Since the problem is not a large-scale problem according to NLP criteria (e.g. 1000 or more variables to optimize for) the medium-scale method of \textit{fmincon} is used. The method solves the problem of finding the minimum of a function \( f(x) \) subject to several constraints. The usual notation of these problems is as follows

\[
\begin{align*}
\min & \quad f(x) \\
\text{s.t.} & \quad l \leq x \leq u \\
& \quad g_i(x) = 0 \\
& \quad g_i(x) \leq 0
\end{align*}
\]

in which "s.t." stands for "subject to". The vectors \( l \) and \( u \) are the piecewise lower bound and upper bound for the vector variable to solve for \( x \). The structure of the constraints varies in literature but the one with a lower and upper bound is best suited for the problems described in Chapter 3 (recall that the value of \( \kappa \) should lie between 0 and 0.9951). The most general way to specify constraints is to set a certain function of \( x \), \( g_i(x) \), to be equal to or smaller than zero for bound constraints and set it equal to zero for normal constraints. All types of constraints can be written that way. So the equations (3.11), (3.13) and (3.15) can serve as functions to minimize for directly and can be seen as \( f(x) \) to the NLP solver used. Specification of the bounds is also quite trivial, since all \( \kappa s \) should lie between 0 and 0.9951 and a phase shift of more than \( 2\pi \) is theoretically not meaningful either. After the optimal delay tuning is finished and the crosstalk problem is considered, phases of more than \( 2\pi \) could be introduced.
Therefore the $\phi$s will be constrained to lie between 0 and $2\pi$.

The general structure together with the method of input of the NLP solver is clear now. The next steps to solve the problem are taken inside the \textit{fmincon} function. This is described in more detail in Appendix A.

Concluding from this description of the used NLP algorithm there is no accuracy loss for the kind of problem used in this research. One might wonder why an NLP solver is used and not just a Linear Programming (LP) or Quadratic Programming (QP) technique. This is because the objective function (the metrics from Chapter 3) is not linear. Therefore direct application of LP or QP is not possible.

The only drawback of this solution is that there is a risk that the algorithm terminates at a local optimum instead of the global optimum. This is where the initial guess input for the algorithm is needed. When the solution is tried to be reached from all possible directions while the same solution is reached it must be the global solution. When this is not the case, the region in between the solutions has to be checked too and then the global solution can be picked out.

Since the problem is not really difficult in the case of this research the effect is not expected to occur or it will be easily tractable. However while tuning with more rings in a single DE the effect should occur. This is because interchanging the tunings of two rings of course does not influence the response, but it results into two different solutions with equal objective function values. In the next section results of the tuning algorithm will be discussed and from that more conclusions about this problem can be drawn. The problem could also be redefined by tightening the constraints so that only one solution is possible and symmetry can be exploited by expressing some of the $\kappa$s and $\phi$s in others. See for example the manual tuning principles in Section 2.2: for two rings the $\kappa$s should be equal and $\phi_1$ should be equal to $-\phi_2$. However this is not done to prevent loss of generality and because the calculation capacity is expected to be by far enough to solve the problem in reasonable time. When a rule of thumb in Chapter 5 is derived these concepts will be exploited since calculation time and memory needed is very important then.

In Chapter 3 it is described that the set of frequencies in (3.11), (3.13) and (3.15) should be chosen equal to the carrier frequencies of the system. However when there is not enough information about the system or when less accuracy is needed, the set of frequencies can be chosen differently. When that is the case the points should be spread out evenly in the target bandwidth. Since the responses with multiple ORRs are quite steep at the edges, both edge points will be chosen as optimization point in the algorithm. The more points are chosen the more accurate the solution will be. However the effect of more then 10 points is not expected to be significant since the responses are quite smooth.
4.2 Results for delay response tuning

4.2.1 One ring

The first optimization runs were done for DE sections with only one ring in order to start simple and get more insight in the material. For delay response tuning of a single ORR the formula of (3.11) can be used as object function and the constraints for $\kappa$ and $\phi$ from Section 4.1 will be used. The precision for both the object value and the step size will be set to $10^{-16}$. This is because the calculation time is still below one second for such a small size and it is way below the precision of the used D/A converters of [4]. Due to some issues in the handling of small numbers in MATLAB the object function is multiplied with $10^9$, which of course does not influence the actual solution. Note that this case with only one ring could have been solved analytically too, but that will not be possible for two or more rings and therefore this one will not be solved analytically either.

The results of tuning one ring at 0.2 ns and 0.1 ns are shown in Figure 4.1. A roundtrip time for the ring of 0.08 ns was used and a wavelength of 1550 nm. This corresponds to a carrier frequency of 193.5393817 THz which is chosen as the center point for the part of the spectrum which has to be optimized. These numbers are chosen in consensus with the current research project. The algorithm also works for completely different numbers, but these ones are easier to compare the results with prior research.

The optimization results are presented in Table 4.1.

<table>
<thead>
<tr>
<th>Target delay (ns)</th>
<th>$\kappa$</th>
<th>$\phi$</th>
<th>Metric value (ns$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9850</td>
<td>0.0092</td>
<td>$1.0 \times 10^{-5}$</td>
</tr>
<tr>
<td>0.2</td>
<td>0.7814</td>
<td>0.0092</td>
<td>$4.2 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

Concluding from the table, the $\kappa$ for 0.1 ns is higher than for 0.2 ns and both phases are the same. This is just as one would expect according to the manual tuning principles of Section 2.2. So far the algorithm works fine. Supplying different initial guesses gave the same result, so there was no local minimum in which the minimization terminated. Also the inaccuracy in the calculation of the center frequency was compensated. This inaccuracy is caused by the fact that the delay response is a function, which does not have a maximum exactly at the corresponding frequency for 1550 nm. The asterixes in the graph give more insight in the validity of the tuning: the center asterix is slightly below the response and the others are slightly above. This seems reasonable since the response is always bell-shaped. As could be expected the total squared error is higher
Figure 4.1: Optimization result for one ring tuned at 0.2 ns delay (solid line) and 0.1 ns (dashed line). The stars indicate the reference points for the tuning process. The distance between the outer most stars is 2 GHz.

for 0.2 ns than for 0.1 ns as can also be seen from Table 4.1.

An optimalization over three points is not very accurate, but was still expected to give acceptable results in Section 4.1. To verify this the calculations with a target delay of 0.1 ns were repeated for 10, 100 and 1000 points. The result for $\phi$ was the same for each calculation, but the $\kappa$s differed from the first calculation with only three points. With 10 points $\kappa$ was 0.9861, for 100 it was 0.9863 and 1000 gave 0.9864 as result. Adding more than 1000 points to the calculations did not make any difference, so that result could be considered as the actual solution. This confirms the theory of Section 4.1 since the $\kappa$ of three points is only 0.14% different from 1000 and the one with 10 points is only 0.03% different.

The calculations of 0.2 ns were also repeated and the result with three points differed 0.82% from the one with 1000 points, which was 0.7878 and could again be considered as the actual solution since adding more points did not result into a different solution. One might wonder why this deviation is more than in the 0.1 ns case. From Figure 4.1 it can be seen that the ring has to be tuned more peaky than in the 0.1 ns case. This results into a higher error in both the center and the edges of the spectrum. However when taking a closer look at the figure it can be seen that the response is somewhat less too high in the center than it is too low at the edges. Integration over the considered bandwidth confirms this observation since one would expect 0.4 s-Hz
(bandwidth times target delay value) but the result was only 0.3911 s-Hz. So it is better to tune somewhat less peaky than one would expect when the MMSE criterion is used. This effect causes a higher deviation for calculations with fewer points, but at this level it is still acceptable. Furthermore it is not recommended to use only one ring for a target delay of 0.2 ns in this case since the ripple is quite high.

So there seems to be some limit of how much delay can be reached for a certain number of rings. When this limit is approached the delay ripple gets more and more. This is quite reasonable since the total area under the bell-shaped response is constant and thus independent of $\kappa$ for one period. When a $\kappa$ of one is set, the delay value is equal to $T$ according to (2.1). The period of the response is $1/T$ and thus is the total area under the bell shaped response is equal to 1 s-Hz. This area is also the absolute limit for the bandwidth delay product per ring. Therefore it can be concluded that the assumption from Section 4.1 about the number of points needed holds when the tuning is done with a target not too close to the theoretical maximum for the number of available rings.

### 4.2.2 Two rings

For two rings the results become more interesting, since more parameters influence the result and it is more difficult to obtain manually. For one ring the ripple became quite high at 0.2 ns. Therefore the first optimization was done with a target delay of 0.2 ns. All other parameters were kept the same except for the number of rings, which is now two, and the number of points in the optimization is now set to 10. The second optimization was done at 0.4 ns to test the algorithm in an area where more ripple was expected.

The resulting responses are shown in Figure 4.2. Note that the frequency axis is now a normalized one with respect to $T$ and its center is located at 1550 nm. The corresponding tuning results and metric values in Table 4.2.

<table>
<thead>
<tr>
<th>Target delay (ns)</th>
<th>$\kappa$</th>
<th>$\phi_1$</th>
<th>$\phi_2$</th>
<th>Metric value (ns²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.9470</td>
<td>-0.8663</td>
<td>0.8847</td>
<td>$1.1907 \times 10^{-7}$</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6507</td>
<td>-0.4065</td>
<td>0.4249</td>
<td>$9.6455 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

The response for 0.2 ns is very flat, since the ripple is hard to notice in the figure. For 0.4 ns the ripple is larger, but not as large as for 0.2 ns with only one ring even though it is just as close to the maximum delay for the number of rings used (double the number of rings double the delay). This emphasizes the strength of using more rings. In contrast with the one ring case is the area under the curves not significantly
Figure 4.2: Optimization result for two rings tuned at 0.2 ns delay (solid line) and 0.4 ns (dashed line). The bandwidth in which the optimization took place is 2 GHz and is marked by the dashed axes.

different from the bandwidth target delay product. This is as one would expect since the reasoning behind the one ring case does not hold in here.

The symmetry for tuning two rings as described in Section 2.2 becomes clear from the results. The $\kappa$s are always the same for both rings and the $\phi$s are at equal distance from the center. When checking the last statement it should be kept in mind that the center is at a $\phi$ of 0.0092. The most interesting part of the result is how far the rings should be tuned away from the center optimally, which is difficult to determine manually. In Chapter 5 information from the algorithm will be used to obtain a rule of thumb for this parameter to tune optimally.

As mentioned in Section 4.1 there should be more than one global minimum in the MMSE function. This is indeed the case. In the described optimizations, the initial estimates were set at -1 for the first ring and 1 for the second ring. When those two numbers are interchanged, the phases in the final solution also interchanged and thus the other global minimum was reached. Some checks with changing the initial estimates confirm the assumption that the minimum found is indeed the global minimum. There was only one local minimum in the function, which could be reached when both rings are set at zero phase for the initial estimate. Then the NLP solver gives a solution in which both rings are placed in the center with the same $\kappa$.

The next assumption to verify is the number of points needed. Therefore the
calculations were repeated with 10,000 points, which could be considered as the actual solution. It resulted in a 0.1% higher value for $\kappa$ and 0.33% for $\phi$ at 0.2 ns. For 0.4 ns the differences were 1.66% and 2.11% respectively. Such a small difference will be hard to notice in a graph and therefore the metrics for both tunings with 10,000 points are compared. The actual solution has a metric of 0.5100 and the 10 point solution 0.6697. So the inaccuracy for 0.4 ns is larger than for 0.2 ns. This confirms the conclusion from the one ring case that there are more points needed when the tuning is done closer to the theoretical maximum for the number of available rings. The differences are larger than in the one ring case from which it can be concluded that more points are needed when more than one ring is used. This is also reasonable since there are more parameters to optimize for and thus more points should be needed.

### 4.2.3 Three rings

When tuning for three rings another problem comes in. Recall the tree structure of Figure 1.3. In there it can be seen that the section of two rings is used twice: it is the only delay section for output 3 and forms output 4 together with a one ring section. So tuning the three rings also influences the output with two rings. To emphasize these effects a comparison is made between tuning for three rings at 0.6 ns without taking the two ring section output into account and the sum of an optimization for one ring at 0.2 ns and two rings at 0.4 ns, which should give a 0.6 ns response as result.

The resulting responses are shown in Figure 4.3a-c. Note the differences between separate tuning and tuning for three rings at once. The corresponding $\kappa$s and $\phi$s are presented in Table 4.3.

<table>
<thead>
<tr>
<th>Tuning type</th>
<th>One ring $\kappa$</th>
<th>One ring $\phi$</th>
<th>Two ring $\kappa$</th>
<th>Two ring $\phi$s</th>
</tr>
</thead>
<tbody>
<tr>
<td>separate</td>
<td>0.7814</td>
<td>0.0092</td>
<td>0.6507</td>
<td>-0.4065 and 0.4249</td>
</tr>
<tr>
<td>at once</td>
<td>0.6478</td>
<td>0.0092</td>
<td>0.5749</td>
<td>-0.5475 and 0.5749</td>
</tr>
</tbody>
</table>

The assumption that 10 points is enough still holds for three rings since the parameter with the largest deviation from the actual solution was only 1.57% away. Changing the initial values revealed that the described solution indeed was the global minimum. The only local minimum could be reached when all rings are set in the center of the spectrum initially. Then the NLP solver gives a solution in which all rings are placed in the center with the same $\kappa$.

As can be seen from the figure the solid line is optimal for three rings but the other outputs have unacceptable ripple. For the dashed line the first two outputs are optimal but the third one is significantly below the target at the edges. So the optimal way to tune for three rings is to put the maxima of the two ring section further outside the
center so that there is a dip in the target region. The dip should then be compensated by a one ring response which is more peaky than in the normal one ring case.

This confirms the statement from Section 2.2 that there is some kind of trade-off between tuning the three ring output optimally and the others. This emphasizes the need to optimize the whole chip at once, which will be done in the next subsection. An argument against this could be that the numbers of the example are not chosen very well since 0.2 ns is somewhat outside the tuning range for one ring. Therefore the optimization is repeated for 0.5 ns for three rings, consisting of 0.4 ns for two and 0.1 ns for one.

The response for 0.5 ns is shown in Figure 4.3d. From there it can be seen that
4.2. Results for delay response tuning

summing 0.1 ns and 0.4 ns still has an acceptable ripple, which is comparable with the optimized 0.4 ns response. However the solid line which is optimized for three rings has a much flatter response. So there still is a trade-off despite the fact that the one ring section is not close to its maximum capacity anymore.

Later on, in Section 4.5, another comparison between tuning all rings at once and tuning separately will be made according to the power metric.

4.2.4 Tuning a 1×4 optical beam forming network chip at once

This subsection will treat the tuning of a whole chip at once. This is done to find the optimum of the previously discussed trade-off and to check whether there is such a trade-off. The implementation is a quite easy step when starting from the three ring situation. Since the only thing which has to be done is to add the two ring objective function to the three ring one. The objective function of the one ring section does not need to be added for tuning the complete chip at once. This can be seen in Figure 1.3, since there are two separate one ring sections for output 2 and 4. So when tuning a complete ring the results from Subsection 4.2.1 can still be used as part of the optimal solution.

The rings for output 3 and 4 can then be optimized at once. The results are presented in Table 4.4. The one ring section was tuned in the center with a slightly more peaky tuning than the one ring section tuning of Subsection 4.2.1. This is one of the advantages of taking a look at tuning all rings at once, since now a different value for the one ring section for output 4 is chosen. Although the deviation is small, it is in contradiction with the manual tuning rules from Section 2.2, which assumed that both one ring sections should be tuned the same way.

<table>
<thead>
<tr>
<th>Target delays (ns)</th>
<th>Two ring $\kappa$</th>
<th>Two ring $\phi$s</th>
<th>Third ring $\kappa$</th>
<th>Metric value (ns²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6 and 0.4</td>
<td>0.6235</td>
<td>-0.4302 and 0.4486</td>
<td>0.7777</td>
<td>5.6 $\cdot$ 10⁻³</td>
</tr>
<tr>
<td>0.5 and 0.4</td>
<td>0.6505</td>
<td>-0.4094 and 0.4278</td>
<td>0.9831</td>
<td>2.0 $\cdot$ 10⁻³</td>
</tr>
</tbody>
</table>

Just like in the previous subsection the optimizations were repeated for 0.5 ns as three ring delay and 0.4 ns as two ring delay. Again the one ring section was tuned in the center with a $\kappa$, which is also slightly more peaky then the separate one ring tuning. The responses are shown in Figure 4.4.

From the results it can be concluded that the optimal tuning for the whole chip is somewhere in between the tunings for two rings and one ring separately and three rings at once. However it is closer to the two and one ring solutions than to the three ring solution.
Figure 4.4: Group delay response optimized for the complete chip at once. Both the three ring outputs (solid lines) and two ring outputs (dashed lines) are shown. In (a) the target delay was 0.5 ns for the three ring output and in (b) it was 0.6 ns. In both cases the target for the two ring output was 0.4 ns.

4.3 Results for phase response tuning

4.3.1 Method used for phase tuning

The phase will also be used as a possible criterion to tune the ORRs as mentioned in Chapter 3 with equation (3.13). The formula for the phase response of (3.12) is repeated in here

\[
\psi(f) = \arctan\left(\frac{\sin(2\pi f T + \phi_l)}{\sqrt{1 - \kappa_l} - \cos(2\pi f T + \phi_l)}\right) - \\
\arctan\left(\frac{\sqrt{1 - \kappa_l} \sin(2\pi f T + \phi_l)}{1 - \sqrt{1 - \kappa_l} \cos(2\pi f T + \phi_l)}\right)
\]  

(4.2)

A plot of the phase response is shown in Figure 4.5. In the figure it can be seen that there are several phase jumps in the function. These are caused by the first term in (4.2). Since the cosine term in the denominator can become smaller than the \(\sqrt{1 - \kappa_l}\), the fraction goes from minus infinity to plus infinity as the cosine approaches \(\sqrt{1 - \kappa_l}\) from above and eventually becomes smaller than that. This causes the arctangent function to make an upwards jump of \(\pi\). When the cosine is on the rising edge one could think that there should be a jump of \(\pi\) downwards, but this is not the case since the sine in the numerator is then negative.

This phenomenon makes comparison with an ideal linear phase response quite difficult and therefore a trick should be applied. Converting to the Z-domain could be an option and according to the results of [2] it gives a phase response without jumps
in the normalized domain. However conversion of the system to the Z-domain and calculation of the phase response from that is unnecessarily complex.

Since the phase jump is always in the positive direction and always of magnitude $\pi$, the data can be modified easily to remove the jumps. Furthermore it is easier to do this while working with normalized frequencies since else all periods of the function before the considered bandwidth should formally be modified too to prevent a wrong offset in the phase function.

Symmetry can also be exploited easier when working in the normalized domain. It is justified to use it since delay tuning already proved that the optimal solutions are indeed symmetric, so there will not be loss of generality when symmetry is implemented before the optimization starts. When the center of the optimization bandwidth is set to a normalized frequency of zero several phase parameters can be fixed. The phase of the one ring sections can be fixed at zero and for the two ring sections the phases should lie at equal distances from the center, so $\phi_1 = -\phi_2$. With these fixed parameters, the optimization now only has to be carried out in half of the bandwidth from the center frequency to one edge. This is justified because the other half is equal because of symmetry.

To get better insight in the differences between the results of phase and delay tuning, the same bandwidth and target delay values as in the previous section should be used. Therefore the bandwidth of 2 GHz should be converted into a normalized bandwidth. The normalization is done with respect to $T$ and the new normalized
bandwidth becomes 0.16. The target delay value is more difficult to convert since now a phase is considered. The delay was derived from the phase by taking the derivative with respect to frequency and multiplication by $-1/2\pi$. For 0.2 ns this means the slope of the phase should be $-1.26 \cdot 10^{-9}$ rad Hz$^{-1}$ (see also (3.13)) and after normalization the slope should be -15.708. A direct formula to calculate the slope in the normalized phase response is thus given by

$$\frac{d\psi}{d\Omega} = \frac{D \cdot 2\pi}{T} \tag{4.3}$$

in which $\Omega$ is the normalized frequency, $D$ the target delay and $T$ the roundtrip time of the ring. The starting point of phase responses with normalized frequency is always at zero. This can be seen from (4.2) since the one-ring sections should have a phase of zero, meaning that the sine terms are all zero too when $n = 0$. In the two ring sections the phases of the individual rings cancel out because of symmetry. Strictly spoken the arctangent should however be replaced by the complex argument operation according to the theory in [2] and then it would start at $\pi$, but using the normal arctangent is also possible in this research. Using this last derivation about the starting point, the ideal phase response is exactly known.

### 4.3.2 One ring

Now there is enough information to implement the criterion of (3.13) into the NLP solver. The first situation considered is one ring with target delays of 0.1 ns and 0.2 ns. The optimizations were done with 10 points in accordance with Section 4.1. As explained the phases are fixed at zero so there will not be a result for them from the optimization. The results for the $\kappa$s are shown in Table 4.5 together with the corresponding metric values. The responses are shown in Figure 4.6.

<table>
<thead>
<tr>
<th>Target delay (ns)</th>
<th>$\kappa$</th>
<th>Metric value (rad$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9868</td>
<td>1.63 $\cdot$ 10$^{-5}$</td>
</tr>
<tr>
<td>0.2</td>
<td>0.7903</td>
<td>5.40 $\cdot$ 10$^{-3}$</td>
</tr>
</tbody>
</table>

From the figure it can be seen that the response for 0.1 ns is nearly perfect just as the objective function suggests (see the metric value in the table). For 0.2 ns, the error becomes more visible. It depends on the application whether or not this error is still acceptable. The result of 0.1 ns with 10 points was only 0.01% away from the actual solution and for 0.2 ns it was 0.33%. So the statement about the number of points needed from Section 4.1 still holds in case of phase tuning. Some checks with
4.3. Results for phase response tuning

The initial value revealed that there were no local minima in the function, which is of course quite obvious with only one ring.

4.3.3 Tuning a 1×4 optical beam forming network chip at once

The individual optimizations for two and three rings are not useful since the two ring section is used for both outputs as explained in Section 4.2 about the delay tuning. Therefore the next step for tuning the complete chip is to optimize for the two and three ring outputs at once. The one ring tuning can be taken from the previous subsection again. The tuning of the whole chip is done by adding the metrics for the two and three ring outputs just like in Section 4.2 for delay tuning.

The tuning results are presented in Table 4.6 together with the corresponding metric values. The results for 0.6 and 0.4 ns for the outputs are shown in Figure 4.7 and for 0.5 and 0.4 ns in Figure 4.8.

From the figures the errors are hardly visible. This is because of two reasons. The first is that the error is harder to see because of fact that it should be compared with a linearly decreasing line instead of a certain value on the Y-axis. The second is that the error is indeed less than in the delay tuning case as can be seen from the values of the

Figure 4.6: Phase response plot optimized for one ring. The solid lines represent the actual responses for 0.1 and 0.2 ns, while the dashed lines are the theoretical ideal responses. The optimization bandwidth was from 0 to 0.08 on the normalized frequency axis.
objective functions. This can be explained with the relation between delay and phase, which is an integration to go from delay to phase. Then errors can cancel out in the phase domain. For example, when the delay in the center is too high while the edge is way too low, then the integration compensates the way too low part on the edges with some kind of reserve which was build up in the center.

This also emphasizes the need to tune for phases instead of delays, since some different effects occur. Besides the stressed effect about error cancellation there is on the other hand an effect that when the delay is slightly too high in the complete bandwidth, then the error starts to build up because of integration. However this effect does not occur very often because delays are most of the time fluctuating around the target value in the target bandwidth. Since the performance is directly dependent on the phase error, as explained in Chapter 3, these effects should be taken into account.
4.4 Results for power tuning

4.4.1 Method used

The last criterion is based on (3.15) and relies solely on the output power from the system. This is in fact the final metric for the system performance. However the phase error should not become too large for reasons already mentioned in Chapter 3. These reasons include stronger side lobes, higher NRNL and gain reduction. A maximum of $\pi/16$ was proposed, and this should be checked once the optimization has taken place.

When taking a closer look at (3.15) some factors which are not dependent of the ORR tunings can be observed. These factors are $a_m$ and $|H_m(f_o + f_{IF,n})|$ and they can

---

**Figure 4.8:** Phase response plot of two and three ring output optimized for the complete chip at once for 0.4 and 0.5 ns. The solid lines represent the actual responses for the outputs, while the dashed lines are the theoretical ideal responses. The optimization bandwidth was from 0 to 0.08 on the normalized frequency axis.

Therefore phase tuning is better than delay tuning.

The deviations from the actual solution for 10 points were still acceptable with at maximum 0.89% for 0.6 and 0.4 ns and 0.80% for the tuning at 0.5 and 0.4 ns. However care should be taken with the local minima problem since with 10 points in the objective function, the initial values needed to be set very close to the solution to prevent the solver to end at a local minimum.
be replaced by ones in (3.15), resulting in the following expression

\[
\mu = \sum_{n} M - \left| \sum_{m=1}^{M} \exp\left(j(\psi_{\text{total},m}((f_0 + f_{\text{IF},n})) + D_m (f_0 + f_{\text{IF},n}))\right) \right|^2 \tag{4.4}
\]

Since the method of implementing the phase error was already explained in Section 4.3, all parameters to implement the problem into the optimizer are now known. Another difference with the previous criteria is that the mutual relations between different outputs are now taken into account. This means that looking at individual outputs is meaningless and therefore only results for the whole chip at once are presented in this section. Of course the parameters of the two one ring sections will be treated as separate variables for the optimization. Also the symmetry as described in Section 4.3 will be exploited again.

### 4.4.2 Results

For power tuning the same target bandwidth and delays as for phase tuning were used. However this time also the one-ring output target can be varied whilst keeping the other outputs equal. Therefore more interesting combinations were possible and the results are presented in Table 4.7.

<table>
<thead>
<tr>
<th>Target delays (ns)</th>
<th>One ring (\kappa)</th>
<th>Two ring (\kappa)</th>
<th>Two ring (\phi)</th>
<th>Third ring (\kappa)</th>
<th>Total error (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6,0.4,0.2</td>
<td>0.7898</td>
<td>0.6689</td>
<td>0.3990</td>
<td>0.7898</td>
<td>-61.73</td>
</tr>
<tr>
<td>0.6,0.4,0.1</td>
<td>0.9868</td>
<td>0.6450</td>
<td>0.4189</td>
<td>0.7898</td>
<td>-68.21</td>
</tr>
<tr>
<td>0.5,0.4,0.2</td>
<td>0.7898</td>
<td>0.6945</td>
<td>0.3714</td>
<td>0.9867</td>
<td>-68.56</td>
</tr>
<tr>
<td>0.5,0.4,0.1</td>
<td>0.9868</td>
<td>0.6697</td>
<td>0.3994</td>
<td>0.9867</td>
<td>-92.97</td>
</tr>
</tbody>
</table>

From these results it can be seen that when the one ring output is tuned at 0.2 ns, which results into too high delay at the center and too low at the edges, that the two ring section is also tuned with more delay at the center. So the target delay of the one-ring output does affect the other outputs even though it does not share ORRs with those outputs. This agrees with the theory of Section 3.3, which was the main reason to derive the power criterion for tuning.

When looking at the results for the total error one can see that the total error and thus the loss of power is very small over the 10.000 points considered. Even the worst performing target delays have an average power loss of -61.73 dB which is not significant.
The algorithm took more calculation time than previous methods. While the phase tuning took 8 seconds at maximum for 10,000 points, the power tuning took 38 seconds. These times are wall clock times on a AMD Athlon 64 processor rated 3200+ with 1 GB of RAM. Another difference with the other tuning methods was the presence of many local minima, especially when few points were used. An initial estimate within 0.05 for the $\kappa$s and $\phi$s was needed to reach the correct solution. In most cases when the algorithm was not on its way to the global solution it did not converge resulting into high calculation times. Mostly it was limited by the maximum number of iterations in \textit{fmincon}.

### 4.5 Comparison of tuning methods

So far three methods for delay tuning have been carried out. The results were satisfactory since the optimal solution could be reached in any case. However in the scope of this research the algorithms were needed to derive rules of thumb. Therefore a choice has to be made which algorithm is best suited to derive those rules.

The first one, delay tuning, was most intuitively appealing and easily comparable with prior research results. However in the formulas (3.2) and (3.3) it can be seen that the delay error does not play a direct role in the output power of the optical detector. It is the absolute value of the addition of the complex phase vectors, which determines the output power. Also the other negative side effects of improper tuning, like the rise of sidelobes and the NRNL do not directly depend on the delay error. In fact they do depend on the phase error. This phase error can be derived from the delay by integration and multiplication with a factor.

So therefore another criterion was implemented in the optimization algorithm. It uses the formula for the phase response and determines the phase error from it which should of course be minimized. A major advantage of phase tuning is that the negative side effects of improper tuning are minimized. The method is also one step closer to maximizing the output power, but still not there since the power is determined by the absolute value of the addition of complex phase vectors.

From this it is a small step to go to direct optimization of the power. Theoretically this is the best way to tune the chip when the negative side effects (like side lobes etcetera) of improper tuning are still kept in mind. So the algorithm is only optimal for the height of the main lobe. However the algorithm for power tuning has its drawbacks. Many points and an accurate initial estimate were needed to reach the correct solution. Also the different results for the $\kappa$s and $\phi$s of output 3 and 4 when output 1 is changed make it difficult to derive rules of thumb.

The different metric values for all solutions of the different methods are shown in Table 4.8. This is to compare the results of the different optimization methods. For
the delay and phase metrics the values of the one ring output and combined two and three ring output are added, since the same happens with power tuning.

**Table 4.8**: Comparison between different tuning methods by determining all metric values of the results for each method with each other. All metric values are averaged values per point over 10,000 points.

<table>
<thead>
<tr>
<th>Target delays (ns)</th>
<th>Tuning type</th>
<th>Delay metric (ns²)</th>
<th>Phase metric (rad²)</th>
<th>Power metric (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6,0.4,0.2</td>
<td>Delay</td>
<td>7.4819×10⁻⁴</td>
<td>9.4097×10⁻⁴</td>
<td>-56.17</td>
</tr>
<tr>
<td>0.6,0.4,0.2</td>
<td>Phase</td>
<td>8.0809×10⁻⁴</td>
<td>6.9570×10⁻⁴</td>
<td>-59.26</td>
</tr>
<tr>
<td>0.6,0.4,0.2</td>
<td>Separate phase</td>
<td>1.1555×10⁻³</td>
<td>9.0970×10⁻⁴</td>
<td>-60.13</td>
</tr>
<tr>
<td>0.6,0.4,0.2</td>
<td>Power</td>
<td>1.0796×10⁻⁴</td>
<td>9.4929×10⁻⁴</td>
<td>-61.73</td>
</tr>
<tr>
<td>0.6,0.4,0.1</td>
<td>Delay</td>
<td>3.4455×10⁻⁴</td>
<td>4.3716×10⁻⁴</td>
<td>-64.35</td>
</tr>
<tr>
<td>0.6,0.4,0.1</td>
<td>Phase</td>
<td>4.1841×10⁻⁴</td>
<td>2.6740×10⁻⁴</td>
<td>-66.90</td>
</tr>
<tr>
<td>0.6,0.4,0.1</td>
<td>Separate phase</td>
<td>7.4683×10⁻⁴</td>
<td>4.8139×10⁻⁴</td>
<td>-61.93</td>
</tr>
<tr>
<td>0.6,0.4,0.1</td>
<td>Power</td>
<td>3.9236×10⁻⁴</td>
<td>2.9633×10⁻⁴</td>
<td>-68.21</td>
</tr>
<tr>
<td>0.5,0.4,0.2</td>
<td>Delay</td>
<td>5.0966×10⁻⁴</td>
<td>6.0624×10⁻⁴</td>
<td>-61.21</td>
</tr>
<tr>
<td>0.5,0.4,0.2</td>
<td>Phase</td>
<td>5.4878×10⁻⁴</td>
<td>4.6814×10⁻⁴</td>
<td>-64.68</td>
</tr>
<tr>
<td>0.5,0.4,0.2</td>
<td>Separate phase</td>
<td>5.5762×10⁻⁴</td>
<td>4.6880×10⁻⁴</td>
<td>-65.00</td>
</tr>
<tr>
<td>0.5,0.4,0.2</td>
<td>Power</td>
<td>8.2341×10⁻⁴</td>
<td>7.0732×10⁻⁴</td>
<td>-68.55</td>
</tr>
<tr>
<td>0.5,0.4,0.1</td>
<td>Delay</td>
<td>1.0602×10⁻⁴</td>
<td>1.0243×10⁻⁴</td>
<td>-82.50</td>
</tr>
<tr>
<td>0.5,0.4,0.1</td>
<td>Phase</td>
<td>1.4015×10⁻⁴</td>
<td>3.9827×10⁻⁵</td>
<td>-92.30</td>
</tr>
<tr>
<td>0.5,0.4,0.1</td>
<td>Separate phase</td>
<td>1.4899×10⁻⁴</td>
<td>4.0494×10⁻⁵</td>
<td>-91.02</td>
</tr>
<tr>
<td>0.5,0.4,0.1</td>
<td>Power</td>
<td>1.4344×10⁻⁴</td>
<td>4.1451×10⁻⁵</td>
<td>-92.97</td>
</tr>
</tbody>
</table>

As mentioned in Subsection 4.2.3 another comparison was planned to be made between separate tuning of the one and two ring DEs for output 3 and 4 and tuning them at once. The results of separate tuning are also presented in Table 4.8. The phase criterion was used in separate tuning because it was already considered better than delay tuning. Separate power tuning is not possible, since it fully relies on the mutual relationships between the outputs.

Compared with the normal phase tuning algorithm its performance is slightly better when the delays are linearly increasing or close to that when looking at the power metric. However the performance is clearly worse with 5 dB more power loss when the delays are not linearly increasing. Although -60 dB does not seem to be a high power loss, things will become worse when more difficult target delays have to be tuned. On the other hand separate tuning is easier than doing it at once and especially the rules of thumb will be easier to derive. Therefore it will depend on the demands of the final system whether the benefits of separate tuning compensate for the possible loss
of performance. Also the higher phase errors of the separate method can strengthen the negative side effects, since they all depend on $\Delta \phi_{\text{max}}$ (see Section 3.1). This should of course also be kept in mind.

From the metric values it becomes clear that delay tuning has the worst performance (the delay criterion was not considered to be useful) and that phase and power tuning do not differ much from each other. When looking at the power metric values, it can be seen that the power loss of the phase optimized tunings is not significant. Even when all the power loss of the largest value of phase tuning is concentrated in one point, only $9.8577 \cdot 10^{-3}$ W will be lost when the maximum power would have been 1 W, which is a power loss of less than 1% (for the phase criterion this would be significant, since when all errors for the worst case are concentrated in one point it would be $9.4929 \text{ rad}$). This worst-case scenario is of course not possible when 10,000 points are taken and therefore it can be concluded that power tuning does not have a practical meaning. So the theoretical advantage of power tuning does not match the practical disadvantages when compared to phase tuning, because of these insignificant differences. Therefore the final conclusion from this comparison is that phase tuning will be used as a starting point in Chapter 5 for the derivation of rules of thumb.
Chapter 5

Derivation of rules of thumb

5.1 One ring

Now that there is an algorithm to acquire the optimal settings, the rules of thumb can be derived. To start simple, the one ring tuning of output 2 from Figure 1.3 is treated first. The phase tuning algorithm will be used with normalized bandwidths and delays. The result of the derivation will be an optimal value for $\kappa$. Therefore the target delay and bandwidth for each application should be calculated back to a normalized version in order to use this. For the delay ($D$) the following formula can be used:

$$D_{\text{norm}} = \frac{D}{T}$$  \hspace{1cm} (5.1)

with $T$ the roundtrip time of the ring and for the bandwidth ($B$)

$$B_{\text{norm}} = B \cdot T$$  \hspace{1cm} (5.2)

Since the phase tuning method only gives a $\kappa$, the $\phi$ should be calculated afterwards. This $\phi$ should then be added to all the rings after determining the $\kappa$s and $\phi$s using the rules of thumb. The calculation is quite straightforward using the theory of the previous chapter, where it was concluded that the $\phi$ should be set to such a value that the maximum of the delay curve is at the center frequency ($f_c$). Since the delay response curve repeats every FSR, multiples of $2\pi$ should be substracted as is done in the following formula for the phase:

$$\phi = 2\pi [f_c \cdot T - \text{round}(f_c \cdot T)]$$  \hspace{1cm} (5.3)

where round() is a rounding operation to the nearest integer. Using these normalized parameters, the optimal solution could have been calculated easily with the phase tuning algorithm. A great advantage of using normalized parameters is that the rules of thumb can be used for every application no matter what frequency or bandwidth is used. Also any value of $T$ can be used in the system.
To derive a rule of thumb a polynomial fit should be made of $\kappa$ as a function of the normalized delay and bandwidth. Here a problem arises since $\kappa$ is a function of two variables: the bandwidth and delay. Since the bandwidth is fixed in most applications, $\kappa$ should be derived as a function of delay for fixed bandwidth values. A plot of $\kappa$ as function of the delay is given in Figure 5.1. In this plot 120 points are used between the start and stop delay value. The stop delay value is chosen at a normalized value of 2.5. This is because the error becomes unacceptably high at that point, as will be shown in Section 6.4.

![Figure 5.1: The optimal $\kappa$ as a function of the normalized delay for different bandwidths.](image)

The most straightforward solution to put this into the system is a lookup table. Then there is a trade-off between accuracy and memory requirements. Another solution is to derive a formula which approximates the optimal curve. However there is no formula which satisfies all curves for the different bandwidths as can be seen from the figure. The curves are not linear but can be very well approximated by a second order polynomial of the following form.

$$\kappa = a \cdot D^2 + b \cdot D + c$$

(5.4)

Because of the limit of 0.995 for $\kappa$ the derivative of the curve is discontinuous at a target delay of approximately 1.2. Therefore the polynomial should be fitted in the region from 1.2 to 2.5. The function `polyfit` of MATLAB is used to fit these polynomials and the results of that are presented in Table 5.1.
Table 5.1: Results of polynomial fitting for the one ring output.

<table>
<thead>
<tr>
<th>Target bandwidth (ns)</th>
<th>$a$ (ns$^{-2}$)</th>
<th>$b$ (ns$^{-1}$)</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>-0.0186</td>
<td>-0.0724</td>
<td>1.1082</td>
</tr>
<tr>
<td>0.15</td>
<td>-0.0251</td>
<td>-0.0623</td>
<td>1.1052</td>
</tr>
<tr>
<td>0.25</td>
<td>-0.0370</td>
<td>-0.0465</td>
<td>1.1030</td>
</tr>
</tbody>
</table>

After using these polynomials all values larger than 0.995 should be set back to 0.995. The polynomials are fitted using the curves with 120 points. More points are not needed since the function is very smooth.

Figure 5.2: The curve-fitted polynomial (solid line) and optimal $\kappa$ (dashed line) as a function of the normalized delay for a normalized bandwidth of 0.25.

The largest error occurs for the $B = 0.25$ curve which is shown in Figure 5.2. From the figure, the error of the approximation does not appear to be very large. The largest error is 0.0036 at a bandwidth of 0.25. At that point the approximation was 0.7554 while it should have been 0.7590. The metric value for the total squared phase error over the entire bandwidth increased from 5.9445 to 6.0039, which is a 1.00% increase. The relative increase is higher in the region with low target delay, for example at 1.2 the $\kappa$ was 0.0035 too high which resulted in an increase of 0.0109 to 0.3908. However that is in a region of less interest since the errors are much smaller in there. The errors are considered to be small enough and therefore third or fourth order polynomials were not tried. For a third order polynomial the maximum deviation of the $\kappa$ was 0.0010
and for fourth order it was 0.0002. So when the demands are high it could be decided
to use third or fourth order. However it should be kept in mind that it is not useful
to have such a high precision when other parts of the system don’t. When the heating
voltages are steered by an 8 bit DA converter for example, it makes no sense to have
64 bit precision in the calculated $\kappa$s.

5.2 Two and three rings

Derivation of rules of thumb was quite easy for the one ring case. However since
the two and three ring tunings depend on each other, this one will not be that easy.
The most simple solution would be to ignore the problem, which would be possible
in some cases as described in Section 4.5. The procedure to derive rules of thumb
would not need an assumption about the mutual relationships then. At the end of
this section this method will be described. However when the problem is taken into
account an assumption about the mutual relationships has to be made in order to reach
an unambiguous solution.

Linear increasing delays are the best as starting point for an approximation and
therefore the $\kappa$s and $\phi$s will be derived from a target delay, which is linear increasing
from the two to three ring output. The starting point for the derivation will be at
normalized delays of 2.0 and 3.0 and it will continue until 5.0 and 7.5 using 241 points.
Fixed normalized bandwidths of 0.05, 0.15 and 0.25 will be used again.

When the system has to be tuned in a way that the delays are not linearly increasing,
the tuning algorithm has to take the $\kappa$s and $\phi$s for the two ring section as if it were
linearly increasing. For the third ring a $\kappa$ should be taken which corresponds to the
correct difference between the two and three ring output. When for example 3.0 and
4.5 are taken as starting point and the actual delay should be 5.0, the metric value
becomes 0.0460 instead of the 0.0313 in case of direct optimization for those values.
This was for the case when the third ring has to be tuned at a higher delay then linear,
when 3.0 and 4.0 are taken with the same starting point the metric rises from 0.8118
to 1.6310. Because the third ring can be tuned at a higher and lower delay than linear,
a linear increasing delay as starting point is good as a starting point.

When these errors are unacceptable a two-dimensional lookup table can be created
which gives the correct $\kappa$s and $\phi$s as a function of both the two and three ring target
delays. Of course this is at the cost of a lot of memory since the size of the table would
almost square (combinations where [output 4 - output 3] $\leq T$ are not possible).

While calculating the ideal $\kappa$s and $\phi$s another problem arises. The initial estimate
needs to be adapted when the target delays change. This was not a problem in the one
ring case since there were no local minima. However when optimizing for two and three
rings at once there are local minima and especially in the low target delay regions an
accurate initial estimate is needed. This is because \( \kappa \)s around the maximum of 0.995 are needed to achieve these small delays, which in their turn cause the bell-shapes to be very flat resulting in minor effects of \( \phi \) changes. To solve this problem the initial estimate will be set equal to the previous solution while working through the set of normalized target delays.

![Figure 5.3: The optimal \( \kappa \) of both rings for the two ring section as a function of the normalized delay for different bandwidths.](image)

The results of the derivation for the ideal \( \kappa \)s and \( \phi \)s are shown in Figure 5.3, Figure 5.4 and Figure 5.5. The delays for the three ring output are only reached when the two ring output is tuned at 2/3 of the three ring value. That way the outputs are tuned linearly increasing. Another possibility would be to put the difference between the two and three ring output on the x-axis, which would result into a graph almost equal to that of the one ring tuning. However the linear increasing delay is more in consensus with Chapter 4.

Of course the data from the graphs can be put directly into a lookup table again, but just as in the one ring case, a polynomial is also possible. The coefficients for the different tunings together with their maximum deviation from the actual data are shown in Table 5.2.

The results for the \( \kappa \)s are quite accurate. The worst case occurs for \( B = 0.25 \) with two rings and the deviation in \( \kappa \) of 0.0050 results in an increase of the metric from 7.3286 to 7.5693. According to the power metric the loss summed over 1000 points raises from 0.0152 to 0.195. This is an acceptable error since the power loss is still
Figure 5.4: The optimal $\phi$ of both rings for the two ring section as a function of the normalized delay for different bandwidths.

Table 5.2: Resulting coefficients for polynomial fitting for the two ring $\kappa$ and $\phi$ and the three ring $\kappa$ together with the maximum parameter deviation.

<table>
<thead>
<tr>
<th>Tuning type</th>
<th>$a$ (ns$^{-2}$)</th>
<th>$b$ (ns$^{-1}$)</th>
<th>$c$ ($)</th>
<th>Maximum deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two ring $\kappa$, $B = 0.05$</td>
<td>0.6869-10$^{-2}$</td>
<td>-0.1536</td>
<td>1.2834</td>
<td>0.0034</td>
</tr>
<tr>
<td>Two ring $\kappa$, $B = 0.15$</td>
<td>0.5150-10$^{-2}$</td>
<td>-0.1537</td>
<td>1.2913</td>
<td>0.0040</td>
</tr>
<tr>
<td>Two ring $\kappa$, $B = 0.25$</td>
<td>0.1549-10$^{-2}$</td>
<td>-0.1546</td>
<td>1.3089</td>
<td>0.0050</td>
</tr>
<tr>
<td>Two ring $\phi$, $B = 0.05$</td>
<td>0.1542</td>
<td>-13.5861</td>
<td>0.3407</td>
<td>0.4130</td>
</tr>
<tr>
<td>Two ring $\phi$, $B = 0.15$</td>
<td>0.1530</td>
<td>-13.4175</td>
<td>0.3381</td>
<td>0.4723</td>
</tr>
<tr>
<td>Two ring $\phi$, $B = 0.25$</td>
<td>0.1484</td>
<td>-12.9308</td>
<td>0.3306</td>
<td>0.4733</td>
</tr>
<tr>
<td>Three ring $\kappa$, $B = 0.05$</td>
<td>0.6869-10$^{-2}$</td>
<td>-0.1536</td>
<td>1.2834</td>
<td>0.0030</td>
</tr>
<tr>
<td>Three ring $\kappa$, $B = 0.15$</td>
<td>0.6869-10$^{-2}$</td>
<td>-0.1536</td>
<td>1.2834</td>
<td>0.0032</td>
</tr>
<tr>
<td>Three ring $\kappa$, $B = 0.25$</td>
<td>0.6869-10$^{-2}$</td>
<td>-0.1536</td>
<td>1.2834</td>
<td>0.0034</td>
</tr>
</tbody>
</table>

insignificant (see also Section 4.5).

However the results for $\phi$ have an unacceptable error. This can be understood by taking a look at Figure 5.4. From the graph it is clear that the function is more similar to a function of the form $1/x^2$ than to a polynomial. Therefore the $\phi$s will be fitted to a function of the following form:
Figure 5.5: The optimal $\kappa$ of the third ring for the three ring section as a function of the normalized delay for different bandwidths. Note that these delays are for the three ring output.

$$\phi = a/(D - b)^2 + c/(D - d) + e \quad (5.5)$$

A second order approximation is used just as in the case of (5.4). This is again because the error is not expected to be large. Third or higher order can be used when the demands on the system are very high. Using this method the coefficients found are shown in Table 5.3.

Table 5.3: Resulting coefficients for the fitting method of (5.5) for the two ring $\phi$ together with the maximum deviation between the delay values of 2.5 and 5.0.

<table>
<thead>
<tr>
<th>Target bandwidth</th>
<th>$a$ (ns$^2$)</th>
<th>$b$ (ns)</th>
<th>$c$ (ns)</th>
<th>$d$ (ns)</th>
<th>$e$ (-)</th>
<th>Maximum deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = 0.05$</td>
<td>-1.0835</td>
<td>1.4085</td>
<td>1.3596</td>
<td>1.7187</td>
<td>0.0525</td>
<td>0.0018</td>
</tr>
<tr>
<td>$B = 0.15$</td>
<td>-0.8717</td>
<td>1.4615</td>
<td>1.1989</td>
<td>1.7448</td>
<td>0.1142</td>
<td>0.0015</td>
</tr>
<tr>
<td>$B = 0.25$</td>
<td>-0.6474</td>
<td>1.5183</td>
<td>1.0014</td>
<td>1.7722</td>
<td>0.2098</td>
<td>0.0027</td>
</tr>
</tbody>
</table>

The region of 2.0 to 2.5 has larger errors and is displayed in Figure 5.6. The first datapoint has an error of 0.0941, caused by the existence of two global optima. The approximation chose the first one and the algorithm the other one, but since they are both global optima, the approximated result has the same performance as the
optimal solution. In the region around a target delay of 2.3 the error is also higher with a maximum of 0.0155. However these errors do not have a large effect on the performance of the system, since the metric values in that region are much smaller than close to a target delay of 5.0. Furthermore the region could also be covered with only one ring. The same effects occur for $B = 0.15$ and $B = 0.25$.

![Figure 5.6: The optimal $\phi$s at $B = 0.05$ of the two ring sections together with the optimal approximation of the form in (5.5).](image)

As described at the beginning of this section it was also possible to ignore the problem that the two and three ring tunings depend on each other. Then curves would have been obtained for the two ring $\kappa$ and $\phi$ optimized solely for output 3. The same curve as for the one ring output can be used for the tuning of the third ring used for output 4. This method saves some steps in the derivation procedure and a two-dimensional lookup table is not needed for sure. However it could be at the cost of some performance.

### 5.3 Comparison between lookup table and fitted polynomials

Now that all parameters for the $1 \times 4$ OBFN are known, a lookup table can be compared with fitted polynomials. The size of the table will be 2 bytes per entry, since 1 byte is not precise enough with only 256 values and 2 is with 65536 possible values. When 256 points are used per parameter, the largest resolution of the table for the normalized
delay would be 0.0175 and that is for the table of the third ring $\kappa$. There are four parameters which should be stored in the memory of the chip, so the total amount of memory required is 2 kB: $256 \cdot 2 \cdot 4 = 2048$ bytes. This is 25% of the memory unit (FM25640 by Ramtron) chosen in [3], but there are memories up to 256 kB (AT27C020-90JU by Atmel) on the market for the same price as the one chosen in [3]. The latter device is One Time Programmable (OTP), but that should be enough for a lookup table of this kind. So a lookup table is possible for a 1×4 OBFN chip, but it is approaching the memory limits, since a different memory unit is probably needed to apply it.

The solution of a curve-fitted polynomial requires only 12 coefficients to be stored in the memory for the complete chip. According to [3] there is a microprocessor with a clock frequency of 30 MHz in the system, which should be capable of calculating the actual values of the $\kappa$s and $\phi$s easily within a millisecond. Therefore this solution is more realizable than the lookup table.

It is also important not to forget that both methods require the extra phase calculated in (5.3) to be added afterwards. Of course also the corrections of Figure 2.3 and 2.5 have to be applied afterwards.
Practical aspects of delay tuning methods

6.1 Expansion to $1 \times M$ optical beam forming network

The research was carried out for a $1 \times 4$ OBFN chip, but it is also useful to know how the principles can be expanded to $1 \times 8$ and finally $1 \times M$, with $M = 2^n$ and $n$ is the number of stages. This is because the final system is expected to be of a much higher dimension than $1 \times 4$.

To get a better understanding of what happens when the OBFN is expanded to 8 outputs a $1 \times 8$ OBFN is shown in Figure 6.1. When this figure is compared with Figure 1.3 it becomes clear that stage 2 and 3 contain two structures similar to the $1 \times 4$ OBFN. In stage 1 there is a branch with four rings and a branch without rings. So except for the first stage the two branches are identical, which makes the structure symmetric. Further expansion to $1 \times M$ is done with the same trick: place a copy of the previous structure under the current one and add a new stage in front of them, one without rings and one with $M/2$ rings.

The knowledge about the symmetry can be exploited in the tuning problem. All outputs of the two branches in the first stage should give as little ripple as possible, and therefore the tuning of the other stages should be independent of the first stage. So dips or peaks in the tunings of the other stages should not be compensated by the first stage since the upper branch does not contain any rings to do so. Therefore the four ring section (and all further $M/2$ sections) should be tuned to give optimal phase response independent of the other rings in the chip.

One could ask why the two ring section was tuned together with the third ring of output 4. That seems to be a contradiction with the tuning principle explained in the previous for four and more rings. However the first case was about combining an odd number of rings (one) with an even number of rings (two), while in the further tuning it is about combining an even number of rings with another even number of rings. Take
Figure 6.1: 1×8 binary tree OBFN (taken from [12])

a look at Figure 6.2 to get a better understanding of this.

Figure 6.2: Optimal tunings for 2 rings at 0.5 ns (dashed) and 4 rings at 1.0 ns (solid).

The dip in the center occurs in both responses as can be seen from the figure. Therefore it cannot be compensated. The peaks a little outside the center of the two ring response cannot be compensated either. This is because there are two rings for the four ring output in that area. Giving one of them a lower delay will inherently make the dip in the center or at the edge worse. So therefore a four ring section cannot compensate for the dips and peaks in a two ring section. The same reasoning holds for
the $M/2$ ring sections when further expanding to $1 \times M$ structures.

Another thing to keep in mind is that for every doubling of rings the size of the lookup table will also double just as the calculation time for polynomials. So for a certain size of the OBFN a choice should be made whether a lookup table or a fitted polynomial will be used. The choice will also depend on the processing time and memory requirements of the rest of the system.

### 6.2 Different response formulae

A problem which can occur while implementing the tuning algorithms is that the responses of the chip are not exactly following the formulae used in this research. The formula of (2.2) will now be used as an example how to adapt the tuning procedure to a different response formula. This is the response of an ORR with a loss of $\alpha$ dB for one roundtrip of the ring.

The implementation is quite straightforward. Simply put the new response formula in the function for $\mu$ and start optimizing again. The effect of taking this loss into account, however, was minor. According to [12], the loss was 0.55 dB/cm and with a ring circumference of 1.2 cm this results in an $\alpha$ of approximately 0.7. Using that value the $\kappa$s of both rings should be increased from 0.6614 to 0.6705 while the $\phi$s should be decreased from 0.4070 out of the middle to 0.4010. The resulting optimal response almost exactly matched the optimal response without losses.

So the algorithm works with different response formulae too. So when other parasitic effects are discovered they can be implemented in the optimization without any problems.

In this research it was assumed that the target delays are always larger than or equal to $T$, but it is also possible to use the algorithms for delays smaller then $T$. In the phase tuning algorithms, the $\phi$ should be fixed to $\pi$. That way the minimum of the response is exactly in the center and thus as flat as possible and symmetric. Now a higher $\kappa$ results into a lower delay value as can be seen from Figure 6.3

### 6.3 Fabrication offsets and unpredictable parasitic effects

Another class of problems are those which can not be predicted by formulae which is independent of fabrication process or any other time-variable parameter. These problems should be solved using an adaptive algorithm.

An example of a fabrication offset is the ring diameter ($T$). Some adaptive mechanism is needed which measures the offset and then changes the $\kappa$s and $\phi$s according to
Figure 6.3: Examples of tunings for delays smaller than $T$. Note that a complete FSR (12.5 GHz) is shown on the x-axis.

the values found in the lookup tables or calculated with the polynomials using the new $T$. The normalization procedure, equations (5.1)-(5.3), should also use the new $T$.

A good example of the unpredictable parasitic effects can be seen from the measurement results of [12], which are presented in Figure 6.4.

The first outputs until output 3 have quite smooth responses and their form is quite similar to the theoretical work from Chapter 4 and 5. Output 4 shows some asymmetry, which can be caused by improper tuning since these results are acquired while tuning manually, but it could be caused by parasitic effects too. The higher outputs have more and more irregular response curves and output 8 is the most extreme of these. The sharp and pointy delay response of this output can not be acquired by combining seven rings with responses of the kind used in Chapter 4. Therefore there have to be parasitic effects, which are currently unpredictable since there is no straightforward formula which captures this.

The effects differ over time and probably also between different copies of the chip. Therefore the solution should be an adaptive scheme, which changes the tuning in order to minimize the errors of these parasitic effects. These schemes to overcome errors due to fabrication offsets and other unpredictable parasitic effects still have to be made and therefore belong to the further research for the SMART project.
6.4 Number of rings required

The results of the optimization algorithms do not only contain the optimal values for the $\kappa$s and $\phi$s, but also the resulting metric value of the tuning. With this information a graph can be made of the metric value as a function of the target delay, for different numbers of rings. According to the graph a decision about the number of rings required for a certain delay value can be made. The graph for one, two, three and four rings is shown in Figure 6.5.

All graphs of the different numbers of rings stop when the metric value starts to rise steeply. The absolute maximum should be chosen in consensus with the demands on the system, but it should lie in the region where the graphs stop. The one ring curve is slightly different then the other ones because the curve is descending from a normalized delay of 1 until 1.15. This is because of the limit on $\kappa$ of 0.995 which causes the delay response to be slightly higher than $T$ at minimum.

Another important conclusion which can be drawn easily from this graph is that the tuning with more rings becomes more and more efficient. The total normalized delay per FSR is independent of $\kappa$ and $\phi$ and equal to 1. Therefore the theoretical maximum delay achievable at a bandwidth of 0.15 is 6.67 per ring. When the maximum sum of the squared error is set to 0.67, the limit is reached at a target delay of 2.84, which is 42.56% of the theoretical maximum. For four rings the limit is reached at 18.75, which is 70.31% of the maximum.
Figure 6.5: The total phase error over 1000 points as a function of normalized target delay with a fixed normalized target bandwidth of 0.15.

This is a substantial difference which should be taken into account while designing an OBFN chip, since rings could possibly be saved in the stages with four or more rings. The three ring error for example is 0.10 at a delay of 11.36, which is four times the maximum of one ring. So only three rings are necessary to continue the binary tree. The consequences of this propagate further in a higher dimensional OBFN. For example when a doubling of rings is done starting from three rings towards a 1×64 OBFN, only 24 rings are required in the first stage as opposed to 32 when the doubling started at four.
Chapter 7

Conclusions and Recommendations

7.1 Conclusions

The goal of the assignment was to design a subsystem which converts a desired delay value into a set of optimal $\kappa$s and $\phi$s. This goal was achieved without severe memory or processing requirements of the algorithm. The solution was also very robust to different types of the system, bandwidths and frequencies since it uses normalized bandwidths and target delays. So both the SMART-project and the the Broadband Photonic Beamformer project can use the solution found in this assignment.

The first step towards the final solution was to derive optimality criteria. From that three algorithms were derived which used an NLP solver to find the optimal solution. From the results of those algorithms it could be concluded that the one which minimized the square of the phase error was the best. There was no significant difference in output power with the algorithm which focused on it, but it was much easier to implement and derive rules of thumb from. Also the other negative side effects of improper tuning, like the rise of sidelobes and the NRNL, are minimized with phase tuning since they directly depend on the phase error. The delay tuning was the most intuitive one, but it was the worst performer.

The last step was to derive rules of thumb from the data generated with the phase tuning algorithm, since that one was considered as the best. This was needed because a complete NLP solver would pose too much requirements on the processor and memory of the system. The most simple solution is a lookup table which just stores all values of the $\kappa$s and $\phi$s for a certain target delay and bandwidth. It requires 2 kB for a 1×4 OBFN with a delay resolution of 0.0175 and for larger OBFN chips the size of the lookup table will double for every doubling of rings. The presented solution for the problem of the dependancy between output 3 and 4 took linearly increasing delays as starting point for derivation of the lookup table. However depending on the demands of the final system another choice can be made. The choices include ignoring the problem by just optimizing the one and two ring sections seperately, chosing a different starting
point and using a two-dimensional lookup table.

The other solution was to fit the curves of the $\kappa$s and $\phi$s with a polynomial function. That way only the coefficients need to be stored in the memory, which saves a lot of memory. This is at the cost of some processing time to calculate the values and some inaccuracy which causes an increase of the total error of approximately 1%. Just like in the lookup table option a solution should be chosen for the problem of the dependancy between output 3 and 4. The same options are possible as in the lookup table case. Since the lookup table for the $1\times4$ OBFN was already approaching the memory limits of the system, the curve fitted polynomials are considered as the best solution. For different types of system, however, another choice can be made. This depends strongly on the required accuracy and the other system components. It makes no sense to have a very precise value for $\kappa$ when there’s only an 8 bit DA converter steering it.

When applying this method it should also be kept in mind that it works with normalized parameters. So first the target bandwidth and delays have to be normalized with respect to $T$. After the $\kappa$s and $\phi$s are determined the phase offset of the entire response, which was calculated with the normalization, should be added to the $\phi$s. Also phase corrections for conversion to a new type of ORR and for the heat crosstalk effect should be applied. When these corrections are made the voltages can be calculated.

Another useful result from the optimization algorithms was the total error of the optimal solution. When plotting that as a function of the target delay for different numbers of rings a choice can be made when to use a certain amount of rings. An important conclusion from it was that theoretically only three rings are needed in the first stage of a $1\times8$ OBFN, since more than four times the delay of one ring can be reached with three rings at the same error level. This conclusion can save one ring and its hardware needed to operate it for $1\times8$ OBFN, but many more rings for larger structures.

### 7.2 Recommendations

The first recommendation directly follows from the last conclusion about the number of rings required for larger structures. The conclusion is valid in theory, but research is needed to see whether it also works in practice. Also when larger OBFN chips are designed it should be researched how many rings are needed at the stages where more than four rings were originally planned. Furthermore it should be researched what the effect on the system is when a two-dimensional antenna array is used instead of one-dimensional.

As mentioned in Section 6.3 there are fabrication offsets and unpredictable parasitic effects in the chip which require an adaptive algorithm to compensate the $\kappa$s and $\phi$s for it. Further research should be carried out in order to design such an algorithm. A
calibration scheme could also be a solution to fabrication offset problem and should therefore also be researched. When a different response formula is found during that research, the principles of Section 6.2 should be applied to adapt the tunings to it. It can also be combined with the implementation and testing of the algorithms derived in this research.

The splitters in Figure 1.3 can be used for beam shaping purposes and to correct for differences in power loss between the branches caused by the ORRs. Therefore research should be carried out about how to tune these splitters.
References


Appendix A

Description of Non Linear Programming method used

The general idea behind Linear Programming (LP) and NLP methods is to transform the problem into an easier subproblem and use that to solve the problem in an iterative process. In this case the subproblem is a Quadratic Programming (QP) problem and the NLP method itself is called Sequential Quadratic Programming (SQP). More details of SQP can be found in [14] but a basic description will be given in here. Using the problem description of (4.1) the QP is based on the following approximation of the Lagrangian function

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x)$$  \hspace{1cm} (A.1)

with $\lambda_i$ the Lagrange multipliers, which could take any value except zero. The approximation just assumes that the bound constraints are inequality constraints. This can be seen from the formula since $\lambda$ could be both positive and negative. So therefore the constraint function can be approached from both sides. The constraints used in the problems of this research are all bound constraints so this assumption could theoretically result into an infeasible solution (e.g. some constraints are not matched). However this would immediately be noted by the MATLAB function and corrected, so when a feasible solution is reached no accuracy loss is caused by this step. Lagrange theory is normally used for finding the extrema of a function under a certain number of constraints. The $\lambda$ operator is in fact used to convert the constraints of $f(x)$ into variables for the function $L(x,\lambda)$. The next step is generate the QP problem, which is the basis of the SQP method since this problem will be solved every iteration to get
closer and closer to the solution. It is derived from (A.1) by linearizing the constraints

\[
\min \left\{ \frac{1}{2} d_k^T H_k d_k + \nabla f(x_k)^T d_k \right\}
\]

\[
s.t. \quad \nabla g_i(x_k)^T d_k + g_i(x_k) = 0
\]
\[
\nabla g_i(x_k)^T d_k + g_i(x_k) \leq 0
\]

(A.2)

in which \( d \) is the vector variable to solve for and \( H_k \) is the Hessian matrix of \( L(x, \lambda) \) from A.1. The Hessian is a matrix containing all second order partial derivatives of a certain function and is often used in optimization theory. The linearization of the constraints can be seen directly. The gradient of the constraint functions is multiplied with \( d \) and the current value of the constraint functions is added to it. The objective function is a second order approximation of the original one. It is constructed of two parts: \( d_k^T H_k d_k \) is a quadratic part since all columns of \( H_k \) are multiplied by the \( d_k^T \) and all rows by \( d_k \). The second part is linear since it is the gradient of the original objective function times \( d_k \). This problem can be solved with any QP solver. QP problems are much easier than NLP problems. More details can be found in [15].

The solution of \( d \) provides an improving direction for the metric. So after solving the QP problem the vector \( x \) is updated according to the following rule

\[
x_{k+1} = x_k + a_k d_k
\]

(A.3)

in which \( a_k \) is the step size for the SQP algorithm. It is calculated by an appropriate line search procedure. It should not be too small because then it will take too many iterations to reach the solution and not too large either since then it could be possible to jump over the solution. The next step is updating the Hessian matrix \( H_k \). These two operations are quite involved and will therefore not be explained in here, more can be found in [14]. After those steps a new iteration is started with new values for \( x_k \) and \( H_k \) to find a new solution for \( d_k \) in (A.2). Since \( d_k \) provides an improving direction every iteration will be closer to the final solution. This is done until the stepsize or the metric becomes smaller than a user specified value.
Appendix B

MATLAB scripts and functions

B.1 Visualization functions

The MATLAB functions of this section are used to generate plots of the different types of responses of the ORRs. The first function is used to generate a delay response for a certain values of $\kappa$ and $\phi$. Cascading multiple ORRs can be done easily by just adding the output vectors.

function [output,f] = ORR_delayresponse(k,phi,points);

% ORR_response gives the response of an Optical ring resonator with a
% unit delay of 0.08 ns between 107.522 and 107.527 THz
% usage: [output,f] = ORR_response(kappa,phi,points)
% in which kappa and phi are the parameters of the ring and points is
% the number of calculated points in the frequency range. The resulting
% delay response is given in the vector output and the corresponding
% frequency set is given in f. A graph can be made with plot(output,f).

start = 107.5224817e12;
stop = 107.5274817e12;
T = 0.08e-9;
PI = 4*atan(1);
k = k*ones(1,points);
phi = phi*ones(1,points);
f = start:(stop-start)/(points-1):stop;
output = (k*T)./(2-k-2.*sqrt(1-k).*cos(2*PI*f*T+phi));
The next function takes the loss of one roundtrip time of the ring into account too:

```matlab
function [output,f] = loss_response(k,phi,points,alpha);

% ORR_response gives the response of and Optical ring resonator with a
% unit delay of 0.08 ns between 107.522 and 107.527 THz
%
% usage: [output,f] = loss_response(kappa,phi,points,alpha)
% in which kappa and phi are the parameters of the ring and points is
% the number of calculated points in the frequency range. Alpha is the
% loss of the ring for one roundtrip time is The resulting delay
% response is given in the vector output and the corresponding
% frequency set is given in f. A graph can be made with plot(output,f).

start = 107.5224817e12;
stop = 107.5274817e12;
r = 10^(-alpha/20);
T = 0.08e-9;
PI = 4*atan(1);
k = k*ones(1,points);
phi = phi*ones(1,points);
f = start:(stop-start)/(points-1):stop;
output = T/2.*(1-r^2*(1-k))./( 1+r^2.*(1-k)-2.*r.*sqrt(1-k)...+
   .*cos(8.*atan(1).*f.*.08e-9+phi) )...+
   + T/2.*(r^2-(1-k))./( 1-k+r^2-2.*r.*sqrt(1-k)...+
   .*cos(8.*atan(1).*f.*.08e-9+phi) );
```
The last visualization function is used to generate phase responses:

```matlab
function [output, f] = phase_response(k, phi, points);

% ORR_response gives the response of and Optical ring resonator with a
% unit delay of 0.08 ns between a normalized frequency of 0 and 2 GHz.
% usage: [output,f] = phase_response(kappa,phi,points)
% in which kappa and phi are the parameters of the ring and points is
% the number of calculated points in the frequency range. The
% resulting delay response is given in the vector output and the
% corresponding frequency set is given in f. A graph can be made with
% plot(output,f).

start = 0;
stop = 2e9;
T = 0.08e-9;
PI = 4*atan(1);
k = k*ones(1,points);
phi = phi*ones(1,points);
f = start:(stop-start)/(points-1):stop;
output = atan( sin(2*PI*f*T+phi) ./ ( sqrt(1-k)...
          - cos(2*PI*f*T+phi) ) )...
          - atan( sqrt(1-k).*sin(2*PI*f*T+phi) ./ ( 1 - sqrt(1-k)*...
          .*cos(2*PI*f*T+phi) ) ) );
for l=2:1:points
    if (output(l)>output(l-1)+PI/2)
        output(l:1:points)=output(l:1:points)-PI;
    end
end
f=f*T;
```
B.2 Optimization algorithms

In this section the functions used for the optimization algorithms are presented. Basically they consist of two functions, one for the optimization and one for the metric function. These two are placed in different files to keep things clarifying. The first one presented is the one for delay optimizing the one ring output.

function [tunings,FVAL] = ORR_1opti(points,bandwidth,height);

% ORR_1opti optimizes one ring for a target bandwidth and delay
%
% Usage: [tunings,FVAL] = ORR_1opti(points,bandwidth,heigth)
% In which points is the number of points to be used in the
% optimization. Bandwidth is the target bandwidth in Hz for the
% optimization around the center frequency of 107.525 THz.
% Heigth is the target delay in ns. The vector tunings contains
% the optimal kappa and phi and FVAL is the metric value of
% the optimal solution.

options = optimset(@fminbnd); %options needed for accuracy
options = optimset(options,'TolFun',1e-16,'TolX',1e-16);
global DREF
DREF = height;

global FREQARRAY
global LENGTH
LENGTH = points;
start = 107.5249817e12 - bandwidth/2;
stop = 107.5249817e12 + bandwidth/2;
FREQARRAY(1:LENGTH)=start:(stop-start)/(LENGTH-1):stop;

[tunings,FVAL] = fmincon(@delayfun1,[.995,0],[],[],[],[],...[0,-1],[.995,1],[],options) %see fmincon for more details
The corresponding metric function \textit{delayfun1}:

\begin{verbatim}
function mu1 = delayfun1(x)

    % Function used for optimization in ORR_1opti.
    global DREF
    global FREQARRAY
    global LENGTH
    mu1 = 0;
    for k=1:1:LENGTH
        mu1 = mu1 + ((x(1)*.08)/(2-x(1)-2*sqrt(1-x(1))*cos(8*atan(1)*FREQARRAY(k)*.08e-9+x(2)))-DREF)^2;
    end
\end{verbatim}
The next function is used to optimize for the two and three ring output at once, the separate functions for the two and three ring outputs on their own are not presented in this appendix since they are based on the same principles as the ones presented in here.

```matlab
function [tunings,FVAL] = ORR_totopti(points,bandwidth,height,height2);

% ORR_totopti optimizes one ring for a target bandwidth and delay
% Usage:[tunings,FVAL] = ORR_totopti(points,bandwidth,heigth,heigth2)
% In which points is the number of points to be used in the
% optimization. Bandwidth is the target bandwidth in Hz for the
% optimization around the center frequency of 107.525 THz.
% Heigth is the target delay in ns for the three ring output and
% heigth2 is for the two ring output. The vector tunings contains
% the optimal kappas and phis and FVAL is the metric value of
% the optimal solution.

options = optimset(@fminbnd); %options needed for accuracy
options = optimset(options,'TolFun',1e-16,'TolX',1e-16);
global DREF
DREF = height;
global DREF2
DREF2 = height2;

global FREQARRAY
global LENGTH
LENGTH = points;
start = 107.5249817e12 - bandwidth/2;
stop = 107.5249817e12 + bandwidth/2;
FREQARRAY(1:LENGTH)=start:(stop-start)/(LENGTH-1):stop;

[tunings,FVAL] = fmincon(@delayfuntot,[.6,.8,.6,-.8,.8,0],[],[],[],[],[0,-2,0,-2,0,-2],[.995,2,.995,2,.995,2],[],options)
% see fmincon for more details
```
The corresponding metric function of $ORR_{totopi}$ is given by:

\begin{verbatim}
function mutot = delayfuntot(x)

    global DREF
    global DREF2
    global FREQARRAY
    global LENGTH
    mutot = 0;
    for k=1:1:LENGTH
        mutot = mutot + ( (x(1)*.08)/(2-x(1)-2*sqrt(1-x(1))*cos(8*atan(1)*FREQARRAY(k)*.08e-9+x(2))) + (x(3)*.08)/(2-x(3)-2*sqrt(1-x(3))*cos(8*atan(1)*FREQARRAY(k)*.08e-9+x(4)))
                             + (x(5)*.08)/(2-x(5)-2*sqrt(1-x(5))*cos(8*atan(1)*FREQARRAY(k)*.08e-9+x(6)))-DREF)^2;
    end

    mutot = mutot + ( (x(1)*.08)/(2-x(1)-2*sqrt(1-x(1))*cos(8*atan(1)*FREQARRAY(k)*.08e-9+x(2))) + (x(3)*.08)/(2-x(3)-2*sqrt(1-x(3))*cos(8*atan(1)*FREQARRAY(k)*.08e-9+x(4)))) - DREF2)^2;
end
\end{verbatim}
For phase tuning the following functions are used: \textit{phase\_1opti}, \textit{phase\_2opti}, \textit{phase\_3opti} and \textit{phase\_totopti}. Since their structure is quite similar only \textit{phase\_totopti} will be presented in this appendix:

\begin{verbatim}
function [tunings,FVAL]=phase_totopti(points,bandwidth,...
  height,height2,varargin);

% phase_totopti optimizes one ring for a target bandwidth and delay %
% Usage: [tunings,FVAL]=phase_totopti(points,bandwidth,heigth,heigth2)
% In which points is the number of points to be used in the
% optimization. Bandwidth is the target bandwidth in Hz for the
% optimization around the normalized frequency of 0 Hz.
% Heigth is the target delay in ns for the three ring output and
% heigth2 is for the two ring output. The vector tunings contains
% the optimal kappas and phis and FVAL is the metric value of
% the optimal solution. An initial estimate is optional and can be
% added after heigth2 in the form [2kappa,2phi,3kappa].

if (length(varargin) == 0)
  initiale = [0.65,0.4,0.8];
else
  initiale = [varargin{:}];
end
options = optimset(@fminbnd); \%needed for accuracy
options = optimset(options,'TolFun',1e-16,'TolX',1e-16);
global DREF
DREF = height;
global DREF2
DREF2 = height2;
global FREQARRAY
global LENGTH
LENGTH = points;
start = 0;
stop = bandwidth/2;
FREQARRAY(1:LENGTH)=start:(stop-start)/(LENGTH-1):stop;

[tunings,FVAL] = fmincon(@phasefuntot,initiale,[],[],...
  [],[],[0.1,-2,0.1],[.995,2,.995],[],options)
% see fmincon for details
\end{verbatim}
The corresponding metric function of \textit{phase\_totopti} is given by:

\begin{verbatim}
function mutot = phasefuntot(x)

% Function used for optimization in phase\_totopti.

global DREF
global DREF2
global FREQARRAY
global LENGTH

PI = 4*atan(1);
T = .08e-9;
mutot = 0;

for k=1:1:LENGTH
    diff1 = ( atan( sin(2*PI*FREQARRAY(k)+x(2)) ./ ( sqrt(1-x(1)) -
               cos(2*PI*FREQARRAY(k)+x(2)) ) - atan( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)+x(2)) ./ ( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) + x(2)) ) )+
             atan( sin(2*PI*FREQARRAY(k)-x(2)) ./ ( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) + x(2)) ) - atan( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) ./ ( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) + x(2)) ) ) +
             atan( sin(2*PI*FREQARRAY(k)) ./ ( sqrt(1-x(3)).*sin(2*PI*FREQARRAY(k)) ) - atan( sqrt(1-x(3)).*sin(2*PI*FREQARRAY(k)) ./ ( sqrt(1-x(3)).*sin(2*PI*FREQARRAY(k)) + x(2)) ) ) +
             DREF*FREQARRAY(k)*78.54);

    diff2 = ( atan( sin(2*PI*FREQARRAY(k)+x(2)) ./ ( sqrt(1-x(1)) -
               cos(2*PI*FREQARRAY(k)+x(2)) ) - atan( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)+x(2)) ./ ( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) + x(2)) ) )+
             atan( sin(2*PI*FREQARRAY(k)-x(2)) ./ ( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) + x(2)) ) - atan( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) ./ ( sqrt(1-x(1)).*sin(2*PI*FREQARRAY(k)-x(2)) + x(2)) ) ) +
             DREF2*FREQARRAY(k)*78.54);

    if (diff1 > 3) %remove phase jumps
        diff1 = diff1 - floor((diff1+0.5)/PI)*PI;
    end

    mutot= mutot + diff1^2;
    if (diff2 > 3) %remove phase jumps
        diff2 = diff2 - floor((diff2+0.5)/PI)*PI;
    end
\end{verbatim}
mutot = mutot + diff2^2;
end
The last optimization algorithm treated in this appendix is for power tuning. Since this can only be done for the whole chip at once, there is only one function of this type.

```matlab
function [tunings,FVAL] =
    power_opti(points,bandwidth,heigth,heigth2,heigth3);

% power_opti optimizes one ring for a target bandwidth and delay
% Usage:[tunings,FVAL] =
%   power_opti(points,bandwidth,heigth,heigth2,heigth3)
% In which points is the number of points to be used in the
% optimization. Bandwidth is the target bandwidth in Hz for the
% optimization around the normalized frequency of 0 Hz.
% Heigth is the target delay in ns for the three ring output,
% heigth2 is for the two ring output and heigth3 for the one ring
% output. The vector tunings contains the optimal kappas and phis
% and FVAL is the metric value of the optimal solution.

options = optimset(@fminbnd); %needed for accuracy
options = optimset(options,'TolFun',1e-16,'TolX',1e-16);
% add the following when you think more iterations are needed:
%,'MaxFunEvals',50000,'MaxIter',50000);
global DREF
DREF = heigth;
global DREF2
DREF2 = heigth2;
global DREF3
DREF3 = heigth3;

global FREQARRAY
global LENGTH
LENGTH = points;
start = 0;
stop = bandwidth/2;
FREQARRAY(1:LENGTH)=start:(stop-start)/(LENGTH-1):stop;

[tunings,FVAL] = fmincon(@powerfun,[0.69,0.37,0.95,0.78],[],[],[],[],[0.1,0,0.1,0.1],[.995,2,.995,.995],[],options)
```
The corresponding metric function for power tuning is complicated but the structure is quite similar to the function of phase tuning.

function mu = powerfun(x)

% Function used for optimization in power_opti.

global DREF
global DREF2
global DREF3
global FREQARRAY

global LENGTH

PI = 4*atan(1);
T = .08e-9;
mu = 0;
for k=1:1:LENGTH
  powersum = 1;
  diff1 = ( atan( sin(2*PI*FREQARRAY(k)+x(2)) ./ ( sqrt(1-x(1)) - ...
                     cos(2*PI*FREQARRAY(k)+x(2)) ) ) - atan( sqrt(1-x(1)).*sin(2*PI*...
                     FREQARRAY(k)+x(2)) ./ ( 1 - sqrt(1-x(1)).*cos(2*PI*FREQARRAY(k)...
                     + x(2)) ) )+ atan( sin(2*PI*FREQARRAY(k)-x(2)) ./ ( sqrt(1-x(1))...
                     - cos(2*PI*FREQARRAY(k)-x(2)) ) )-atan( sqrt(1-x(1)).*sin(2*PI*...
                     FREQARRAY(k)-x(2)) ./ ( 1 - sqrt(1-x(1)).*cos(2*PI*FREQARRAY(k)...
                     -x(2))))) + atan( sin(2*PI*FREQARRAY(k))./( sqrt(1-x(4))-cos(2*PI*...
                     FREQARRAY(k)) ) )- atan( sqrt(1-x(4)).*sin(2*PI*FREQARRAY(k))./(...
                     1-sqrt(1-x(4)).*cos(2*PI*FREQARRAY(k))))+ DREF*FREQARRAY(k)*78.54);

  diff2 = ( atan( sin(2*PI*FREQARRAY(k)+x(2)) ./ ( sqrt(1-x(1)) - ...
                     cos(2*PI*FREQARRAY(k)+x(2)) ) )- atan( sqrt(1-x(1)).*sin(2*PI*...
                     FREQARRAY(k)+x(2)) ./ ( 1 - sqrt(1-x(1)).*cos(2*PI*FREQARRAY(k)...
                     + x(2)) ) )+ atan( sin(2*PI*FREQARRAY(k)-x(2)) ./ ( sqrt(1-x(1))...
                     - cos(2*PI*FREQARRAY(k)-x(2)) ) )- atan( sqrt(1-x(1)).*sin(2*PI*...
                     FREQARRAY(k)-x(2)) ./ ( 1 - sqrt(1-x(1)).*cos(2*PI*FREQARRAY(k)...
                     -x(2)) )) + DREF2*FREQARRAY(k)*78.54);

  diff3 = ( atan( sin(2*PI*FREQARRAY(k))./( sqrt(1-x(4))-cos(2*PI*...
                     FREQARRAY(k)) ) )- atan( sqrt(1-x(4)).*sin(2*PI*FREQARRAY(k))./(...
                     1-sqrt(1-x(4)).*cos(2*PI*FREQARRAY(k))))+ DREF3*FREQARRAY(k)*78.54);

if (diff1 > 3) %remove phase jumps
diff1 = diff1 - floor((diff1+0.5)/PI)*PI;
end
if (diff2 > 3)
    diff2 = diff2 - floor((diff2+0.5)/PI)*PI;
end
if (diff3 > 3)
    diff3 = diff3 - floor((diff3+0.5)/PI)*PI;
end
powersum = powersum + exp(i*diff1)+ exp(i*diff2)+ exp(i*diff3);
mu = mu + (4-abs(powersum))^2;
end
B.3 Scripts used for the rules of thumb

In this section the scripts and functions used for the derivation of the rules of thumb will be treated. The first script is called `dplot2` and acquires the optimal values for the $\kappa$s and $\phi$s as a function of target bandwidth and delay for the rings in the two and three ring outputs of the system. The reason why this has been put into a script is that the tuning results are used as initial estimate for the next optimization point. This is to prevent the algorithm from hitting local minima.

```matlab
%note that the first point needs to be calculated manually
for k=2:1:241
    for i=1:1:3
        [Z(k,i,1:1:3),FVAL]=phase_totopti(1000,-.05+i/10,...
            (.159+k/1000)*1.5,.159+k/1000,Z(k-1,i,1:1:3));
    end
end
X = .05:.1:.25;
Y2 = (.16:.001:.4)*1.25;
Y3 = (.24:.0015:.6)*1.25;
```
There was no MATLAB function present for fitting functions of the form (5.5), and therefore a new function was written in order to use in combination with \textit{fmincon} to fit the coefficients of (5.5). The function is presented below:

\begin{verbatim}
function fu1 = fitfun1(x)

global Z
global Y2
fu1 = 0;
fu1 = fu1 + sum(abs(transpose( x(1)./(Y2-x(2)).^2 + ...
    x(3)./(Y2-x(4)) + x(5) )-abs(Z(:,3,2)) ));
\end{verbatim}
% Script used to generate error curve

The last script is used to generate a plot of the metric value as a function of the target delay, for one, two, three and four rings.

% script used to generate a plot of the metric value
% for different numbers of rings. Note that the last
% one, phase_4opti is quite sensitive for local minima
% so postprocessing might be required on that data.

for i=80:1:250
    [leeg,err1(i)] = phase_1opti(1000,.15,i/1000);
end
for i=160:1:600
    [leeg,err2(i)] = phase_2opti(1000,.15,i/1000);
end
for i=240:1:1040
    [leeg,err3(i)] = phase_3opti(1000,.15,i/1000);
end
for i=1240:1:1500
    [leeg,err4(i)] = phase_4opti(100,.15,i/1000);
end

hold on
plot([.90:.01:2.50]*1.25,err1(90:1:250))
plot([1.60:.01:6.00]*1.25,err2(160:1:600),('--'))
plot([2.40:.01:10.40]*1.25,err3(240:1:1040))
plot([3.20:.01:15.00]*1.25,err4(320:1:1500),('--'))