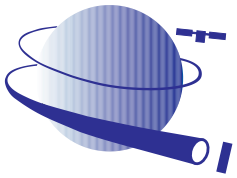

University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science
Telecommunication Engineering Group

Context, Design and Implementation of a Control System for Ring Resonator-based Optical Beam Forming Networks

by

Jan-Willem van 't Klooster

Master thesis
Executed from May 7, 2007 to October 10, 2008.

Supervisor: Dr. Ir. C.G.H. Roeloffzen
Dr. Ir. A. Meijerink
Dr. Ir. B.J.F. van Beijnum

Summary

This Thesis reports on a Master assignment conducted at the Telecommunication Engineering (TE) Group at the University of Twente. The task is to develop and implement a control system for ring resonator-based Optical Beam Forming Networks (OBFNs).

Beam shaping and beam steering, together called beamforming, is needed when processing radio-frequency signals from phased array antennas. This can be achieved in the optical domain, by tuning an OBFN. The tuning of such a network is the task of a control system. That system is the topic of research in this assignment. It enables thermo-optical tuning of ring resonator based OBFNs.

In this thesis, the context, design and implementation of the control system are studied. Measurements are described to show the correct working of the implemented prototype.

The control system may be used in the future in an airborne application. It may form part of a system that provides services such as radio, television, and internet access to en route aircraft. It is a challenge to make this possible.

The control system consists of a controller and an interface to operate it. The controller has hardware and software aspects. The hardware is a modular set of components that can easily be extended when necessary. The software in the controller operates the hardware and provides the means for tuning OBFNs. A graphical interface is provided on a PC to operate the controller. This can easily be adapted to future needs.

Two prototypes for the control system have been delivered in this assignment. The first prototype met the most important requirements but was not very easy to operate. The second implemented also met the more advanced requirements, enabled tuning of larger OBFNs, and was provided with an easy-to-use Java interface.

The system has been used extensively in the TE lab and also at the Dutch Aerospace Laboratory (NLR). In the future, the system can be extended to provide angle and

azimuth steering (to steer the beam to a certain point) and tracking and tracing functionalities (to find and keep focussed on a source).

Samenvatting

Dit afstudeerverslag doet verslag van een afstudeeropdracht bij de Telecommunication Engineering (TE) Group van de Universiteit Twente. Het doel van deze opdracht is om een aansturingssysteem te ontwerpen en te realiseren.

Beam shaping en *beam steering*, samen ook wel bundelvorming genoemd, is noodzakelijk vor het verwerken van signalen van meerdere antennes, zoals antenne-arrays. Deze functionaliteit kan worden gerealiseerd in het optische domein, door middel van het tunen van een optisch bundelvorm-netwerk (OBFN). Een aanstuursysteem zorgt voor het correct afregelen van zo'n netwerk. Dit systeem is het onderwerp van onderzoek in deze opdracht.

In dit verslag worden de context, het ontwerp en de implementatie van dit controlesysteem bestudeerd. Om de correcte werking van het geimplementeerde prototype aan te tonen, worden tevens metingen beschreven.

In de toekomst kan het aanstuursysteem worden gebruikt in de luchtvaart. Het zou onderdeel uit kunnen maken van een systeem dat zorgt voor diensten zoals televisie en internet op vliegtuigen die onderweg zijn.

Het ontwerp van het aanstuursysteem bestaat uit een aansturend gedeelte en een interface om het aansturende gedeelte te bedienen. Het aansturende gedeelte heeft zowel hardware- als software aspecten. De hardware is een modulaire set componenten, die gemakkelijk kan worden uitgebreid, indien noodzakelijk. De software in de controller bedient de hardware en zorgt ervoor dat OBFN's kunnen worden getuned. Op een PC is een interface aanwezig waarmee de controller kan worden bediend. Deze kan eenvoudig worden aangepast aan toekomstige eisen.

Tijdens deze opdracht zijn twee versies van het aanstuursysteem opgeleverd. Het eerste prototype voldeed aan de belangrijkste eisen, maar was niet erg gebruiksvriendelijk. De tweede implementatie voldeed ook aan de meer geavanceerde eisen, en maakte het mogelijk om grotere OBFN's af te kunnen regelen. De bediening van deze versie is

een stuk gemakkelijker omdat voorzien was in een gebruiksvriendelijke, grafische Java-interface.

Het systeem is veel gebruikt in het laboratorium van TE en tevens in het Nederlands Lucht- en Ruimtevaartlaboratorium (NLR). In de toekomst kan het systeem worden uitgebreid met functionaliteiten zoals hoek- en azimutsturing en het traceren en volgen van een bepaalde zender.

Foreword

Welcome. In front of you is my Master thesis, written to accomplish my Master of Science study in Telematics. Writing a thesis has parallels with the story of Theseus, the ancient Greek hero, who faced and overcame a multifaceted problem and a labyrinth, with some aid of good people.

The work on which this thesis reports is conducted within the Telecommunication Engineering Group at the University of Twente. The assignment was to make a controller for optical beamforming networks. You can read all about this in this thesis.

Now I am almost finished writing, I want to thank some people. First of all my daily advisor dr. ir. Chris Roeloffzen, whose aid and ideas have really been useful. Chris has the wonderful ability to really motivate tens or maybe even hundreds of people and stil give them all personal attention. Moreover, he is one of the best down-to-earth teachers explaining difficult topics in very clear language.

I want to thank prof. dr. ir. Wim van Etten for giving me the opportunity to graduate at Telecommunication Engineering, and I also want to thank my advisors dr. ir. Arjan Meijerink and dr. ir. Bert-Jan van Beijnum, for their useful ideas and the good discussions. I would like to thank Jaco, Pieter, Adriaan and Harm from NLR for their support.

Without the support of my family and girlfriend this result would never have been possible, therefore a very big thank you to Anita and Kees and to Tabea, who are always there for me. I also want to thank my father for his professional support and ideas, and for motivating me to publish.

Leimeng, Eduard and Theo I want to thank for their assistance, mainly in the TE lab and I want to thank my fellow (ex-)students Audrey, Roland, Dick, Liang, Sjoerd, Nicolas, Martin, Roelof, Jack and Thomas for the good time at the group, as well as all employees, PhD students and the secretary Lilian for their support. TE is a really inspiring working environment to go to!

Another very good place to work is Menzing in Haaksbergen, I want to thank Erik to being so flexible regarding my tasks there.

Last but not least I would like to thank my good friends Patrick, Wouter, Olivier, Ivo, Evert, Vera, Gert, Martijn, Wilco, Emad and Rob for the good times.

Of course there are people I have forgotten. So do not worry if you are not listed, I also want to thank you. You know why.

Jan-Willem van 't Klooster, October 4^{th}, 2008.

Abbreviations

AE	Antenna Element
CBB	Connexion By Boeing
CDMA	Code Division Multiple Access
COTS	commercial of the shelf
CS	Chip Select
DAC	Digital to Analog Converter
DFD	Data Flow Diagram
DSSS	Direct Sequence Spread Spectrum
DVB-RCS	Digital Video Broadcasting - Return Channel via Satellite
DVB-S	Digital Video Broadcasting - Satellite
DVB-S2	Digital Video Broadcasting - Satellite 2
EDFA	Erbium Doped Fiber Amplifier
EMI	Electromagnetic Interference
EPG	Electronic Programming Guide
FDC	Frequency down conversion
FDMA	Frequency Division Multiple Access
FHSS	Frequency Hopping Spread Spectrum
FRAM	Ferroelectric Random Access Memory
FSR	Free Spectral Range
FTDI	Future Technology Devices International

GES	Ground Earth Station
GPRS	General Packet Radio Service
GUI	Graphical User Interface
IBM	International Business Machines
IDE	Integrated Drive Electronics
IETF	Internet Engineering Task Force
ISDN	Integrated Services Digital Network
IP	Internet Protocol
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LEE	Lammerink Electrical Engineering
LLC	Logical Link Control
LNA	Low Noise Amplifier
MAC	Medium Access Control
MiSo	Master In Slave Out
MES	Mobile Earth Station
MoSi	Master Out Slave In
MVC	Model View Control
MZI	Mach-Zehnder Interferometer
NIVR	Nederlands Instituut voor Vliegtuigontwikkeling en Ruimtevaart
NLR	National Aerospace Laboratory
OBFN	Optical Beam Forming Network
opamp	operational amplifier
ORR	Optical Ring Resonator

OSBF	Optical Sideband Filter
PAA	Phased Array Antenna
PCB	Printed Circuit Board
RISC	Reduced Instruction Set Computer
SACK	Selective Acknowledgement
SD	Secure Digital
SPI	Serial Perhiperal Interface
SSB-SC	Single Sideband Suppressed Carrier
TCP	Transport Control Protocol
TDMA	Time Division Multiple Access
TE	Telecommunication Engineering
TEC	Temperature Controller
UART	Universal Asynchronous Receiver and Transmitter
UDP	User Datagram Protocol
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunications System

Contents

Chapter 1

Introduction

A Phased Array Antenna (PAA) is an interesting alternative to a conventional dish antenna because of its appealing properties such as flatness, electronical steering and multi-beam capabilities.

Beam shaping and beam steering, together called *beamforming*, is needed when processing radiofrequency signals from PAAs. This can be achieved by tuning an Optical Beam Forming Network (OBFN). The tuning of such a network is the task of a control system. That system is the subject of this Master assignment.

The purpose of this Thesis is to present the context, design and implementation of a control system for ring resonator-based OBFNs. That control system was developed at the chair of Telecommunication Engineering (TE), within in the context of the SMART project: **Sm**art **A**ntennas for **R**adio **T**ransceivers [1].

The chapter continues as follows. The motivation for this assignment is given in Section 1.1. In Section 1.2, the background of this research is presented. The system description is given and related work is discussed. Section 1.3 gives the research organization. The methodology that is used is presented, and the research goals and question are formulated. This chapter ends with the structure of the Thesis in Section 1.4.

1.1 Motivation

There are two main reasons[1] for conducting this Master assignment.

1. The development of the controller is a challenging engineering activity that is not finished yet. Important parts of the controller still have to be designed, and the complete control system has to be integrated and implemented as a part of this assignment, in order to tune OBFNs.
2. The TE group is interested in the higher layer aspects of the beamforming system. For example, what applications could be made available with such a

[1] Another main reason (outside this particular scope) is obtaining an MSc degree.

system, and are there any consequences for the infrastructures and protocols on which these applications rely? The project results require higher level developments to reach production readiness eventually.

1.2 Background

The task is to make a controller for the beamformer used in the SMART project. Beamforming is necessary when combining signals from multiple antennas, such as PAAs.

When receiving, the beamformer is tuned, such that it combines the signals of the antenna elements with the right delay. In that way, constructive interference and a stronger signal is achieved.

When transmitting, the signal is split and delayed in the beamformer to a certain extent, before radiated by the antenna elements in the desired direction. This work focusses mostly on receive scenarios however.

Beamforming can be achieved in various ways, including digital beamforming, microwave beamforming, local beamforming, aerial beamforming and optical beamforming [2].

We focus on optical beamforming. This technology provides compactness, large bandwidth, frequency independence, Electromagnetic Interference (EMI) immunity, low loss, potentially low costs, and no beam squinting, which is often a problem with electronic beamforming. Hence it is possible to communicate in a tunable direction, in a cost effective manner. It is possible to enable a large instantaneous bandwidth. So the amount of information that can potentially be processed instantaneously, is large when optical beamforming is employed. For applications such as broadband internet access or mobile satellite television reception, this is a very useful feature.

1.2.1 System overview

The proposed SMART system is a system employing optical beamforming. It is shown in Figure 1.1. For a receive scenario, it consist of the shown parts. The Antenna Elements (AEs) receive satellite signals. These RF signals are intensity modulated onto an optical carrier (E/O block) and fed into the OBFN, where they are delayed and combined. This results in one strong optical signal, which is then converted back to the electrical domain using a photodiode detector (O/E block). Afterwards, it can be detected using for example a set top box.

The tuning of the OBFN is managed by a control system. Its input is information belonging to the steering angle of the antenna array. Using this information, the OBFN is tuned such that there is *constructive interference* for RF signals coming from the

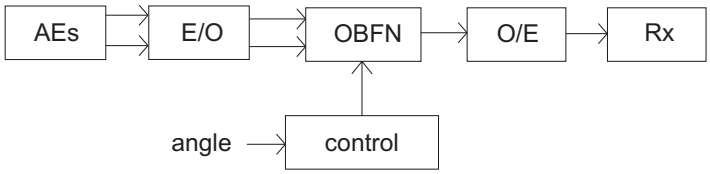

Figure 1.1: System overview of the optical beamforming system for phased array receive antennas.

desired direction.

At the heart of this system is the OBFN, produced on an optical chip. The next part of this introduction discusses the OBFN.

Optical beamforming network

The optical chip in this system is manufactured using planar optical waveguide technology by Lionix B.V. [3]. It consists of the following building blocks: waveguides, Mach-Zehnder Interferometers (MZIs), couplers and Optical Ring Resonators (ORRs). ORRs are chosen because they provide true time delay when cascaded, so beam squinting will not occur. The building blocks are combined to form an OBFN. A 1×8 OBFN for a transmitter phased array is shown in Figure 1.2.

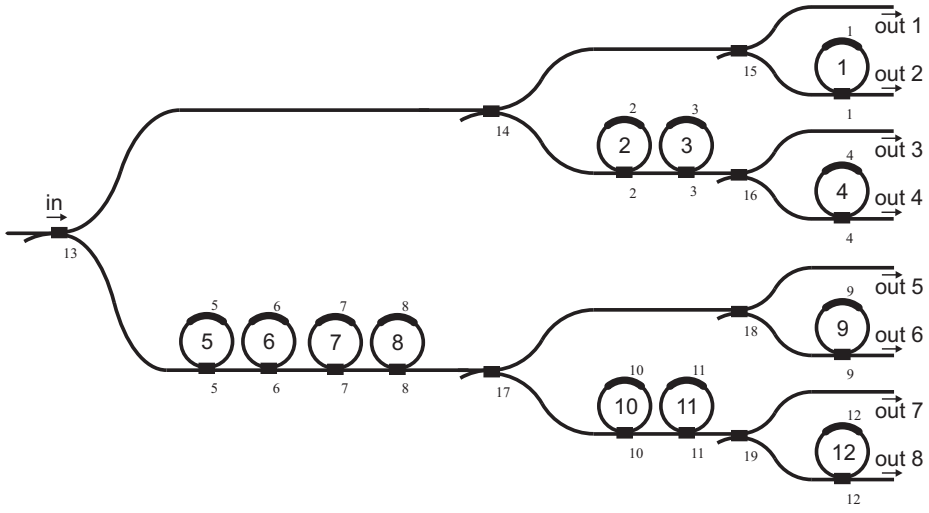

Figure 1.2: A 1×8 binary tree OBFN for a transmitter phased array antenna with 1 input, 8 outputs and 12 optical ring resonators.

It employs a binary tree topology, which has an efficient layout with respect to the required number of ORRs. This puts restrictions on tuning freedom compared to a parallel topology, but reduces tuning complexity. An ORR consists of a straight waveguide and a circular waveguide coupled to it. It has a periodic group delay response, representing the effective time delay to the modulated RF signal. The group delay is

expressed by [4]

$$\tau_g(f) = \frac{(\kappa T)}{2 - \kappa - 2\sqrt{1-\kappa}cos(2\pi f T + \phi)} \tag{1.1}$$

It depends on the round trip time T, the power coupling coefficient κ and additional round-trip phase shift of the ring ϕ. It is possible to control both the phase shift ϕ and power coupling coefficient κ, thereby tuning the ORR peak value delay and resonance frequency. There is a trade-off between peak delay and bandwidth. Therefore it is required to cascade ORRs, creating broadband delay elements. This yields a group delay response that is simply the sum of the individual ORR responses. Figure 1.3 clarifies this: the group delay response of three cascaded rings (shown in the inset) is the sum of the three individual responses, marked with the three dashed lines. As the resonance frequencies of the ORRs gets closer, the ripple becomes smaller. The group delay then becomes more flat, but at the cost of a smaller bandwidth.

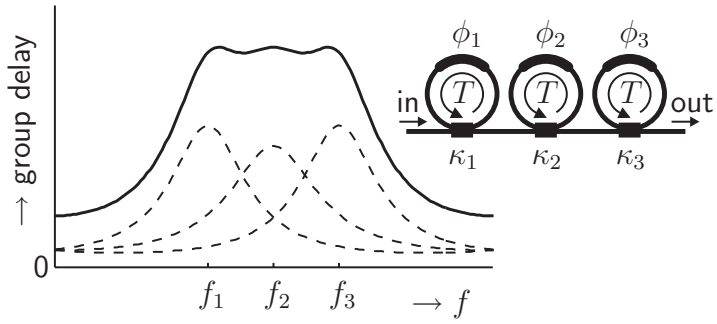

Figure 1.3: Theoretical group delay response of three cascaded ORRs. The three dashed lines in the graph show the group delay responses for the three individually tuned ORRs. The solid line is the total group delay. The inset shows the configuration and tunable elements of three cascaded ORRs. (picture taken from [5]).

In our system, the OBFN is used to realign the individual AE signals, in order to combine them with maximal constructive interference. The output of a single laser is split, after which each AE signal is modulated using filter-based Single Sideband Suppressed Carrier (SSB-SC) modulation, as discussed below.

The optical chip is tuned thermo-optically by electrically heating chromium resistors on it. As a consequence, the optical waveguide heats up and its refractive index changes. This allows for tuning of the resonance frequencies and power coupling ratios. The tuning is performed by the control system, as discussed later in this Section. Thermo-optical tuning itself is very well explained in Section 3.4 of [6].

Modulation and demodulation

A more architectural view of the system is presented in Figure 1.4. After amplification in the Low Noise Amplifier (LNA) and Frequency down conversion (FDC), E/O conversion is done using a Mach-Zehnder Modulator (MZM). The Antenna Element (AE) signals are then processed in the OBFN and enter an Optical Sideband Filter (OSBF). This is used to reduce the optical bandwidth as it filters out one sideband and the carrier of the optical signal, resulting in SSB-SC modulation. Because of this type of modulation, balanced detection is used. A 2×2 directional coupler (combining an unmodulated version of the optical signal and the OSBF output) and two photo diodes as shown in the right part of the Figure serve as the balanced detector.

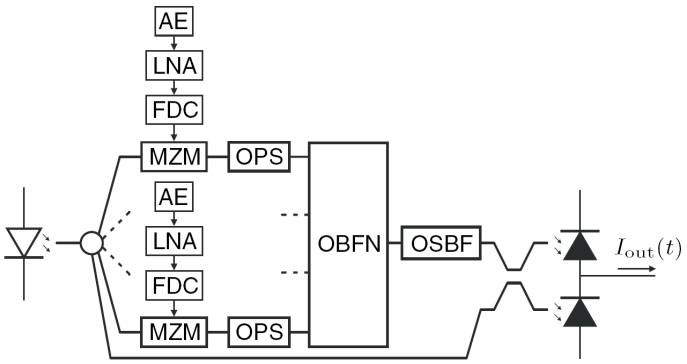

Figure 1.4: The system architecture (taken from [7]).

Control system

This part currently consists of an amplifier array Printed Circuit Board (PCB) and a microcontroller. The microcontroller is a Rowley CrossFire LPC2138 equipped with an LPC2138 Reduced Instruction Set Computer (RISC) microprocessor. It can be programmed via an universal USB. It controls the amplifier PCB that actually heats the chromium heaters on the OBFNs. This is done by sending commands to it via the RS232 port of a PC. Part of the task is to integrate the chosen and existing hardware onto a mother board, to provide the microcontroller with software such that it can tune OBFNs, and to make an interface for it on a PC.

Case study

Throughout the assignment, usage of the SMART system on a passenger aircraft serves as a case study. It is among the goals of the SMART project to provide the airplane industry with antenna systems suited for broadband data communication in-flight. This draws the need for weather-proof steerable antennas and high data throughputs.

When connected to orbiting satellites, data communication thus can be provided to offer single unidirectional communication services such as live television, or bidirectional services such as internet. This is a reason why special attention to these services is given in Chapter 2.

1.2.2 Related work

This section covers achievements and other projects that are related to this work because of similarities.

Connexion By Boeing

Broadband full duplex connectivity was provided to airplanes via geostationary satellites in the Connexion By Boeing (CBB) service. The antenna subsystem used PAAs and conventional dish antennas for transmission and reception via commercial Ku-band links. In this way it was possible to provide high speed internet connections to en route aircraft passengers [8], [9].

There are quite demanding technical requirements for *avionics* (aviation electronics). Airborne equipment should conform to *RTCA/DO-160E* [10]. This is usually driving up the costs of installation. This was also one of the reasons Boeing shut the initiative down: it was too expensive as a non-core business. Moreover, third parties were not willing to invest in it. Boeing did not want to continue the service as the market had not developed as was expected. CBB also carried out experiments using piconet GSM and CDMA2000 cells, providing mobile telephony access in-flight in cooperation with Qualcomm. By closing down the CBB initiave, also this extension stopped however.

Lufthansa

Lufthansa, interested in providing onboard internet access, used the CBB system until 2006 and called it *SkyAccess*. When CBB stopped, Lufthansa provided access to customers via satcom provider Thrane & Thrane. Their solution includes a mobile access router that is also capable of routing traffic via General Packet Radio Service (GPRS), Wireless Local Area Network (LAN), and Integrated Services Digital Network (ISDN) depending on the place of the aircraft. The WLAN interface is called GateLink, a 802.11g based connection to link aircraft to ground crew.

Starling MIJET

An Israelic company called Starling Advanced Communications offers an broadband bidirectional Ku-band satellite link via its service called MIJET. Using this link it is

possible to have full duplex internet access [11]. Ku-band is chosen for several reasons, including the capability to deliver broadband access, and the use of mature existing technology. Satellites are widely available and the ground infrastructure already exists. It uses a integrated, mechanically steered 75 cm diameter, 15 cm heigh antenna that supports bitrates up to 1.25 Mbps (uplink from the aircraft) and 10-15 Mbps (downlink towards the aircraft). Peak test rates were even up to 5 Mbps (uplink) and 20 Mbps (downlink).

1.3 Research organization

This section briefly states the research goals, the methodology used to conduct the research, and the research questions that have to be answered when this assignment is finished. Research goals define the 'what' of an assignment, and the methodology states the 'how', how an assignment is conducted. As an aid for focus and guideline to what should be answered in the conclusions, the research questions are formulated.

1.3.1 Research goals

The goals of this research are listed below in chronological order.

1. The software, hardware and performance *requirements* for the control system should be made clear.
2. Based on the requirements, a *modular design* for the controller will be developed. This design should be documented in this report.
3. Afterwards, it is a goal to develop a working *prototype* based on this design, and based on current developments of the complete system.
4. It should be reported how the system could be used, what *services* should be available, and how they could be made available.

1.3.2 Methodology

A methodology is used to identify distinct actions taken in the process of this assignment. [12]. The following steps are taken in a more or less sequential order.

- **Literature study.** Related papers, theses and books are studied to get acquinted with the subject.
- **Definition of research questions.** In this Thesis, several research questions will be answered. These questions are given further on in this section.
- **Requirements analysis.** In order to present a proper design and implementation of a controlling system, the requirements have to be studied and get clear.
- **Architecture design.** The design of the controlling system is presented based on which a prototype can build.

- **Prototype implementation.** A prototype implementation is developed to test and verify the design, and to provide input for further research.
- **Results and conclusions.** The results are evaluated. Research questions will be answered and conclusions will be formulated.
- **Suggestions for further research.** Indications in which further research could be directed are pointed out.

This thesis documents the above mentioned steps.

1.3.3 Research questions

In this section, the research questions and subquestions for this project are presented. The following questions and subquestions are identified in this research:

1. What would a *design* for a control system for OBFNs consist of?
2. To what extent is it possible to implement and evaluate a *prototype* for such a controlling system, and how could this system be used in an airplane?

Separate Chapters address the main research questions as will be pointed out in Section 1.4. The following subquestions are also addressed. Related to the first research question, the following subquestions are defined:

1. Which software, hardware and performance requirements are important?
2. How can the control system be made in such a way that it is easily adaptable to future needs, and such that it will be a flexible system?

Related to the second research question, the following subquestions are defined:

1. How to develop a prototype for the controller?
2. To what extent is it possible to evaluate the design using this prototype?
3. How could the system be used in an airplane once installed?

1.4 Thesis organization

This chapter provided an introduction to the project, its technology, related work, and the organisation of this research. The rest of this thesis is presented as follows. In Chapter 2, the context of the antenna controller is discussed. In Chapter 3, the design of the control system is presented. In Chapter 4, the implementation of this design is discussed. In Chapter 5, measurement results are presented. Finally, in Chapter 6, conclusions and answers to the research questions are given. Also suggestions for further research are mentioned.

Chapter 2

Context

What if there is a working, fully operational PAA receive system available in an airplane, as described in the previous Chapter? What could one do with the composition of the antenna front-end, modulation, beamformer, detection and controller subsystems?

This Chapter is about the context for which this system is developed, or -said differently- its possible applications. These main applications are the services of live television, in case of only unidirectional communication (reception on the airplane only), and internet access, in case of bidirectional communication. Of course the latter one relies on a more complicated system - but we will see this later on. It is not the purpose of this Chapter to be complete - the implementation of these services is worth a study and some full time employees by itself. However this Chapter tries to provide a little more insight in the details of these services and their dependabilities.

The Chapter starts with explaining the OSI model in Section 2.1, and giving a relation between the SMART project and this reference model to show what steps still need to be taken. In Section 2.2, a motivation for the television and internet services is given. Then, Section 2.3 discusses two possibilities for a system design employing a PAA with beamforming. Both designs (again, one for a unidirectional scenario and one for a bidirectional scenario) are elaborated. Section 2.4 describes what infrastructure is needed. Consequently, operational, hardware, network and software consequences are mentioned in Section 2.5. The Chapter ends with a discussion and a conclusion in Section 2.6 and 2.7, respectively.

2.1 OSI Reference model

The OSI Reference Model (Open System Interconnection Basic Reference Model) is a standard reference model defined by the International Standards Organization ISO. It is very useful when making decompositions of network infrastructures. It consist of seven layers, shown in Table 2.1.

OSI Model
Application
Presentation
Session
Transport
Network
Data Link
Physical

Table 2.1: The OSI Reference Model

Discussing these layers is useful in this context, because the context of the control system is the subject here. The network is part of that context. Therefore each layer is described below:

- *Application layer.* This layer provides the interface to front-end application processes. Example: the HTTP protocol for internet browsing applications.
- *Presentation layer.* Provides the representation and eventually encryption of the data. Example: a service that makes a XML representation of application data.
- *Session layer.* Manages and administers a session between peers. Example: secure sockets.
- *Transport layer.* Provides the transport of messages between peers. Example: The Transport Control Protocol (TCP) protocol.
- *Network layer.* Provides the routing of datagrams. Example: The Internet Protocol (IP) protocol.
- *Data link layer.* This layer provides both the Logical Link Control (LLC) and the Medium Access Control (MAC), together providing data transfer in a network, using an interface such as LAN, Wireless LAN or Bluetooth.
- *Physical layer.* This layer defines the means of transmitting raw bits onto the hardware transmission medium, such as a cable or a radio frequency.

Though current network initiatives are not always bound to just one of these layers, it is very interesting to show this model here. We use it to point out developments that still have to be done to provide a fully operational system in an airplane (as meant in the introduction of this Chapter).

At this moment, work has been done to provide a part of the physical layer interface by means of an antenna array with optical beamforming. The physical layer has partly been provided by means of modulating signals onto this interface, and demodulation after the optical components. However, there is no MAC and LLC at this moment. This Chapter points out that the development of that will not be a trivial task. Next to this, also higher layer developments should be conducted in order to provide for

example internet applications onto this system. For television this is a lot easier, since there is no bidirectional communication, and a medium access function is not needed.

2.2 Motivation

Until now, live television or internet access on airplanes is only very sparsely available. Market research however shows that customers are attracted to these forms of time spending when flying [13]. Especially on long (for example transatlantic) flights this is of interest to a large audience. Also when still docked, people appreciate this kind of services. Next to this, crew and cabin personnel, but especially maintenance personnel, rely on access to distant information for which bidirectional internet access would be very useful. The airline could communicate flight data, electronic techlogs, aircraft documentation, navigational data and more in this way. So both the customer (either as consumer or as businessman) and the provider of the transport service would benefit from live television and internet access.

2.3 Designs

In this Section two possible system designs are presented: a downlink-only design, with unidirectional communication towards an airplane, and a bidirectional link, with bidirectional communication between an airplane and a ground station.

2.3.1 A design employing a unidirectional satellite link

In the context of the FlySmart project a unidirectional communication link from a geostationary satellite to a Mobile Earth Station (MES) is addressed, the MES being an airplane. One of the intended and appealing applications is the live provisioning of television via satellite using Digital Video Broadcasting - Satellite (DVB-S). The architecture of this system is shown in Figure 2.1. A Ground Earth Station (GES) center transmits the DVB-S signal to the satellite, where it is received, amplified and transmitted to decoders such as set-top boxes. The MES is one of these receivers. Its set-top boxes decode the signal and provide the passengers with the channel of their choice, as well as auxiliary information such as an Electronic Programming Guide (EPG). This can be displayed on the in-flight entertainment systems that most long-haul aircraft nowadays have. Partly, it will also replace the back-end components such as file servers of the entertainment systems, because the data can now be streamed to the airplane instead of (costly) storing it in the plane.

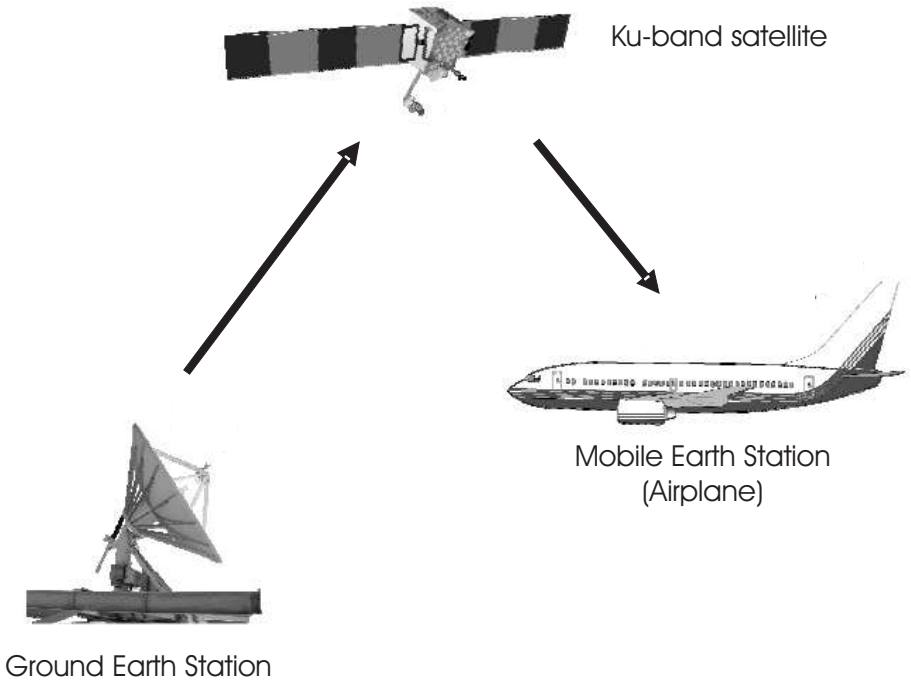

Figure 2.1: Unidirectional communication architecture.

2.3.2 A design employing a bidirectional satellite link

When a bi-directional link is employed, the above sketched service of live television is still possible, but also duplex communication is now available. Hence internet, phone calls, crew services and interactive television may be provided. In this idea two separate system designs can be identified: a synchronous connection using the same frequency band for uplink and downlink, and a asynchronous connection, using different frequency bands via possibly different satellites. These two designs are presented in Figure 2.2 and Figure 2.3.

In Figure 2.2, the scenario is sketched in which the transmission of data (from MES to GES) is achieved via the same link as the reverse link, using the geostationary satellite's Ku-band. However internet traffic is typically asynchronous (the downlink is used more than the uplink). Also the complexity of sharing a single transponder amongst many MESs needs to be regarded. A multiplexing technique must be offered when multiple MESs get data via the same transponder on the satellite. This channel accessing can be achieved using time multiplexing using Time Division Multiple Access (TDMA), code multiplexing as in CBB using Code Division Multiple Access (CDMA) or frequency multiplexing using Frequency Division Multiple Access (FDMA), eventually combined with frequency spreading techniques such as Frequency Hopping Spread Spectrum (FHSS) or Direct Sequence Spread Spectrum (DSSS). However when also the link from the MES to GES is built using a transponder on the same satellite, an increased coordination is needed to address portions of that link to the different MESs.

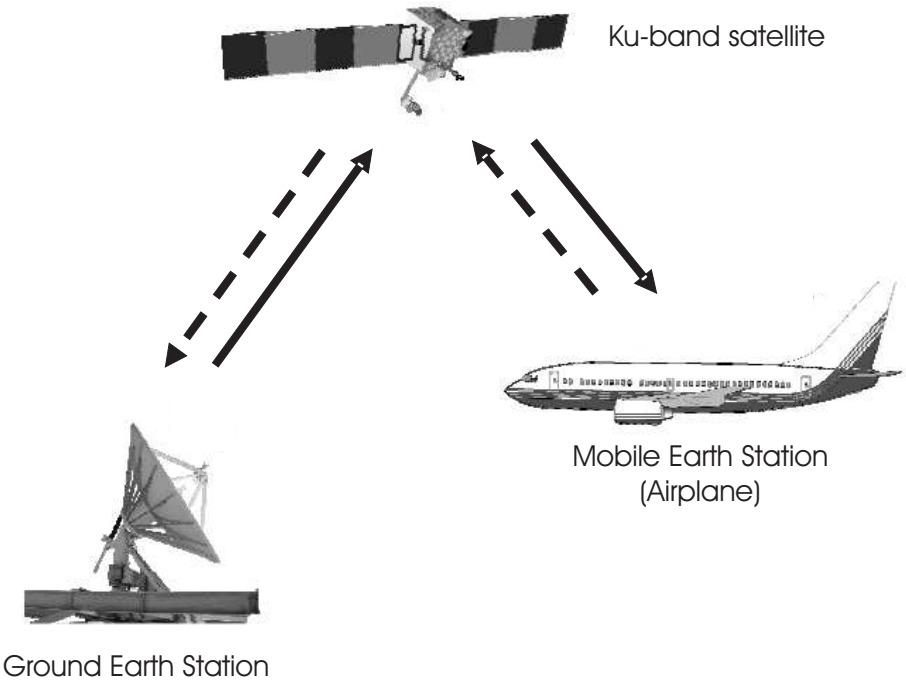

Figure 2.2: Bidirectional communication architecture with broadband uplink via Ku-band satellite.

Therefore a system is proposed in which the MES-GES link is a smaller-bandwidth, individual link from each MES to a GES, without sharing. This link is established using an L-band connection for which a number of antennas is already available, as well as ongoing research related to the integration of Ku- and L-band antennas [14]. The proposed system is shown in Figure 2.3. Another advantage of this system is that transceive functionality for L-band is already common and available for airplanes, whereas for Ku-band it is not. So not to end up with double bidirectional connections as shown in Figure 2.4, for both Ku-band and L-band, the system as described in Figure 2.3 is in favor.

2.4 Infrastructure

This section discusses the in-plane and supporting infrastructure needed for both system designs. First of all, the RF signals should be received by the MES, preferable using a conformal PAA mounted on the fuselage skin on the aircraft. Its signals are downconverted, amplified and processed as discussed in Subsection 1.2.2. The received signal is then sent to a series of set-top boxes for reception of DVB-S or Digital Video Broadcasting - Satellite 2 (DVB-S2) [15, 16]. It may also be processed by a router in case of just internet (IP) data. The DVB-S2 protocol stack can be used as it is capable of both television and IP data. The return channel for internet and interac-

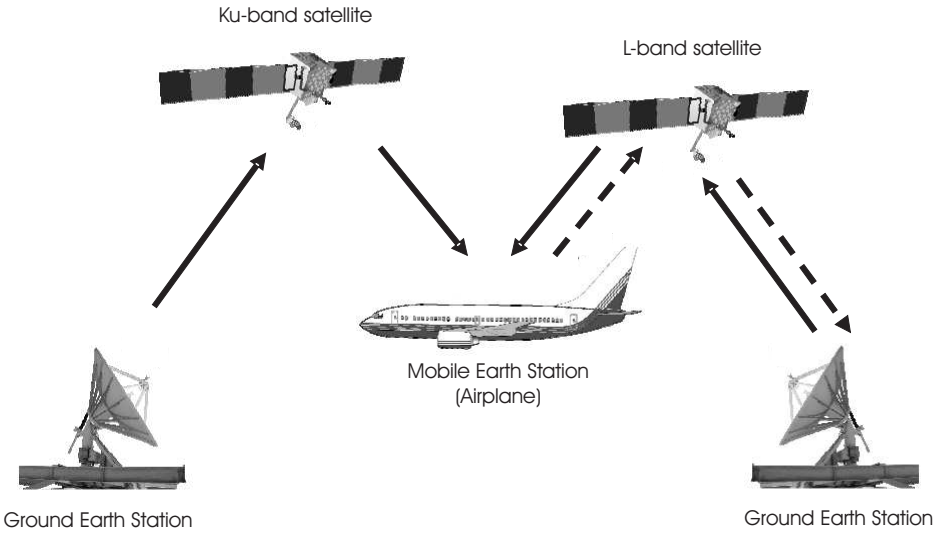

Figure 2.3: Bidirectional communication architecture with uplink via L-band only.

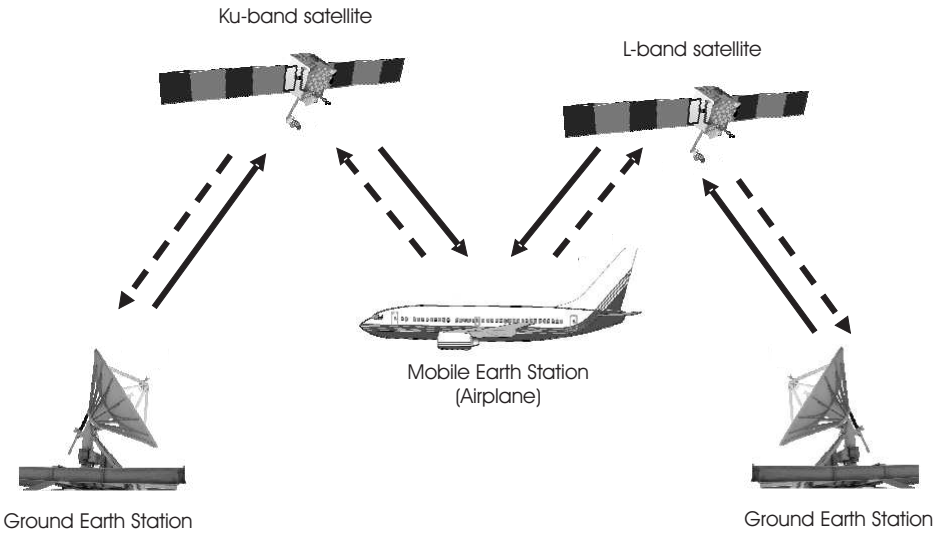

Figure 2.4: Bidirectional communication architecture with uplink via Ku-band and L-band.

tive television can be provided using Digital Video Broadcasting - Return Channel via Satellite (DVB-RCS). However an adaption needs to be made with regard to the standard solution, in case a different satellite transponder is used for the communication from the airplane to the ground. This is not a problem in the sense that DVB-RCS is designed to be frequency and frequency band independent [17]. Also, crew services and maintenance data should be separated from eventual communication with the television operator, which is generally the only practice with DVB-S2 and DVB-RCS. Therefore, the Satellite Operator as discussed in the next Section, should be a trustable party taking care of the separation of these streams. The advantage however of integration of television and IP services into DVB-S2, is that it makes the complete system (which is already quite complex) a little less complex, and already more equipment is available

which is capable of the necessary functions.

2.5 Consequences

Some of the consequences of bringing such kinds of infrastructures to airplanes, are worth mentioning. Firstly we look at some practical aspects, then at hardware aspects, then at some software aspects.

2.5.1 Operational issues

Considering the practical issues, it is sure that the systems sketched above involve multiple parties and multiple desires, rules, targets and working procedures. This will make installing and maintaining such systems quite a complex task. To give an overview of the participants, Figure 2.5 lists all the stakeholders and their relations. The customers, paying to both the Airline (tickets) and the Service provider making the wireless services possible are the main participants. The latter one may be a subsidiary of the airline. It has ties with Taxation entities for billing and administration purposes. The Airport operator is mentioned since it also is in this business, albeit at the airport. Together with the Airline it may provide combined services. The Aircraft manufacturer may include and advertise for wireless services in its planes, and operate them (such as in the CBB subsidiary). The Satellite operator operates the link between the MES and the ground, where it is connected to one or more Content providers to supply content to show in the plane, and to gateways of various Terrestrial networks.

Next to this, airline maintenance processes can now be improved, because maintenance data can now be gathered and communicated via the internet link of the aircraft, and maintenance personnel can update their findings using this connection, reducing administration, paperwork and grounding time of aircraft for maintenance.

2.5.2 Hardware issues

Installing electronical equipment in aircraft is bound to strict regulations, in Europe issued by EASA. Because of this, and because of practical issues, it is recommended to conduct the installment of the hardware in cooperation with, or outsourced to, suppliers of in-flight entertainment systems.

Regarding the test procedures that have to be fulfilled before the equipment can be certified, *RTCA/DO-160E* provides a set of regulations with respect to requirements on the products and precise procedures for testing these requirements [10]. The hardware for such systems shall be tested before it can be said certifiable, the complete certification is conducted on an aircraft as-a-whole. When properly designed, the complete system would not have very big issues in such kind of tests, however some problems

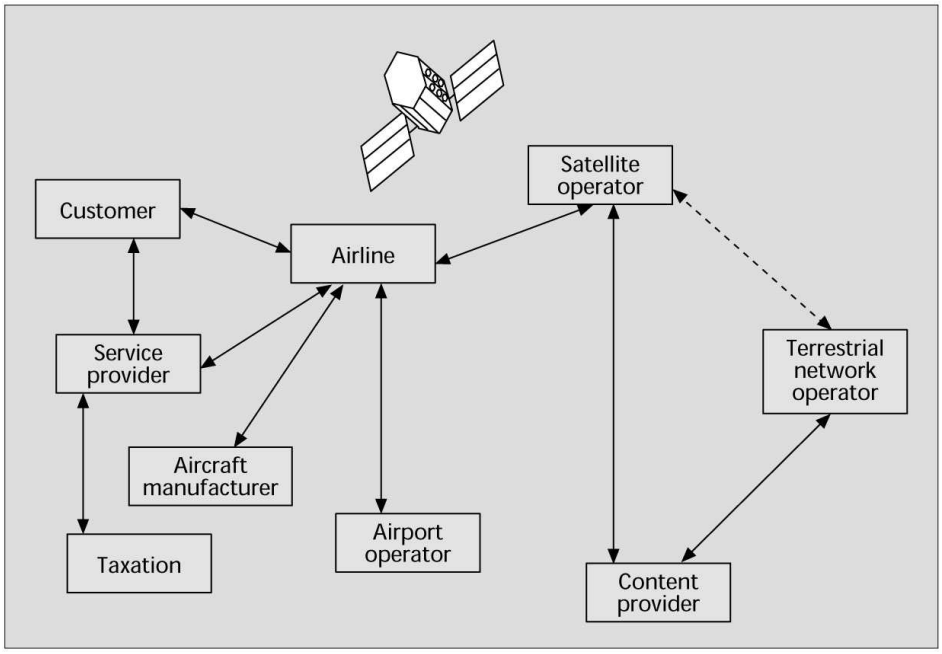

Figure 2.5: Stakeholders in a possible value chain for provisioning of wireless in-plane services (taken from [13]).

that arose in the design of the controller, such as EMI, show that still the requirements for these tests need to be considered along the development process.

Another issue regarding the antennas is the following. Typically, transatlantic flights for which the abovementioned services are interesting, are carried out at a high latitude. The geostationary satellite to point at for television service, is above the equator, so very south. Hence in order to have a large effective aperture, the PAA needs to be very large, (from the left side from the airplane all the way to the right side seen from the fuselage) or two antennas must be placed. Another quite unlikely mechanical rotation mechanism could also be placed (but this is not recommended due to is small mean time between failures). The most feasible option is placing two antennas, but still this increases the complexity of the system and the costs.

2.5.3 Software issues

As for hardware, also for software in aircraft there are regulations. RTCA DO 178 states classifications and requirements with different levels of severeness based on the importance of the software, and danger once shut down. Also here, it would not be an impossible goal to have the software running on the controller (and the tracking mechanism needed for tracking the right satellite) certified, but there need to be taken more steps to reach this level. For example, the software itself must be checked on completeness, code coverage, thread safety and correctness. But also the compiler and eventually other tools used when developing the code must be certified. In order for the

compete system, still functions for tracking satellite, angle calculation and amplitude tapering functions have to be added to the control system before a certification must be started.

2.5.4 Network issues

When communicating via satellites, one mentions longer delays than using landlines, because of the extra distance travelled by the data. For satellite television this is not too much of a problem, but for internet access this is certainly the case. The performance is affected by large and changing delays, occasionally varying error rates, band and path asymmetries and large round trip times. This affects both the Transport Control Protocol (TCP) and User Datagram Protocol (UDP). These problems are well described in [17] (Section 2.5.3). They are a reason for various enhancements proposed by the Internet Engineering Task Force (IETF) working group TCP over Satellite. Amongst them is the implementation of larger TCP windows (needed for more underway data), acknowledgement of only correctly received TCP packets (Selective Acknowledgement (SACK)) instead of cumulative acknowledgements, and proxying of data.

2.6 Discussion

A big issue when discussing such kind of systems is the feasibility of the systems discussed here. Already the CBB service was discontinued because at that time the market did not develop as expected. Not only were there very large development costs involved, also the number of paying customers did not cover enough costs. This makes it questionable whether or not such services will be viable. This will mainly have to do with installing costs, maintenance and contracts with satellite operators. A difference in this regard compared to roughly 10 to 5 years ago, is that more and more people are willing to pay to have mobile internet access, as can be seen for example with mobile data subscriptions using Universal Mobile Telecommunications System (UMTS) technology. A drawback is that such kind of systems are rather complex and it will take some more development exercises to make the system robust.

Starting in 2008, airlines were beginning to include piconets in their aircraft, to provides cellular connectivity aboard. Such services can in future be integrated with internet connectivity and crew communication services aboard.

2.7 Summary

The motivation for airborne connectivity was given and different designs for communication between an airborne MES and a GES were presented. The discussion was based on the OSI Reference Model, which was introduced in Section 2.1. Both unidirectional designs and bidirectional designs for different purposes were introduced in Section 2.3. Their advantages and disadvantages were discussed. Some of the consequences of these infrastructures were given in Section 2.4. Issues regarding the stakeholders and their relations, the hardware, the software and the network were given in Section 2.5. Finally, the feasibility was discussed in Section 2.6.

Chapter 3

Design of the Control System

The OBFN should be controlled thermo-optically, by means of a control system. This chapter presents the design of the control system, consisting of a controller and an interface for the controller, which is running on a PC. Firstly, the requirements have to be clear; they are presented in Section 3.1. As illustrative examples for the desired functionalities, some use cases are elaborated afterwards in Section 3.2. In Section 3.3 the design itself is given, consisting of a hardware and a software design. In Section 3.4 a summary is presented.

3.1 Requirements analysis

In this section the requirements for the OBFN control subsystem are presented. Software, hardware, and performance requirements are identified. For each of them, the requirements are mentioned and explained.

3.1.1 Hardware requirements

1. The hardware used should have low costs, using commercial of the shelf (COTS) components where possible, and should be fast enough to execute tasks, such as the calculation of voltages for the tuning of the optical chip, with delays in the maximal order of milliseconds.
2. The hardware should be able to provide the operator of the system with information, by means of a screen and status Light Emitting Diodes (LEDs).
3. The hardware should be easily connectible using standard plugs and cables.
4. The hardware should be designed in a modular way, so that it is easy to add parts as other parts of the total system grow, for example if the OBFN expands with more tunable elements.
5. A storage place is needed to store information about the optical chip such as offsets, number of heaters, and voltages corresponding to different tuning settings.

This storage should be fast, random accessible, rewritable and large enough for current OBFNs, and also for OBFNs of near future.
6. The hardware should be accessible using a PC but also able to operate independently.
7. In a final stadium, the hardware should conform to avionics requirements listed in DO 160/E and other relevant requirements for aviation systems.
8. In a final stadium, the complete system should be compact, packaged in a small sized box.

3.1.2 Software requirements

Next to being able to handle the use cases presented in the next Section, the control system should be able of the following software requirements:

1. The software running on the control system should make it possible to tune the OBFN, given a set of κs and ϕs, (for the tunable elements described in Subsection 1.2.1) or voltages corresponding to these κs and ϕs.
2. The software should be able to communicate fast enough with the hardware parts involved in tuning, according to the performance requirements stated below.
3. The software should be able to present operational information on what it is doing to the user.
4. The software should be easily upgradeable.
5. Debugging and logging data produced by the software, must be stored if needed.
6. The software should include drivers necessary to communicate with the storage device or devices it communicates with.
7. In a final stadium, the software should conform to avionics software requirement listed in RTCA DO 178B.

3.1.3 Performance requirements

Below, the most important performance requirements are addressed.

1. The system should be able to perform tuning within 1 ms, as this is the maximum speed of the heaters.
2. The system should boot reasonably fast, within 60 seconds.
3. The accuracy of tuning should be within 1 mV to be able to tune precisely enough. Voltages applied on the optical chip should match inputted values on the microcontroller.
4. The applied voltages should be stable and not oscillate.

3.2 Use cases

The system may be used as follows, although usage is not limited to the use cases presented below. They are presented as illustrative examples. The use cases below are given as from a user perspective. Note that the user may either be, initially, a human being, but in a later stadium, the user should be a software process.

Use case 1. Initialization, tuning and shutting down

1. Initialization. After turning the system on, the user may verify that the system starts. He may also see the number of tunable channels.
2. Setting values. After initialisation, the user may set values for the individual heaters.
3. Recall values. After initialisation, the user may set values for the heaters stored from memory.
4. Shutting down. After initialization, the user may shut down the system. The user may verify that the system exits properly.

Use case 2. Calculation

1. Initialization. See use case 1.
2. Setting phase values. After initialisation, the user may set desired phases for a given number of rings. The system then calculates the corresponding voltages and the user may verify that the system tunes the OBFN accordingly.
3. Shutting down. See use case 1.

Use case 3. Calibration

1. Initialization. See use case 1.
2. In order to make use case 2 possible, crosstalk parameters should be put in the system. The user may store these parameters in the controller.
3. Shutting down. See use case 1.

Use case 4. Loading and saving

1. Initialization. See use case 1.
2. Loading. The user may recall a set of values that form a setting for tuning of the OBFN.
3. Saving. The user may save a similar set of values. The last settings are automatically saved.
4. Shutting down. See use case 1.

3.3 Design

In this section the actual design of the controller is elaborated. This consists of both hardware and software aspects. The actual hardware design is presented first. Consequently, the software that runs on the microprocessor is presented in global. Previous work on both hard- and software have been conducted by [18], [19], and [20]. In this assignment their works have been revised, extended and integrated. The rest of this Section discusses the hardware design (Subsection 3.3.1) and software design (Subsection 3.3.2) of the control system.

3.3.1 Hardware design

The controller will be built as a modular piece of hardware. Globally, it consists of a mother board with the following properties:

- A bus architecture for stackable Digital to Analogue (D/A) converter PCBs;
- A socket to place the processor board;
- The processor board itself;
- Non-volatile memory;
- A connection to a PC.

The next part discusses the controller design in more detail.

Controller design

The first design of the controller has been published in [21].

In detail, it consists of a PCB Board, on which the D/A chip as designed by [18] is placed, equipped with an ARM7 microprocessor, the NXP LPC2138 (actually on an evaluation processor board from Rowley Associates). For proper powering there is some power circuitry. For monitoring the software on the microprocessor, and for flashing the microprocessor, there are two USB connectors. The first one actually is a USB to RS232 connector, which is connected to UART1 of the LPC2138. For nonvolatile storage, an 512kb Ferroelectric Random Access Memory (FRAM) module from RAMTRON is used, a follow up module on the FRAM memory used in [20] and [19]. However the same driver and memory structure is used. To be more explicit, the FRAM communicates with the microprocessor using the well known Serial Perhiperal Interface (SPI) protocol. This protocol is also in use for communication between the microprocessor and the DAC PCB. An actual photograph of it is shown in Figure 3.1.

The layout of the control system is shown in Figure 3.2. The ARM7 microcontroller runs its control software stored on the LPC2138 in flash memory. An Liquid Crystal Display (LCD) display may be used to display some status information. The dedicated D/A converter and amplifiers are combined on a PCB. They consist of 32 channels of

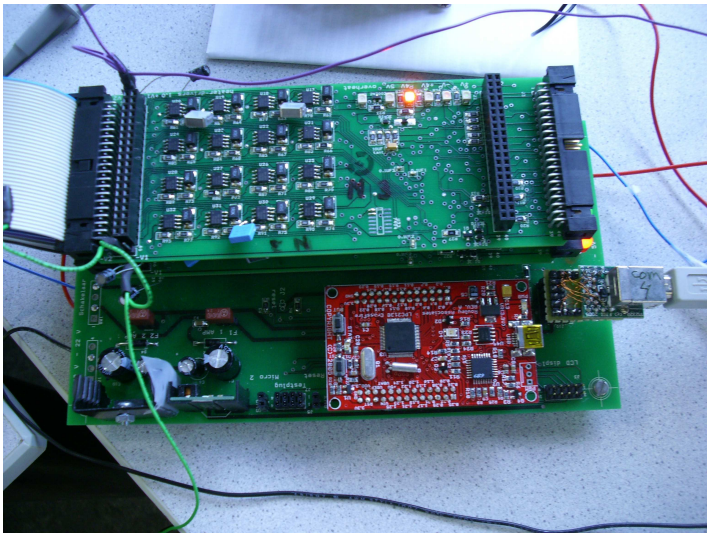

Figure 3.1: FlySmart controller with DAC PCB, (top) power circuitry, (lower left) processor board (lower middle) and USB connection towards PC (lower right). The FRAM is placed under the DAC PCB and can therefore not be seen.

14-bit resolution each. Each channel is connected to an operational amplifier (opamp) which boosts the voltage level 6 times. In this way, 0 to 30 volt can be addressed with 14 bit resolution.

The D/A converter is connected to the microcontroller. Choosing SPI here allows for tuning of larger optical systems in the future. This is not only because SPI easily allows individual addressing of duplicate slave modules, but it is also a lot faster than its alternative, I^2C. Storage should be provided to store and recall tuning settings. It was chosen to do this in a nonvolatile FRAM. The control system is able to communicate with a PC using USB, via the Universal Asynchronous Receiver and Transmitter (UART) of the microcontroller. It may however operate in standalone mode as well.

When this design is implemented, it is a more featured, more channel, higher precision follow-up heater driver of International Business Machines (IBM) boxes that are currently in use in the TE laboratory to conduct thermo-optical tuning.

3.3.2 Software design

From a very abstract point of view, the functionality of the control software in the microcontroller can be described by the Data Flow Diagram (DFD) as displayed in Figure 3.3. Firstly, initialisation takes place in the *Initialise* process. The system component drivers for the UART, FRAM and DAC are initialised, and connections to these components are made. Then, the program will start to listen to an external device issuing tuning commands in the next process, *Receive Tuning Parameters*. These commands

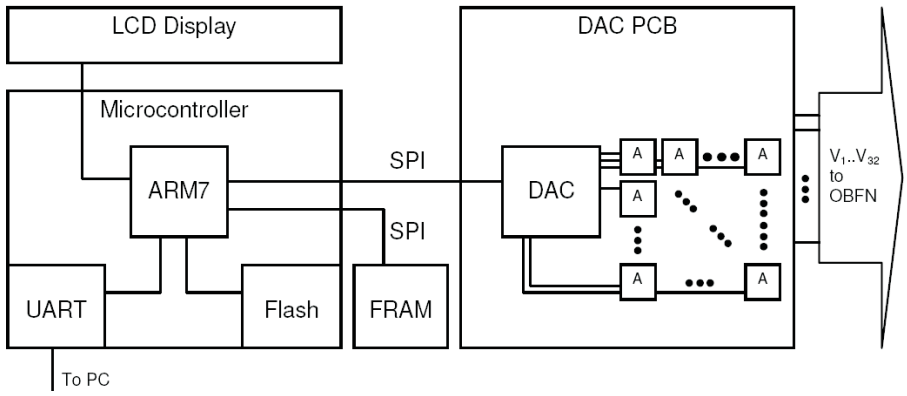

Figure 3.2: Architecture of the control system. In the left part, the microcontroller is shown, with a connection to a display and a PC via its UART. Flash memory is used for the controlling software storage. A DAC PCB is shown on the right, connected to the microcontroller using SPI. It consists of a 32-channel DAC and an amplifier for each channel; the amplifier outputs go to the OBFN. For storage of tuning parameters, an FRAM is used.

are actuated in the process *Actuate Tuning*, after which the reception of tuning parameters usually will be repeated. Otherwise, the system may close connections, turn off the DAC chip and close down in the *Shut Down* process.

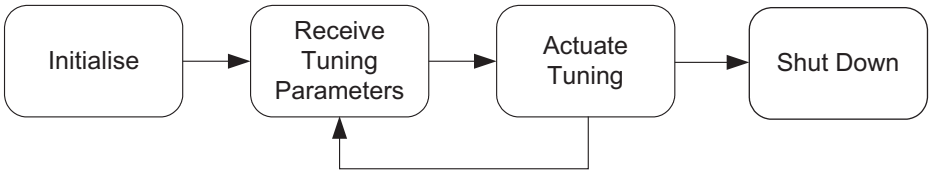

Figure 3.3: Controller software DFD.

Software on PC

To make a complete control system, an interface is needed to provide the means of operating the controlling. Regarding the software running on the PC to issue commands to the microcontroller, our initial strategy will be to reuse the software that was developed to control the IBM boxes. This is LabView software that contains VISA (Virtual Instruments) blocks to communicate over RS232. In the next Chapter, the implementation of these functionalities is discussed in detail.

3.4 Summary

This chapter presented the design and design decisions of the control system. The control system consists of a controller and an interface to operate the controller. The requirements for the system were addressed for hardware, software and performance aspects in Section 3.1. Some illustrative use cases were presented in Section 3.2. In Section 3.3 the system design was addressed. Both the hardware design and the software design were discussed. A detailed architecture of the components composition was given in Subsection 3.3.1 as firstly presented in [21].

Chapter 4

Implementation of the Control System

This Chapter discusses the process of the implementation of the control system. During this project, three consecutive releases of the controller have been delivered: a basic version aimed at tuning the 4×1 Yeti[1] optical chip and a more advanced version with graphical interface for the 8×1 FlySmart[2] optical chip. Finally a version that could calculate channel values had been implemented and tested. In this Chapter both the preliminary version and its two follow-ups are discussed, though we would like to note that this division into versions is not because the controllers are very different. They are more identifiable milestones in the development process and therefore nicely describable sequentially.

The remainder of this Chapter is divided as follows. Firstly, the implementation approach is given in Section 4.1. Then, the three versions of the controller are described. At the end of this Chapter, we discuss the Graphical User Interface (GUI) on the PC separately, and the steps taken to drive multiple Digital to Analog Converter (DAC) Printed Circuit Boards (PCBs). Finally a summary with conclusions is presented.

4.1 Implementation approach

In this section the implementation approach is discussed. The high level functionality of the controller was decribed in Section 3.3.2. Figure 4.1 describing the data flow (the DFD) from that Section is repeated below , to refresh the to-be-programmed functionality.

The main task here is to implement this functionality and to provide an interface

[1]Yeti is the given name of a optical chip produced with funding of the Nederlands Instituut voor Vliegtuigontwikkeling en Ruimtevaart (NIVR) project 'OBFNSys'. It is named after an extinguished animal species,which has become -without any reason- usual in products made with NIVR fundings.

[2]The FlySmart chip is an OBFN chip produced for and named after the 'FlySmart' project, a collaboration between NLR, UT, Lionix BV and Cyner Substrates. The FlySmart project is the Dutch part of the European 'SMART' project.

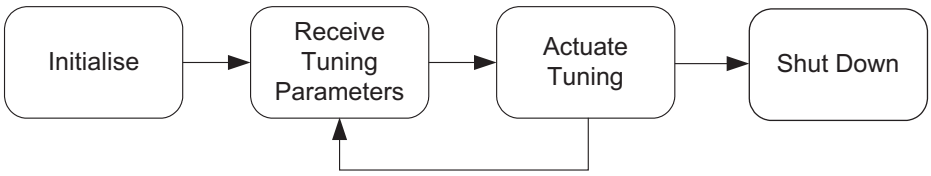

Figure 4.1: Controller software DFD.

to operate it. Firstly the hardware and software that are needed to implement this are reviewed. The next Sections discuss the milestones of this process..

4.1.1 Hardware

The hardware consists of a few main components, which are a microcontroller, a DAC PCB and a mother board. The motherboard contains power circuitry, an FRAM, and sockets for the microcontroller and DAC PCB as discussed in Section 3.3.1. The implementation of the individual hardware components was already conducted, but the components still had to be integrated and tested where necessary, which was firstly done when developing the Yeti controller. After discussing the implementation approach for the software, the Yeti controller is discussed in detail.

4.1.2 Software

The software running on the microprocessor should communicate with the PC (from which it receives tuning parameters) and control the different hardware elements in the control system (actuate the tuning). These two processes are usually repeated, otherwise the program may shut down in the Shut Down process. The software to accomplish these tasks, is written in C and is implemented following a Model View Control (MVC) design principle. For the LCD and SPI driver functionality, two libraries from Lammerink Electrical Engineering (LEE) were used as a starting point. This forms the heart of the control part of this design principle. The data is stored in the FRAM and accessible via a virtualization layer that addresses the actual storage and retrieve functions. In this way, the user of the data model is not bothered with the data location and the protocols to retrieve or store the data. The view part of the design principle is running as a LabView program on a PC. This program is separately discussed in Section 4.5.

The files and a detailed Unified Modelling Language (UML) diagram listing all code content, is given in Appendix C. Now we will discuss the steps taken to come to this final code package, via the versions of the controller that were developed subsequently.

4.2 Implementation of the Yeti controller

This section discusses the implementation phase of the controller for the Yeti chip in detail. The motherboard, the other components and their drivers, and the communication between PC and controller is discussed sequentially in the next three Subsections.

4.2.1 Motherboard

When all components were available we started measuring the mother board, which houses a socket for the microcontroller, voltage level conversion circuitry, the FRAM and a socket for the previously designed DAC PCB. The initial idea was to use a 30 V power supply, from which 8 V is taken for the low-voltage components (FRAM, microcontroller, DAC). This did not work because the voltage level converter had to dissipate to much energy. Therefore both 30 V voltage supply (for the opamps on the DAC PCBs) and 8 V supply for the before-mentioned low voltage components were installed. As decided earlier in [18], both negative and positive voltage supplies are used to mitigate the ground.

4.2.2 Other components

Having a properly fed mother board, firstly the FRAM functions were implemented and tested. Measurements to verify this were done with an oscilloscope (see Subsection 5.1.1), and procedures were programmed to verify successful storing and loading of values.

Afterwards, the same was done for the DAC PCB, firstly without 30 V supply to the opamps, and when the SPI was properly set up, the voltage supply to the opamps was turned on to verify output voltage adjustment functionality. The measurements regarding this component are discussed in Section 5.2.

4.2.3 Communication between PC and controller

The next step was to enable interaction between a PC and the controller, so functionality to provide this, using RS232, was implemented. This way the USB-RS232 conversion module FT232R from Future Technology Devices International (FTDI) [22] could be verified. It provides a USB Serial Port when turned on, for which drivers need to be get from FTDI. It is implemented without support for flow control, and used at 115200 bps, which is a lot faster than the 9600 bps used in the IBM boxes.

Now we are able to operate the controller interactively, so the next step is to add functionality to control the channels of the DAC PCB. This is implemented in a backwards compatible way with the IBM boxes, such that current LabView programs as discussed in 4.5 could still be used. Therefore the ASCII command

`wrch␣CHANNEL,VALUE↩` is implemented as the method to set channels of the controller to specific values. The carriage return to end the command is denoted by the ↩ sign.

4.3 Implementation of the FlySmart controller

This section discusses the implementation of the controller for the FlySmart chip in detail. This controller is a follow-up of the controller discussed in the previous Section. Some parts were changed; these parts are described in this Section. There were a few targets for improvement for this version:

- *User interaction.* It should be easier for the user to set the controller to the right properties.
- *Speed improvement.* Setting the right properties should be completed much sooner.
- *More channels.* For the FlySmart chip, 38 channels need to be controlled, which cannot be handled by a single DAC PCB.
- *Setting calculation.* This version of the controller should be able to calculate voltages, when phase shift and coupling numbers are given by the user.

These improvements are discussed in the next four Subsections.

4.3.1 User interaction

In the first version of the controller the user had to input the channel number and the desired voltage value in a terminal screen. Not only this works slowly, but it also lacks possibilities to input complete settings for multiple channels and quickly reset or adjust values. In this version, the strategy as described in Section 3.3.2 will be implemented, using a GUI to enable a user easy access to the control functions. This is discussed in more detail in Section 4.5.

4.3.2 Speed improvement

Speed improvement was one of the targets for this version. This was reached by first omitting all unnecessary delaying code statements, after which refreshing voltages, the DAC chip still took 3 till 4 seconds before its registers were updated. Hence it is interesting to discuss a technique used to gain a speed-up by increasing the protocol efficiency. Since all channel values may be known on forehand when starting the tuning, there is no explicit need to iteratively issue commands to the microcontroller to set just one channel at a time. So instead of sending a `wrch` command for each channel, one may send just one command `wrAll` followed by multiple channels and their values.

After all, one knows all channel values when issuing this set. The speed-up of this loop-lifting technique can easily be seen, using the definition of protocol efficiency η_p

$$\eta_{\mathrm{p}} = \frac{I}{I+O} \tag{4.1}$$

in which I is the actual protocol information, the useful protocol data, and O is the protocol overhead, which is the overhead data of the protocol, such as command identifiers.

For the single write command statement `wrch channel, value`, the efficiency is

$$\eta_{\mathrm{p},1} = 6/(6+7) = 0.46 \tag{4.2}$$

since the number of informational characters is 6 (2 for the channel number and 4 for the value to set), and the overhead, which is the number of characters of `wrch␣,`↩, is 7. A space and newline should be added as one ASCII command, the latter one is again denoted by the ↩ sign.

For the loop-lifted `wrAll` command, the protocol efficiency is

$$\eta_{\mathrm{p},2} = \frac{\sum_{i=1}^{n} I(i)}{O + \sum_{i=1}^{n} \left(I(i) + O(i)\right)} \tag{4.3}$$

in which i iterates from channel 1 to n. The efficiency now is 0.73 for 32 channels ($n = 32$, $O = 6$, $I(i) = 6$ and $O(i) = 2$) and will increase for more channels. When this method is implemented, the command sent to the microcontroller would look like `wrAllCHANNEL`$_i$`,VALUE`$_i$`;CHANNEL`$_{i+1}$`,VALUE`$_{i+1}$`;`↩ which will be a sequence up to n channel-value pairs long.

After this bulk tuning (tuning set values at a single time) it is still possible to tune a single channel, sending just 1 channel-value pair. Next to speed-up, another advantage of bulk tuning is its application to the functionality described in the next Section.

4.3.3 Addressing more channels

For the FlySmart chip, more channels need te be controlled when the complete chip (as shown in Figure D.1 in Appendix D) is tuned. There are 48 heater bondpads, 42 of them are actually used on the optical chip itself. Four of them are ground, so there are 38 channels needed to tune the complete chip. A single DAC PCB can only drive 32 channels. Hence another DAC PCB is needed for this chip. The actual extension needed in the hardware to control multiple DAC PCBs, is discussed in Section 4.7. The FlySmart controller finally used is shown below in Figure 4.2. It consists of a motherboard, microcontroller, and 2 DAC PCBs, so all 38 channels of the FlySmart chip can be tuned.

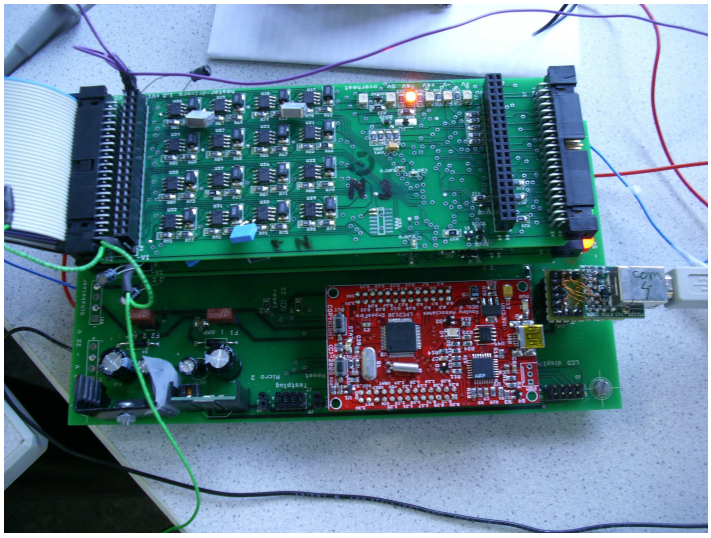

Figure 4.2: FlySmart controller with stack of two DAC PCBs.

4.4 Setting calculation

So far, the basic controller with terminal input and the improved controller with more advanced tuning options have been discussed. Now we shall discuss the final controller, for which setting calculation functionality has to be included. After this part, the extension in tuning range (from 32 to 96 channels) is discussed. Firstly, we look at the setting calculation functionality.

This functionality is implemented using the following step plan:

1. The user provides a desired ϕ value for a ring.
2. The controller system takes these values and calculates voltages for these values using a mathematical model, with stored parameters, as explained afterwards.
3. The controller shows the calculated values, and the user may apply or discard them.

The background of this step plan discussed sequentially now.

Tuning

The group delay behavior of ORRs can be tuned using κ and ϕ, setting the peak delay and the resonance frequency, respectively. The peak delay itself is actually tuned using a tunable phase shifter in a MZI. The resonance frequency is tuned using a phase shifter in the ring resonator.

It is desirable to provide high level tunability for this. Therefore in this project a functionality has been created to set the per-ring desired phases to implement one part of this high-level tunability. With this functionality, the position of the resonance frequency can be set. The rings have periodic behavior, so there are multiple resonance frequencies, which are all one Free Spectral Range (FSR) apart. Therefore it makes no

sense to tune more than one FSR. We may now speak about tuning the *phase* of the ring. Tuning it to a phase of π is the same as tuning it exactly in the middle of two resonance frequencies, or in the middle of two original FSRs. For example: when the FSR of the ring is 16 GHz, and we tune the ring with a phase shift of $\pi/2$, it is tuned $16/4 = 4$ GHz, the first resonance is shifted 4 GHz.

The control system has to translate this phase shift into proper voltages. So, the per-ring desired phases are sent to the microcontroller, using one of the commands listed in Appendix A. When the microcontroller receives such a command, it calculates the corresponding voltages for the ring belonging to these 2 channels. We will now explain how this is performed, first in a scenario without crosstalk, then in a scenario where crosstalk between voltage applied to different rings is taken into account.

No crosstalk

The calculation for the ORR ϕ_{desired} is related to the voltage V_{out} as follows:

$$\phi_{\text{ring}} = a{V_{\text{phi}}}^2 + \phi_{\text{offset}} \tag{4.4}$$

In which a is the directional coefficient and ϕ_{offset} is the phase offset needed to compensate for fabrication process differences. When the a and ϕ_{offset} are calibrated, the voltage V_{out} that has to be applied can easily be calculated (in case there is no crosstalk):

$$V_{\text{out}} = \sqrt{(\phi_{\text{desired}} - \phi_{\text{offset}})/a} \tag{4.5}$$

V_{out} can become negative using equation 4.5, which should be prevented by choosing ϕ_{offset} right.

With crosstalk

However, the phase change of a ring is not only dependent on the a from the ring itself. It can also be affected by heat dissipated by, and current flowing through other heaters.[3] Therefore it is necessary to address this crosstalk from other heaters in matrix form:

$$\vec{\phi}_{\text{desired}} = \mathbf{A}\vec{V}^2_{\text{out}} + \vec{\phi}_{\text{offset}} \tag{4.6}$$

in which $\vec{\phi}_{\text{desired}}$ is the vector with the desired phase values. $\mathbf{A}$ is a two-dimensional square matrix sized the number of tunable rings. On its diagonal, the as of the heaters belonging to the ring itself are written. On the other positions, the crosstalk factors are put (for example, on position $\mathbf{A}_{2,4}$ the crosstalk effect factor on ring 2 coming from

[3] For better understanding, the effect of this latter phenomenon, electrical crosstalk, needs more study than conducted and planned in the framework of this project.

ring 4 is put). $\vec{V}^2_{\text{out}}$ are the squared voltages, and $\vec{\phi}_{\text{offset}}$ now is the vector with phase offsets ϕ_{offset}. Both are similar to the previous paragraph, but now written as vectors.

Since we like to have the voltages calculated, it is convenient to rewrite (4.6) to

$$\vec{V}^2_{\text{out}} = \mathbf{A}^{-1} * (\vec{\phi}_{\text{desired}} - \vec{\phi}_{\text{offset}}) \tag{4.7}$$

$\mathbf{A}^{-1}$ is the inverted matrix $\mathbf{A}$, so the condition

$$det(\mathbf{A}) \neq 0 \tag{4.8}$$

should hold which is the case if the effect of the heaters on the diagonal is greater than the effect of the crosstalking effects in the matrix $\mathbf{A}$. (This can easily be seen in a 2×2 matrix $\mathbf{M}$, where $(M_{1,1}, M_{2,2})$ is the diagonal, and

$$det(\mathbf{M}) = M_{1,1}M_{2,2} - M_{1,2}M_{2,1} \tag{4.9}$$

For larger matrices Laplace expansion may be used.

Having the squared voltages, it is easy to put them onto the right channels once the mapping between rings and channels is known. However, the squared voltage in 4.7 may not be negative, which can be prevented by rightly choosing the offset values in $\vec{\phi}_{\text{offset}}$. [19]

Next steps

For formula (4.7), functionality has been added to the microcontroller to calculate voltage values while compensating crosstalk effects. Using the measured data presented in Section 5.4.2, these effects were measured, calculated and stored in the FRAM. Now the microcontroller is able to calculate a set of outputs given a desired set of phases for the rings. All commands that can be isued to operate the controller, including the commands related to the tuning and the calculations described above, are listed in Appendix A.

4.5 Controlling the channels

A LabView program has been made to set the voltage values for the channels of the controller. The philosophy is that the LabView programs is a graphical layer on top of a process that sends plain text commands to the controller using RS232. This makes tuning easier, and settings can be stored in the program. For this, the Virtual Instrument (VISA) functionality (a standard I/O language for instrumentation programming) of LabView has been used. The GUI is shown in Figure 4.3. The main part consists of sliders to set the channels of the controller to a specific value. The values can eventually

be saved in order to store complete tuning settings. The LabView progam has the option to set the COM-port and the flow control (in default mode, no flow control is used for communication with the controller). Moreover it shows status information and the data that is written over the RS232 connection towards the controller. As shown, the GUI is quite large, which is not a problem for up to 40 channels, but was reconsidered after the addition of more channels. The program is very extendable though, because adapting it to set more channels is very easy: only the width of the slider panel in the middle needs to increased (then more sliders will appear and they can be enabled by double-clicking) and the iterator in the block diagram must be set accordingly to the number of channels.

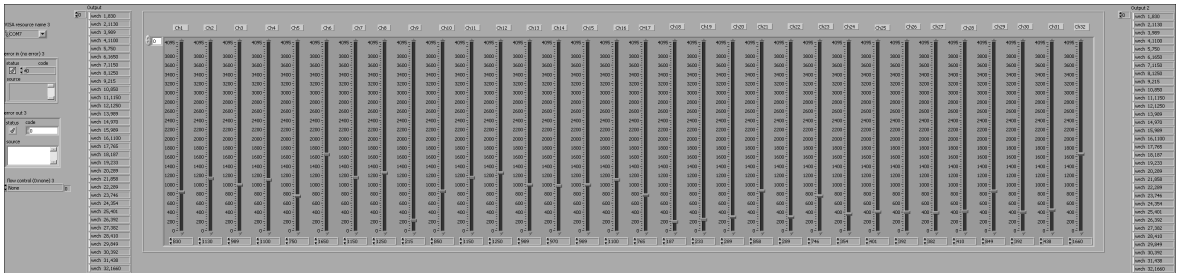

Figure 4.3: GUI for setting the channels of the controller.

4.6 A reconsideration of the interface

Regarding Figure 4.3, it is clear that this interface is not very user friendly for a very large number of slider bars. Also, from a technical point of view LabView was not the most ideal solution, as it did not work event-based (so each time a slider was adjusted, in the worst case this update is only sent after first processing all other slider values first, which consumes unneeded time) and it also issued commands when nothing changed. Moreover, due to the fact that LabView implements RS232 communication using VISA with native code, some closing statements did not work as wanted, as the controller is not a VISA device, but LabView thought it was, resulting in 'blue screens'. Therefore, a new Java interface was implemented, whose GUI is shown in Figure 4.4.

This Java program is named (after its purpose) Heater Controller. It is implemented using a Model View Control (MVC) pattern and an Observer/Observable pattern. The main classes are shown in the UML diagram in Figure 4.5. The Model View Control pattern and the Observer / Observable pattern can clearly be identified.

The Observer is the GUI model class, which registers to two Observers: the control class for the data model and the control class that represents the communication with the RS232 interface. The control class for the data model stores and loads data from XML files representing persistent Java Properties, so tuning settings can be stored

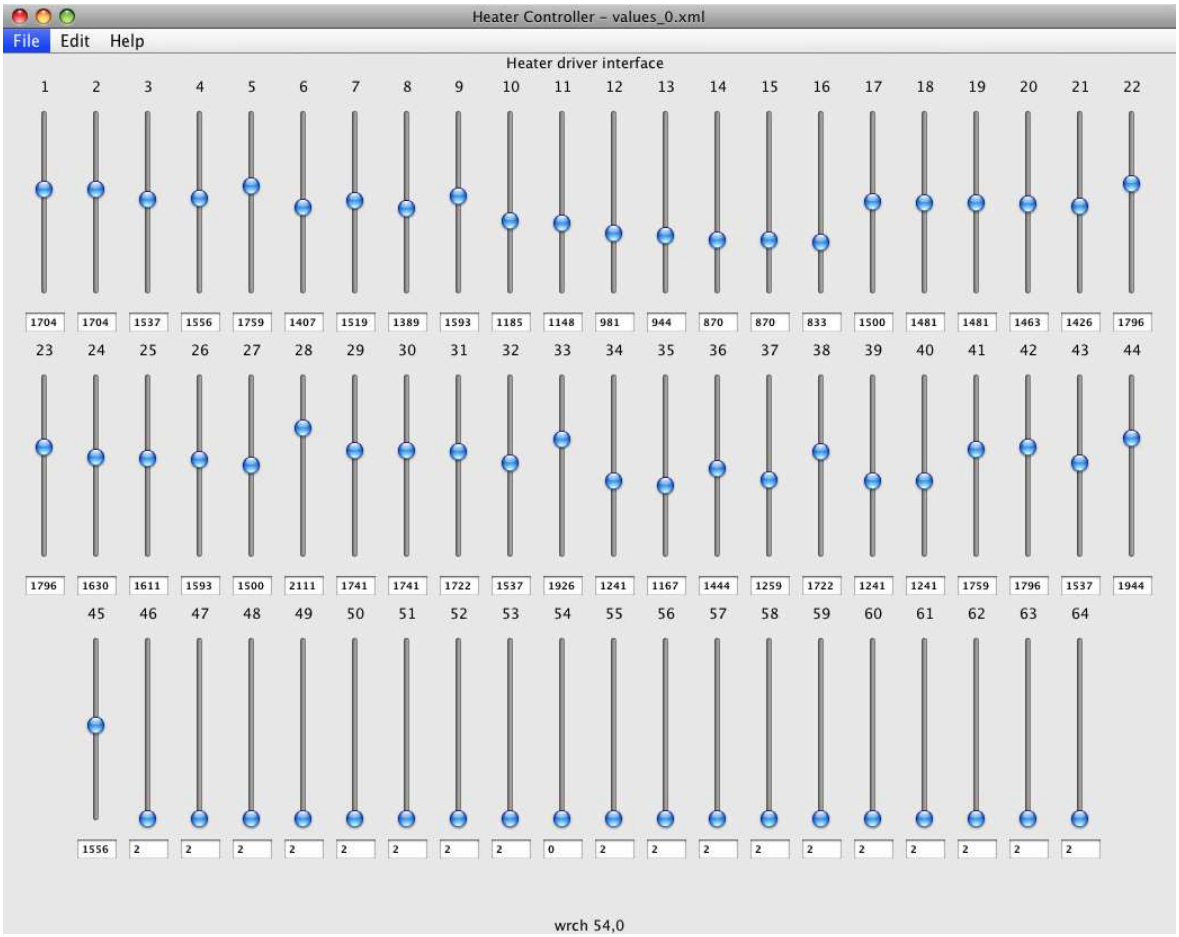

Figure 4.4: Novel Java GUI for setting the channels of the controller.

and loaded. The Property framework is also used for persistent GUI settings such as the number of slider bars. The control class for the RS232 interface is the only Windows specific part in the program, as it uses the Windows specific com32.dll-based javax.comm library for InputStream- and OutputStream-connections with a COM-port. Because this program is event-based, it actuates changed channels faster, and the need to implement the loop-lifting algorithm was not there anymore.

4.7 Extending the number of DAC PCBs

The possibility to extend the number of connected DAC PCBs was implemented in a late stadium of this assignment. The final result is shown in Figure 4.6. It makes use of the bus structure of the PCBs, to stack them on top of each other. This bus structure is also used to feed each PCB with the same data lines of the SPI protocol. On the PCBs, the wiring is the same for each board, except for the _Chip Select (CS) data line. So each PCB shares the clock, input and output data lines of the 4-wire SPI protocol. To address a single board however, only the _CS line of that specific board

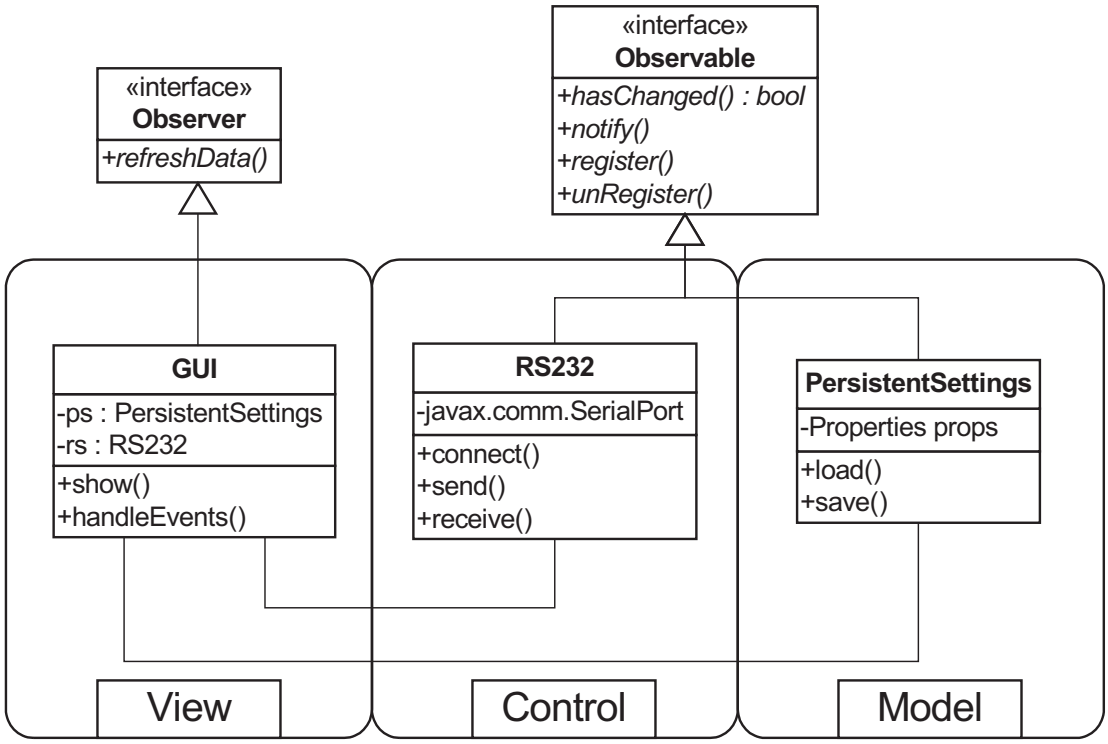

Figure 4.5: UML class diagram showing the main Java classes of the Heater Controller software package.

should be put low. To address all PCBs in parallel, the software puts all _CS lines low, so all PCBs will listen to the next command. Whereas the individual addressing is useful for setting voltages to certain channels, the parallel addressing is useful for commands that hold for all PCBs, such as initialisation and reset. The implementation of this extension was done by adapting only the SPI_Mod.c driver file, in such a way that it is backwards compatible with the rest of the program. So it does not affect the other process steps in the software. The mother board was adapted with extra wiring for the new _CS lines, this should be integrated in the PCB in a new version.

4.8 Summary and conclusions

The implementation of the controller was presented. This chapter discussed the hardware and software aspects of implementation approach. The subsequent versions of the controller were described, for the Yeti and FlySmart chips. The extensions for speed-up, crosstalk calculation, the interface on PC, and finally multiple DAC PCBs, were discussed in detail. A Java interface replaced a LabView interface for more user friendliness, and better performance. Therefore the loop-lifting algorithm was not implemented. Extra wiring and driver modifications were added to drive up to three DAC PCBs, enabling the controlling of 96 channels of 14-bit each, with the system shown

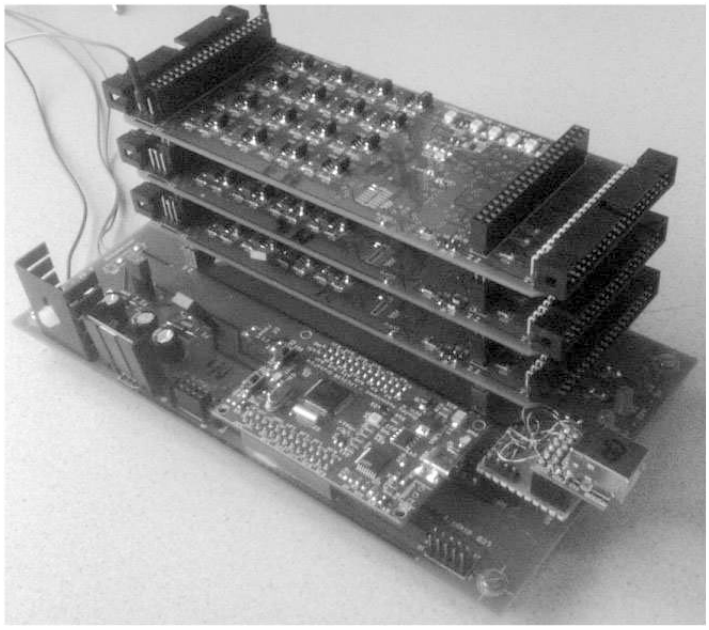

Figure 4.6: Controller with stack of three DAC PCBs.

in Figure 4.6.

Chapter 5

Measurements

This Chapter describes measurements conducted on components of the controller, on the complete controller, and on a 4×1 and an 8×1 optical beamforming chip. The optical chip measurements were conducted using the hardware designed and implemented in this project. The following Section (5.1) describes the measurements on the electronic controller components. The measurements on the output channels of the controller are presented afterwards in Section 5.2. Then, the tuning measurements are given for the 4×1 Yeti chip in Section 5.3. Next in Section 5.4, measurement results for the FlySmart[1] 8×1 chip are presented, including crosstalk measurements. A short discussion is then given in Section 5.5, in which some issues that turned up during the measurements are described. This Chapter closes with conclusions on the measurements in Section 5.6.

5.1 Controller measurements

During the implementation of the controller, the SPI protocol implementation (for communication between the microcontroller and the DAC and FRAM) was verified. Also, the DAC output register voltages were measured using a multimeter, to check the behaviour of all channels. The next two Subsections describe firstly the FRAM and secondly the DAC measurements. The SPI protocol itself is very thoroughly explained in Appendix B.

5.1.1 FRAM measurements

The SPI protocol is used in both the communication and controlling of the FRAM and the DAC chip. To debug the mother board of the set-up and the SPI protocol implementation, the correct working of these two components had to be verified. This was done using an oscilloscope. Because there are different modes in the SPI protocol,

[1] See the footnotes at the beginning of Chapter 4 for an explanation on 'Yeti' and 'FlySmart'

and the FRAM uses mode 0 or 3, which are -unexpectedly- other modes than the DAC, it was time-consuming to debug this part of the hardware. The modes and all the wire namings of the SPI protocol are given in Appendix B.2.

In Figure 5.1 an operation to store data in the FRAM is shown. It shows the clock signal above, the CS in the middle, and the Master Out Slave In (MoSi) below. In three consecutive write cycli when _CS is low, the following happens: an opcode bitword to indicate a write action is sent, the address is sent, and at last the value 31 is written. (The most significant bit is a don't care due to the limit of the number of bits per address (6 bits needed for 32 channels). The sweep afterwards is below the threshold for a logic 1).

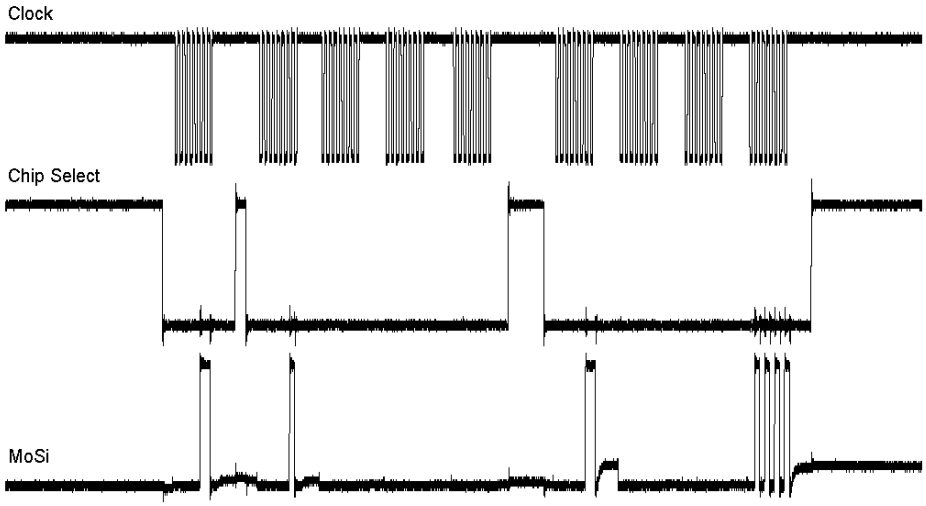

Figure 5.1: Measurement of the SPI protocol at the FRAM. Above: clock signal. Middle: chip select signal. Three times the _CS goes from high to low. Below: MoSi. In three consecutive write cycli when _CS is low, the following happens: an opcode to write is sent, the address where to store the value is sent, and finally the value to store in the FRAM is written.

In Figure 5.2 a closer look at the synchronization timing of the signals of the SPI protocol between microcontroller and FRAM is shown. It can clearly be seen that the value of the MoSi signal is stable at the rising edge of the clock signal.

5.1.2 DAC measurements

The DAC chip uses a different version of the SPI protocol, version 1.[2] In Figure 5.3 a write statement to set a channel to a specific value is issued to the DAC chip. Three clock periods of 8 bits each are shown. Channel 13 (first clock period) is set to 0. The

[2]Again, for the different versions of this protocol, we refer to Appendix B.

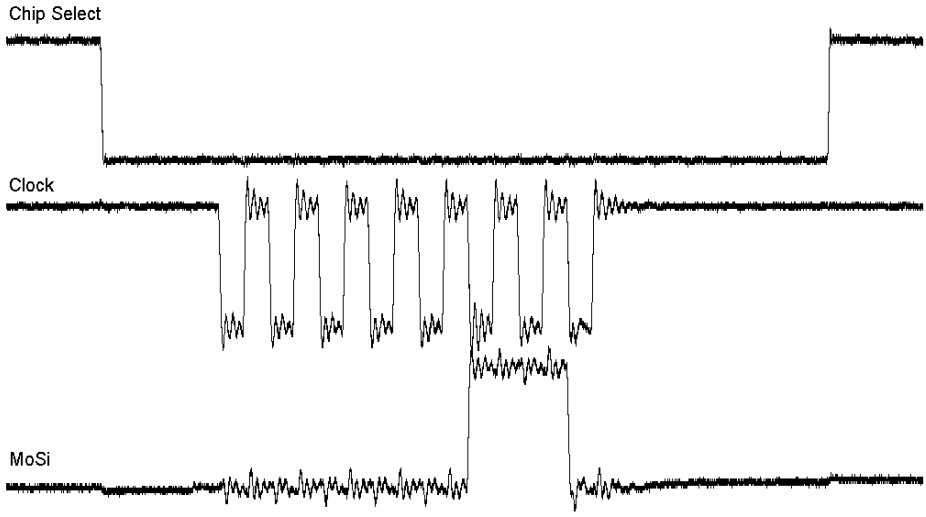

Figure 5.2: Detailed oscilloscope measurement of the SPI communication between microcontroller and FRAM. Above: _CS signal. Middle: clock signal. Below: MoSi. With the _CS low, the value on the MoSi line is stable on the rising edge of the clock signal.

second and third part of MoSi are all logic 0, except for bits 15 and 16 which are don't care since the DAC is 14-bit.

In Figure 5.4 a detailed version of the right part of this measurement is given. It can be seen that the stability of the clocked bits of the MoSi line is now at the falling edge of the clock signal.

5.1.3 Discussion

Because the 2 components using the SPI protocol (FRAM and DAC chip) use different protocol modes, the SPI driver had to be adjusted to change the mode each time one of the two components is used. It was chosen to initialise the SPI driver for proper DAC operation, and to change the mode whenever the FRAM would be used. When the FRAM is ready, the driver changes back the mode for the DAC. This way it is assured that the component that needs to be addressed most quickly (the DAC), can be addressed with minimal delay. More on the C code implementation of the SPI protocol can be found in Appendix B.

5.2 Channel measurements

After implementation, testing and debugging of the software and the SPI driver (discussed in Chapter 4), it was possible to test the voltage outputs of the channels of

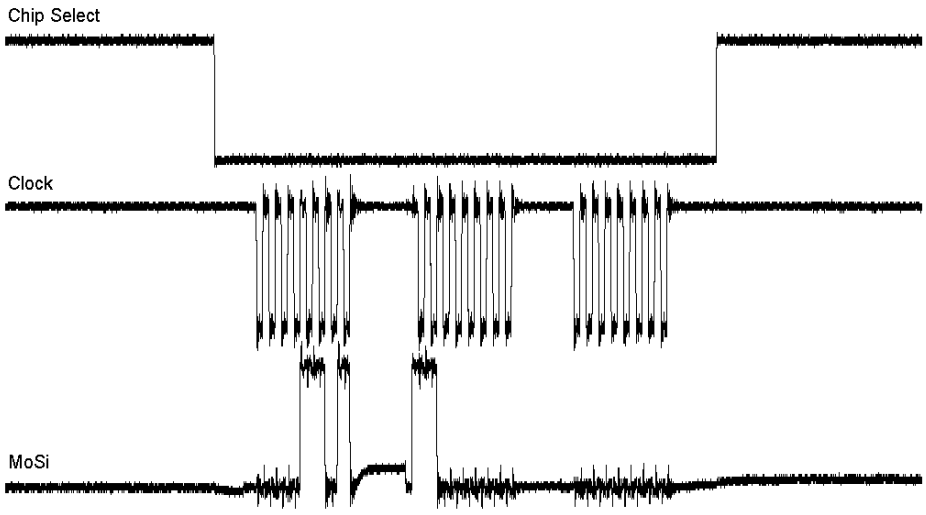

Figure 5.3: Oscilloscope measurement of the SPI communication between microcontroller and DAC. Above: _CS signal. Middle: clock signal. Below: MoSi. Three clock periods of 8 bits each are shown. Channel 13 (first clock period) is set to 0.

the DAC PCB. A cable had to be made to make a connection between the DAC PCB and the optical chip. For the Yeti controller, this was done via a LED Box, useful for check which was connected to which heater. This box has 32 LEDs, indicating positive voltage supply with red light and negative voltage supply with green light. This way it could be checked which opamp and which channel on the DAC PCB, corresponded to which wire in the cable, and what its number and voltage polarity were. The software in the microcontroller was programmed to assure that LED 1 on the LED Box corresponded to Channel 1 on the controller. This is wrong, but we will see that in a minute. First the mapping of the Yeti controller is briefly shown below, then the FlySmart controller, where the error was fixed. The cable for the connection was made out of a computer industry standard Integrated Drive Electronics (IDE) 40-pin flat cable. The first 8 wires have positive polarity (channel 1-8), and then 8 have negative polarity (channels 9-16). Then, 8 wires are ground (GND), and wires 17-32 again have positive and negative polarity. On the connector on the DAC PCB this looks as shown in Figure 5.5. Looking from the opamp side with the DA converter on the bottom of the PCB, the upper-right above pins are channel 1-8 with positive polarity. Below them, channels 9-16 are situated with negative polarity. Next to this, 8 pins are denoted ground (4 on the upper row, 4 on the lower row). Left below, channels 17-24 are put (negative polarity). Above these, the last 8 channels, 25-32, are situated with positive polarity.

All channels were tested separately for one DAC PCB. When doing the channel output tests with the FlySmart controller, with 2 DAC PCBs stacked, the LED Box

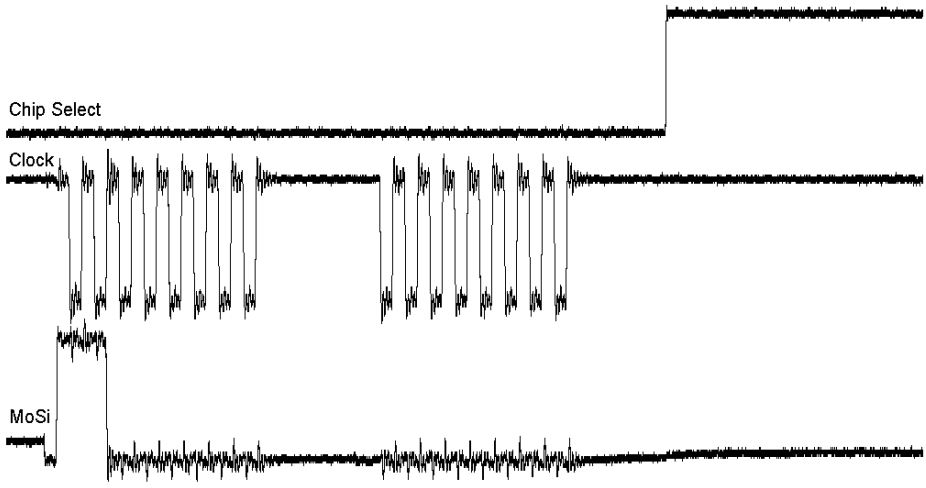

Figure 5.4: Detail of oscilloscope measurement of the SPI communication between microcontroller and DAC. Above: _CS signal. Middle: clock signal. Below: MoSi. With the _CS low, the value on the MoSi line is now stable on the falling edge of the clock signal.

Figure 5.5: Schematic of the IDE Connector (only for the Yeti contoller) with the key pin numbers on the DAC PCB. The grey pins are ground, the upper row has positive polarity, the lower row negative polarity. Note the strange channel mapping, to comply with the LED Box.

was not used anymore. It only then turned out that the mapping of the channels in the LED Box was wrong, not that of the DAC PCB. Therefore the software was adjusted such that the upper row of the DAC PCB connector was 1-16 (positive supply) and the lower 17-32 (negative supply). Actually, the adjustment was just to remove the previous channel changes in the software. The connector for the FlySmart controller is shown in Figure 5.6.

After recording voltage levels for all channels of three delay settings and two filter settings, stability over time was shown by a rebuild of the measurement set-up at the National Aerospace Laboratory (NLR) premises. It turned out that the same voltage level settings should be used to reach the same settings as recorded at the TE laboratory. This means the tuning is stable in time: after finding a set of voltage levels, one can store this and use it for tuning later on, for both delay and filter settings. However at NLR some voltage oscillation problems were found; they are discussed in Subsection 5.4.4.

Figure 5.6: Schematic of the IDE Connector (only for the FlySmart contoller) with the key pin numbers on the DAC PCB. The grey pins are ground, the upper row has positive polarity, the lower row negative polarity. Now the channel numbering is normal.

The tuning settings were entered on a terminal emulator program on a PC, interfacing with the microcontroller via USB. This process is documented in Section 4.3. The voltage settings were entered and verified with the LED Box and a multimeter. This showed very good agreement with the level expected after issuing it on the PC. So, when the user issues a command to set a channel to 14 V, that channel will also operate at 14 V. Once all channels operate, it is possible to conduct tests on optical chips. These are discussed afterwards. Similar to the delay measurements presented in [23], the measurements on the optical chips are conducted using the phase shift approach, by means of the network analyzer shown in Figure 5.13.

5.3 YETI chip

The Yeti chip was the first chip used for obtaining measurement results in this assignment. The chip consists of a 4×1 OBFN and an OSBF. This section discusses delay and filter measurements conducted on the Yeti chip.

5.3.1 Delay measurements

In Figure 5.7, measurement results of group delay responses of the 4×1 OBFN Yeti chip are given. The rings are tuned, such that a flat group delay spectrum is obtained over a signal bandwidth of roughly 1.5 GHz. The largest delay value is approximately 1.3 ns (corresponding to 39 cm in air). The maximum ripple is about 0.05 ns. The measurements show already that a sufficient delay bandwidth can be reached for satellite TV communication (10.7 – 12.75 GHz). However this chip is not meant for this frequency range, whereas the FlySmart 8×1 OBFN chip is.

5.3.2 Filter tuning

The optical sideband filters of the discussed chips are also tuneable with the control system as they rely on the same components: ORRs and tuneable couplers. They can thus be set using some channels from the controller. In Figure 5.8, the filter response is shown with a periodic response behavior, measured with a 'sweep' time of 0.5 s. This

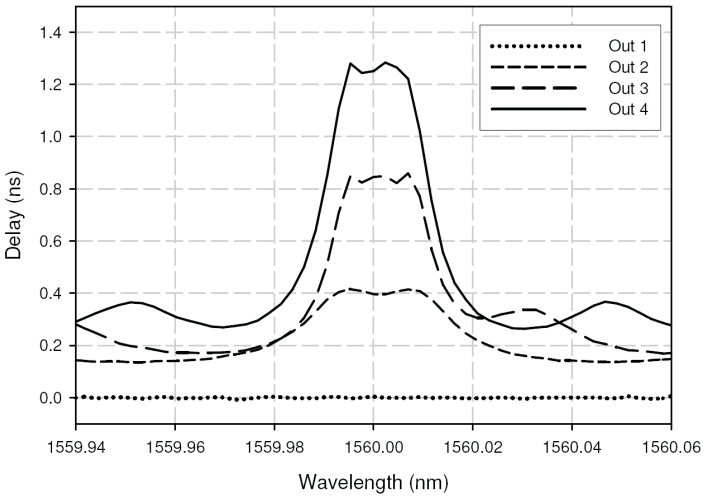

Figure 5.7: Measured group delay responses at the 4 outputs of the Yeti 4×1 OBFN.

measurement was conducted with the set-up of Subsection 5.4.3. The laser output current varies (sweeps) over time, and with it its output frequency. So at time $t = 0$ s the laser has its lowest current and frequency, and and time $t = 0.5$ s it has its highest current and frequency. This explains the increasing power response in the three pass bands in the Figure. Since it cannot directly be measured, it is not exactly known which frequency corresponds to which point on the x-axis. (A way to get to know this is using laser heterodyning.) The measured suppression is in the electrical domain, so the suppression is (-25 dB - -75 dB) = 50 dB.

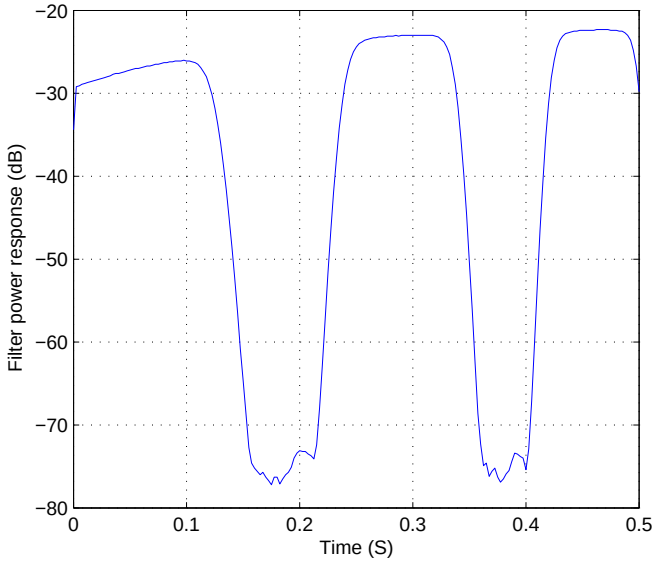

Figure 5.8: Filter output power response in time domain of the Yeti OSBF.

5.4 FlySmart chip

The FlySmart chip was also measured. The next part discusses delay measurements, and again filter measurements. Then an extensive look at the behavior of the rings taken, and crosstalk between the rings are measured and described.

5.4.1 Delay measurements

In Figure 5.9 delay measurement results conducted on the FlySmart chip are shown. This Figure shows both the group delay (in the inset) and the signal phase shift. In the inset, three group delay curves are shown, for 0 ns, 0.75 ns, and 1.5 ns, which are all more or less flat over a bandwidth of 1 GHz (enough for satellite TV tuning, after downconversion). In the Figure itself, the recovered RF signals, over the frequency range from 1 to 2 GHz, are shown in terms of RF-to-RF phase response. Hence this is the response after modulation, delaying, and detection. The phase response for the 0 ns group delay has zero phase response over the signal frequency, the other two show good agreement to the corresponding delay values. The ripple in the responses mainly has to do with the optical phase fluctuations, which comes from slight fluctuation in temperature and position of the fibers. In future implementations, integration of modulators, lasers and splitters onto a single chip will solve this problem.

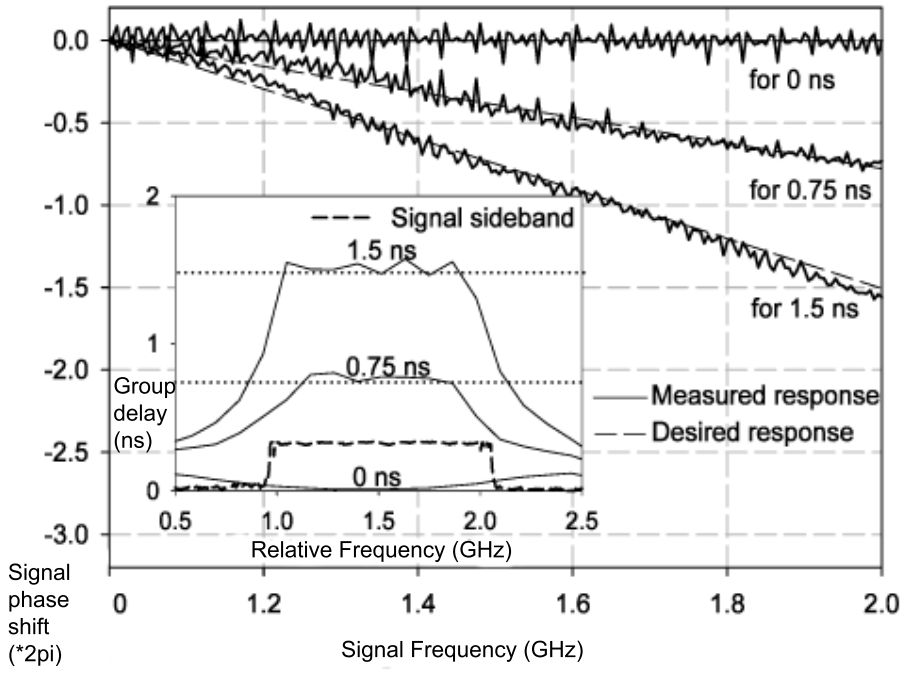

Figure 5.9: Phase delay and group delay response of the FlySmart 8×1 OBFN (taken from [24]).

5.4.2 Filter measurements

The filter was tuned using the same method as discussed in Subsection 5.3.2. In Figure 5.10 the optical power response is shown. Note that now the relative frequency is put on the horizontal axis. This was measured using laser heterodyning. This means using a second, fixed-frequency laser. The superposition of it on the modulating laser can be measured, and so the exact frequency of the modulating laser is known. This way it can be seen what the final result of the sweeping is in terms of frequencies instead of time, as in Figure5.8. The OSBF is tuned resulting in a pass band and a stop band. The dotted line shows the simulated filter behavior with 0.3 dB/cm losses. The continuous line follows the dotted line closely as can be seen in the Figure.

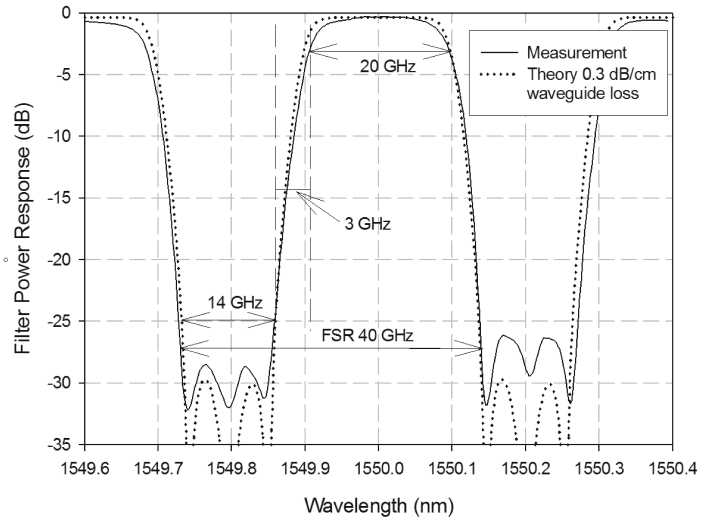

Figure 5.10: Filter response of the FlySMart 8×1 chip.

5.4.3 Ring behavior measurements

Some important parameters of each of the ORRs in the OBFN of the FlySmart chip were measured. This was done to obtain the behavior of the rings. These parameters are:

- $\Delta V\phi_{2\pi}$. With zero voltage applied as reference, this is the voltage needed to obtain a shift of 1 FSR, or said differently, 2π phase shift. The values are obtained at a κ (coupling) of 14.00 V. This κ should be the same for each ring to obtain results that can be mutually compared. It cannot be set to zero, because then no light is coupled in the ring and the resonance frequency cannot be seen. Therefore it should be a fixed value for all rings. All other phase and coupling heaters for other rings should be set to zero for no crosstalk.
- $\Delta V\phi_{\text{ref}}$. The reference voltage, needed to obtain a pre-set reference phase. This voltage corresponds to the phase in which all rings have the same initial phase

ring no.	1	2	3	4	5	6	7	8
$\Delta V\phi_{2\pi}$	24,6	25,8	24,6	24,2	26,2	24,8	25,43	25,3
$\Delta V\phi_{ref}$	11,45	14,2	21	9,15	14	20,4	11	8,43

Table 5.1: Voltage measurements on ORRs in the FlySmart chip.

shift. Said differently, this is to align the resonance frequencies of the rings, being the values ϕ_{offset} in equations (4.4) to (4.7).

With these measured values, listed in Table 5.1, the parameters for the algorithm described in Section (4.4) can be calculated.

It is repeated from Section 4.4 that if there is no crosstalk:

$$\phi_{\text{ring}} = a{V_{\text{phi}}}^2 + \phi_{\text{offset}} \tag{5.1}$$

The initial a value per ring is calculated as:

$$a = \frac{2\pi}{{\Delta V\phi_{2\pi}}^2} \tag{5.2}$$

This gives the slope of the relation phase versus squared voltage. The a-value determines, at a κ of 14.00 V, the factor needed to calculate a desired phase or a desired voltage. We come back to this after stating that the initial offset b is calculated as

$$b = \phi_{\text{offset}} = a{\Delta V\phi_{\text{ref}}}^2 \tag{5.3}$$

which is needed to align the resonance frequencies of the rings as explained at the beginning of this subsection.

Now we have all information to set a single ring using Formula 5.1, since a and ϕ_{offset} are now known. So with (5.1) and the a-values in Table 5.1, individual ring phases can be related to their corresponding voltage.

Crosstalk measurement

However, there is crosstalk in the system. This means that when tuning one ring, it not only affects the to-be-tuned ring but also there is a (smaller) influence on other rings. This effect can be both negative and positive: it may lead to a phase addition on another ring, or to a phase substraction. In both cases, a compensation has to be made.

Various aspects of this crosstalk effect are described in [20]. However, due to thermal instability of previous chips it was not possible yet to quantify this effect. The Yeti and FlySmart chips are very stable in thermal sense, and they are mounted on a copper plate kept to 30°C by a Peltier element, with a water-cooled 20°C (room temperature)

copper plate under it to guide away all heat produced in the optical system, including the heat produced by the heaters. The copper plates are clearly seen in Figure 5.11.

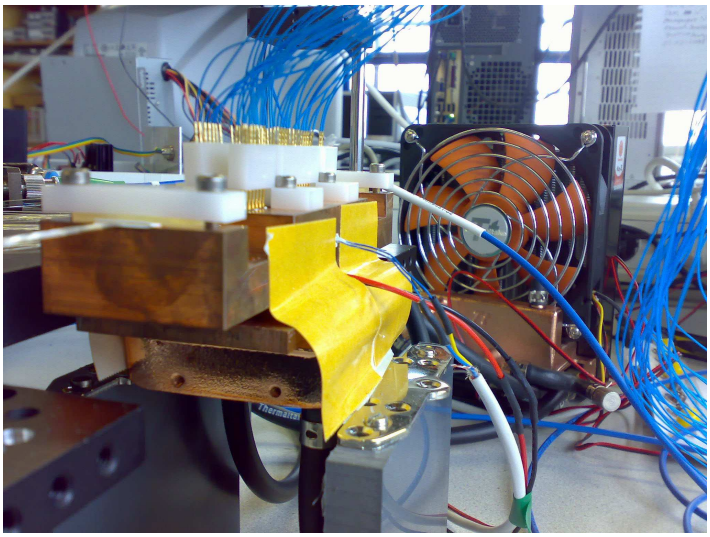

Figure 5.11: Copper footprint of the optical chip to guide heat. Below the lower plate, tubes can be seen, used for the water cooling. The water pump with fan is shown behind the copper parts. The wires on top and at the right are for the controlling of the heaters.

Because the chips are stable, they allow for measurements on the effect of crosstalk between the rings. This is discussed in the next part of this Chapter.

Crosstalk measurement setup

This measurement was done using the set-up using laser current ramping, to see the response of the system for multiple FSRs. It is shown in Figure 5.12 and it works as follows: the laser current source is ramping between a lower and a higher current, such that the optical frequency of the laser changes over a certain bandwidth. This is tuned such that the network analyzer, measuring a time window more or less in phase with the ramping, shows the power response at each frequency. The ramping is set to include multiple FSRs, so on the network analyzer we see multiple resonance frequencies of the rings when they are tuned. The laser temperature is kept stable using a Temperature Controller (TEC). The modulator modulates an RF signal onto the optical carrier. This goes through the optical chip, (DUT) after which it is amplified by means of an Erbium Doped Fiber Amplifier (EDFA). Then the signal is detected and fed to the second port of the network analyzer. Here the forward transmission in the system, S21, is measured and plotted. It shows a non-linear frequency domain (horizontal axis) and the received power (vertical axis). At the resonance frequencies of the measured ring, a dip is visible, due to the roundtrip losses in the ring. Hence at these places, the

measured power is about 15 dB lower.

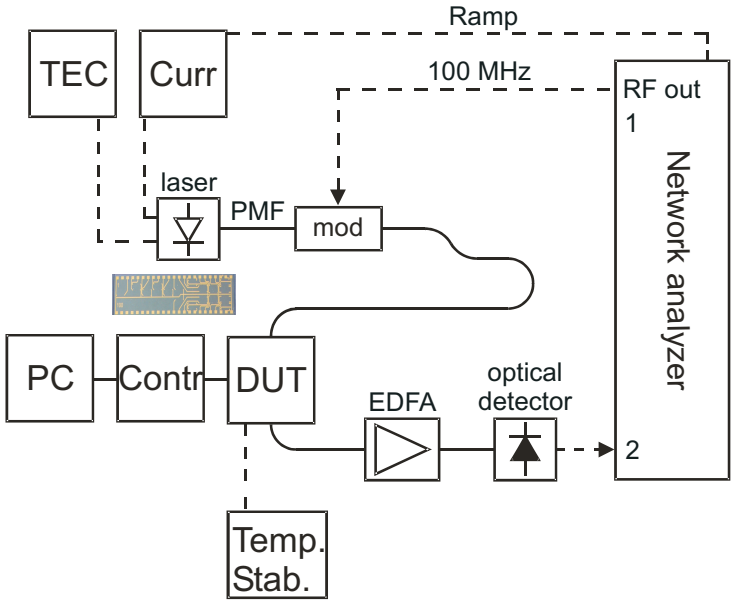

Figure 5.12: Measurement setup. TEC = Temperature Controller for the Laser. Curr = Current controller for the laser. Its current is sweeping, controlled by the Network Analyzer. DUT = Device under test, the optical chip in the photo between the laser and the Contr block. Mod = modulator. PC = computer, interfacing the controller (Contr). EDFA = Erbium Doped Fiber Amplifier.

For illustrational purpose only, a photo of a measurement on the network analyzer is shown in Figure 5.13. On a x-axis (which is not linear), three resonance frequencies of an ORR can be identified. On the y-axis the measured power in dB is shown.

Crosstalk measurement execution

The measurement was done as follows: given the a-value (from Table 5.1) of some OBFN ring i, this ring i is shifted by 2π. Its offset b nor the filter is taken into account because that would make it unnecessarily complicated (we need this later on however). Another ring j (the *crosstalker*) is tuned to its 2π voltage. Now we may see a phase change in i due to this crosstalker being turned on. This effect is, when there is only thermal crosstalk, independent of any other rings. So when all other rings are tuned to zero V, we may quantify the crosstalk effect of j. Its effect is compensated, using the control system, by adjusting the voltage i is set to. The factor that belongs to this effect, is then stored on the (i, j) position in the $\mathbf{A}^{-1}$ matrix. The numbers of the measured rings in the OBFN of the Yeti chip are numbered as labeled in Figure 5.14.

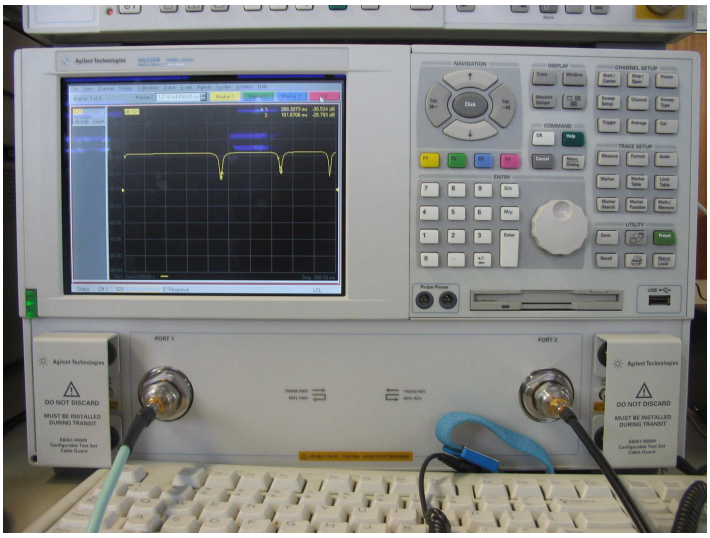

Figure 5.13: Network analyzer plotting a result of a delay measurement.

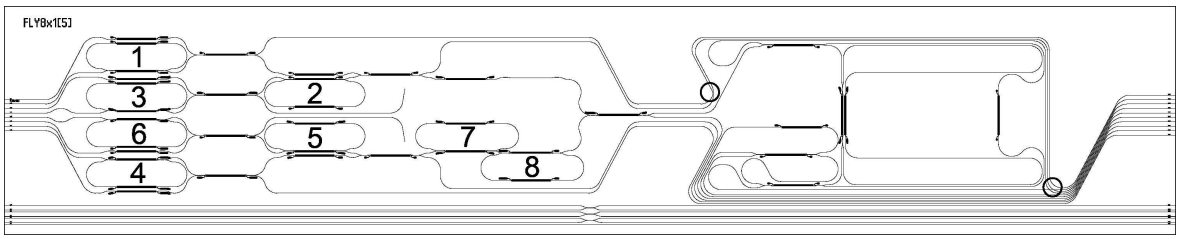

Figure 5.14: Numbering of the OBFN ORRs for the crosstalk measurements The thin black lines are the waveguides. The short, thicker black lines are the resistor used for the tuning. The two circles on the right indicate waveguide crossings.

Crosstalk: electrical and thermal

When measuring the crosstalk between phase shifters of the ORRs, it turned out that there exists not only negative but also positive crosstalk. In the normal case, a ring will get an extra phase shift when other phase shifters (on other rings) are also turned on, because it heats up more than when the crosstalker is turned off. Between rings 5, 6, 7 and 8 however, this effect was the other way around. Let us explain this with two examples:

1. When ring 5 is tuned at 20 V, and ring 6, 7 or 8 is also tuned to 20 V, the phase shift of 5 becomes less and 20 V has to be increased to 20.3V to compensate. This was not expected.
2. When ring 5 is tuned at 20 V, and ring 1, 2, 3 or 4 is also tuned to 20 V, the phase shift of 5 becomes more due to thermal crosstalk and 20V has to be decreased to compensate the thermal crosstalk, which was as expected.

This effect could not have to do with a falldown of the voltage supply to ring 5, as this was measured to be stable. All ground bondpads of the chip are connected to the

ground of the connector shown in Figure 5.6. It is suggested therefore to rewire the heater wires to verify this problem. Other effects that could cause this problem are interference of the other stage, or a small voltage decrease. Most likely a reshuffle of the wires will help, because rings 2 and 5 are supplied with negative voltages and rings 7 and 8 with positive voltages. The layout and the connection of the chip are found in Appendix D. More import in the scope of this assignment is however, that the control system can cope with both crosstalk forms (electrical and thermal) so from that point of view it is not a problem.

Crosstalk matrix

The crosstalk matrix finally looks as shown in Table 5.2. It is an 8×8 matrix since it describes behavior for the 8 rings in Figure 5.14, so apart from the self-tuning, there are 7 possible crosstalking OBFN rings for each ring. The self-values, which means the effect from a heater on the ring itself (no crosstalk), are in bold. This is the diagonal of the matrix. The other values are all crosstalk factors. They are either negative due to thermal crosstalk, or positive due to electrical crosstalk. In the latter case they are in italics to show that this effect is only there between rings 2, 5, 7 and 8. It shows therefore that both positive and negative crosstalks can be taken into account.

It is clear that in the matrix, ring 7 and 8 do not have influence on ring 1, 3, 4 and 6 and viceversa. This is because they are geographically 'far' away, as shown in Figure 5.14. 2 and 5 are closer to 7 and 8, they do have influence on each other. 1 and 3 are very close to each other. Therefore, their thermal crosstalk effect is high. 3 and 6 and 4 are also close to each other, which also gives a reason for high thermal crosstalk. The matrix itself is not symmetric because the non-diagonal factors depend on the size of the bold value in the same row, which is different for each row, and also because of measured differences that is not yet explained. Moreover, it should be measured whether indeed thermal crosstalk is proportional to V^2, and electrical crosstalk is proportional to V. Also, the exact resistance of the heaters should be measured for better understanding.

Doing this measurement and obtaining these values was nontrivial, due to the time consumption (it took 3 days) and the sensitivity of the optical chip to temperature. The self-values had to be adjusted on day 2 with respect to day 1 of measuring because different values were measured, due to measurement setup changes. However it is not exactly known how large the error of the compensation may be as it leads to a ripple error in the delay bandwidth tuning and the tolerance to this error is dependent on the application.

Moreover, it is laborious to do this measurement for a complete chip with 8 rings at once. It is therefore suggested to use this algorithm for smaller tuning scenarios. The OSBF is a very good option here.

	1	2	3	4	5	6	7	8
1	**97.89**	-2.11	-7.11	-1.4	-1.4	-2.11	0.00	0.00
2	-2.21	**107.59**	-1.6	-1.11	-1.6	-1.11	*2.50*	*2.50*
3	-4.6	-1.11	**100.27**	-0.61	0.00	-3.61	0.00	0.00
4	0.00	0.00	-0.61	**97.99**	-0.61	-7.11	0.00	0.00
5	-0.61	-2.11	-0.61	-2.11	**107.59**	0.00	*2.50*	*2.50*
6	-1.61	-0.51	-2.91	-3.51	-1.61	**101.07**	-1.11	-0.61
7	0.00	*2.50*	0.00	0.00	*2.50*	0.00	**100.00**	*2.50*
8	0.00	*2.50*	0.00	0.00	*2.50*	0.00	*2.50*	**95.00**

Table 5.2: $\mathbf{A}^{-1}$ matrix self-values (in bold) and crosstalk values (electrical crosstalk stated in italics

In Appendix D it is stated which controller channels and input versus output channel combinations were used to measure the rings.

5.4.4 EMI measurements

Both in the power supply line towards the controller, and on the DAC PCB at the opamps, voltage oscillations were found. Also because no decoupling capacitors were added at the opamps, voltage oscillations arised at the DAC PCBs.

During tests at NLR, it turned out that these oscillations had negative impact on the optical phase in the chip. This problem was tackled by adding a 33 nF decoupling capacitor between ports 4 and 7 of a positive supplied opamp. Later on, the other DAC PCBs at the TE laboratory were also supplied with decoupling capacitators on both the positive supplied opamps and the negative supplied op-amps of 100 nF as also stated in the datasheet of the opamp. It is suggested to follow these statements in a next design of this DAC PCB for each of them. The addition of the decoupling capacitor for the positive supplied opamp is illustrated in Figure 5.15. Decoupling with more capacity should also be added at the end of the power supply wires.

After this addition, the EMI source was much lower, but at the TE laboratory this has never caused a problem, and it was not possible to provide clear measurements (and figures) on that.

5.5 Discussion

It turned out that the tuning using LabView and the Java GUI was fast enough (less than 0.5 s) for all the measurements with respect to the results being plotted on the network analyzer. Hence it was chosen not to implement the loop-lifting algorithm (discussed in Section 4.3) but instead focus on other developments of the microcon-

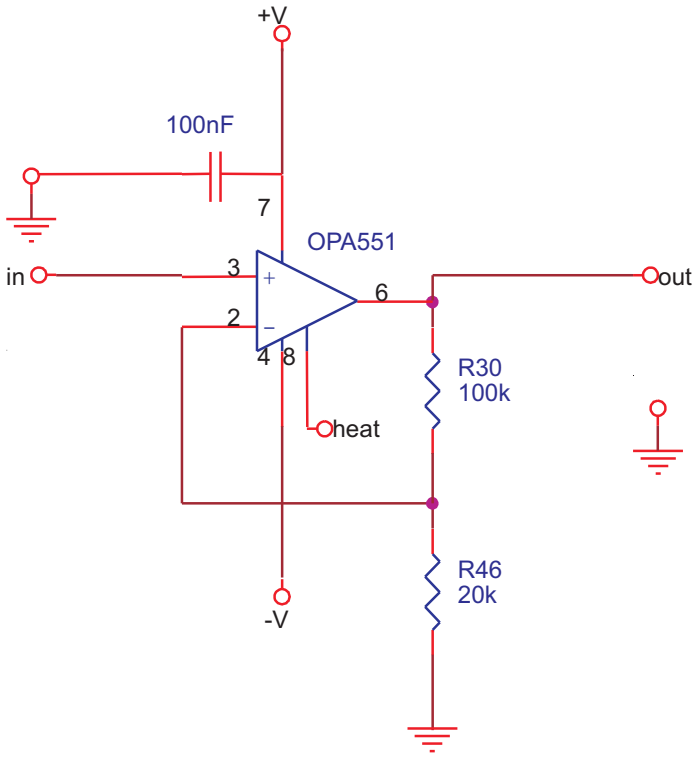

Figure 5.15: Addition of a 100 nF decoupling capacitor between V+ and ground in a opamp.

troller, such as the interface on the PC and the extensions to multiple DACs. With regard to the crosstalk measurements, this certainly need more research to give input to more detailed discussions, which was not in the scope of this research. However the modeled framework provides possibilities to both describe electrical and thermal crosstalk, so this framework can be used for research on both phenomena. Moreover, it should be measured whether indeed thermal crosstalk is proportional to V^2 , and electrical crosstalk is proportional to V. Also, the exact resistance of the heaters should be measured for better understanding of the differences in the self-values in the matrix.

5.6 Conclusions

It is shown that the controlling system designed for tuning the OBFN is able to tune 32 channels between 0 and 30 V for the Yeti chip, and 64 for the FlySmart chip. In general, 96 channels can now be tuned and this is expandable by minimal wiring and software additions. These optical chips can successfully be tuned with the control system: both stable and fast enough. The speed of updating the tuning settings (about 1 ms), is within the speed of measuring it (0.5 s).

Both measurements on the filter and the delay sections of two optical chips were shown. Also crosstalk measurements were performed and the effect of the crosstalk

was quantified using the control system. It can cope both with electrical and thermal crosstalk.

Tuning with recorded settings showed the same results in different premises, so there is stability in this sense. This also showed that the control system can be transported without getting damaged.

Oscillations in the output voltage were minimised by adding decoupling capacitors. Once actuated, the tuning is then very stable.

Chapter 6

Conclusions and directions for further research

This chapter presents the conclusions of this work and directions for further research. The conclusions are related to the research questions stated at the beginning of this Thesis (Section 1.3.3). They are presented in Section 6.1. The directions for further research are presented in Section 6.2. These are based on the current phase in which this work and the project resides. It is noted that not all suggestions made here, are fully in the direct context of this assignment.

6.1 Conclusions

In this assignment a control system has been designed and implemented to tune ring resonator-based optical beamforming networks (OBFNs). Such an OBFN forms the heart of the SMART system that was introduced in Chapter 1. The hardware and software that was integrated in this assignment, provide the means to tune different OBFNs and other optical chips by means of thermo-optical tuning. This thesis documents on the assignment to conduct this tuning. The design of the control system was discussed in Chapter 3. The implementation was discussed in Chapter 4. Different measurements and demonstration showed the correct working of the system. Measurement results were given in Chapter 5.

The software, hardware and performance requirements for the control system have been made clear. The design is elaborated and documented. It is a modular design which can easily be extended. A working prototype has been used extensively in the Telecommunication Engineering lab. It has also been used for measurements and demonstrations at the Dutch Aerospace Laboratory. The thesis also described how the system could be used, what services should be made available and how this can be achieved. This contextual part of the assignment was reported in Chapter 2.

The next part will focus on answers to the research (sub)questions stated in 1.3.3.

What would a design for a control system for OBFNs consist of?

The control system comprises a integrated piece of modular hardware components, the controller, and an interface on a PC to operate the controller. The controller consists of a motherboard with connection and power functions, a programmed microcontroller, and one or more D/A Converter boards. A bus architecture is used to stack one or more of these D/A Converter PCBs. These PCBs provide 32 channels with 14-bit resolution each. A RS232 connection, available via USB, connects this controller to the PC. A Java application is used to tune an adaptable number of channels. It is possible to load and store settings. The controller also allows for crosstalk calculations using a mathematical model, to calculate tuning settings for optical chips.

For quick tuning, it is desirable that the tuning speed is in the same order of the speed of the heaters on the optical chips. 1 ms is easily feasible with processing in the order of a MHz, event-based tuning and communication speed in the order of kilobytes per second, and D/A actuating in the order of MHz as well.

Also important is that the software on the microcontroller is easily adaptable with respect to addition or removal of commands. Also adding wiring is easily reflected in the drivers. The software on the PC can also easily be maintained as it is object-oriented, and designed using very well-known software patterns. Both packages are well documented.

The controller can be modified to increase the number of channels. Also a display can be added. This was removed in this assignment due to its slow update speed. Moreover, logging functionality was already available on the PC. Also switches can be added to the hardware. For a display and for switches, space is already available on the mother board.

To what extent is it possible to implement and evaluate a prototype for such a controlling system, and how could this system be used in an airplane?

The prototype has been used extensively for research within the Telecommunication Engineering group, and also for demonstration purposes. A redesign to cope with technical findings (see next section) should be made before it can be installed in an airplane. Also tracking functionality needs to be added. When ready, various RTCA / Eurocae tests need to be passed. Moreover, other parts of the system from the SMART project should be delivered as well.

The system may then be used to provide live broadcast services (television and radio) and, in a bidirectional scenario (with inherent more challenging infrastructural complications) internet access. This is described in Chapter 2.

6.2 Directions for further research

It is suggested to research evolutionary versions of the control system. They will be used a lot in the future and will be an important part to show results that are expected from already approved research projects. The following points can be modified to improve the system.

With respect to the controller:

- Add decoupling capacitors where needed, following best practices for PCB design in general, and the datasheet of the opamps specifically.
- Remove the FRAM and replace it with an industry standard SD Card slot. This also works with the implemented SPI protocol[1].
- Integrate a ventilator on the motherboard and reconsider the power circuitry. Then pack the controller to prevent it from damage. Make it such that components can be interchanged easily.

With respect to the software:

- Conduct research to provide the means for azimuth and angle steering.
- Remove the virtual COM port and use D2XX to communicate with the controller.
- Currently, for backward compatibility reasons a tuning range of 0 to 3000 centiVolt is used. The centiVolt measure is not needed anymore, hence it can be set to Volt (with millivolt precision) units. Moreover, the full resolution of the 14 bit DA conversion is then available.
- Research the effect of the crosstalk (electrical and thermal) and integrate the results in the tuning software. Take the effect of crosstalking rings (phase shifters and couplers) and splitters/combiners into account. The mathematical model and measurement approach and execution can be used as a starting point for this. The last point is certainly not a trivial study. Due to the inherent disadvantages of thermo-optical tuning, other means of tuning should therefore also be researched, as well as the proposition that the control system can still be used for that.

Next to these tasks, on a longer term, one should also research the integration of the hardware and software functions in a system that can be used in an airplane. This research should include a requirement analysis for airworthiness.

[1] See Section B.4

References

[1] Various writers, "Smart full project proposal," Euripides, Tech. Rep., 2005.

[2] T. Ohira, "Adaptive array antenna beamforming architectures as viewed by a microwave circuit designer," *2000 Asia-Pacific Microwave Conference*, vol. 1, pp. 828–833, 2000.

[3] H. Schippers, J. Verpoorte, P. Jorna, A. Hulzinga, A. Meijerink, C. Roeloffzen, L. Zhuang, D. Marpaung, W. van Etten, R. Heideman, A. Leinse, A. Borreman, M. Hoekman, and M. Wintels, "Broadband conformal phased array with optical beam forming for airborne satellite communication," *Aerospace Conference, 2008 IEEE*, pp. 1–17, March 2008.

[4] G. Lenz, B. Eggleton, C. Madsen, and R. Slusher, "Optical delay lines based on optical filters," *Quantum Electronics, IEEE Journal of*, vol. 37, no. 4, pp. 525–532, Apr 2001.

[5] L. Zhuang, C. Roeloffzen, R. Heideman, A. Borreman, A. Meijerink, and W. van Etten, "Single-chip optical beam forming network in LPCVD waveguide technology," in *International Topical Meeting in Microwave Photonics (MWP'2006), Grenoble, France*, 3–6 Oct. 2006, p. F1.4.

[6] C. Roeloffzen, "Passband flattened binary-tree structured add-drop multiplexers using sion waveguide technology," Ph.D. dissertation, University of Twente, 2002.

[7] A. Meijerink, C. Roeloffzen, L. Zhuang, D. Marpaung, R. Heideman, A. Borreman, and W. van Etten, "Phased array antenna steering using a ring resonator-based optical beam forming network," in *13th IEEE/CVT Symposium Benelux, Liège, Belgium*, 23 Nov. 2006.

[8] C. Adler, A. Monk, D. Rasmussen, and M. Taylor, "Two-way airborne broadband communications using phased array antennas," *Aerospace Conference, 2003. Proceedings. 2003 IEEE*, vol. 2, p. 5, 2003.

[9] A. D. Monk and C. Adler, "Calibration and rf test of connexion by boeing airborne phased arrays," *IEEE International Symposium on Phased Array Systems and Technology*, vol. 405-410, p. 10, 2003.

[10] E. J. Borgstrom, "EMC requirements for avionics: RTCA/DO-160e," *Interference Technology*, vol. 1, pp. 1–7, 2004.

[11] D. Mansour. The airborne broadband dream - is it just another bubble? online. Starling Advanced Communications. [Online]. Available: http://www.starling-com.com/imgs/uploads/white%20paper%202002.pdf

[12] W. D. Shoaff, "How to write a master's thesis in computer science," *Department of Computer Sciences, Florida Institute of Technology*, vol. 1, pp. 1–10, 2001.

[13] A. Jahn, M. Holzbock, J. Muller, R. Kebel, M. de Sanctis, A. Rogoyski, E. Trachtman, O. Franzrahe, M. Werner, and F. Hu, "Evolution of aeronautical communications for personal and multimedia services," *Communications Magazine, IEEE*, vol. 41, no. 7, pp. 36–43, July 2003.

[14] (2008) Anastasia. Project Website. [Online]. Available: http://www.anastasia-fp6.org/

[15] *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services*, European Telecommunications Standards Institute (ETSI) Std. EN 300 421.

[16] U. Reimers, *DVB - The family of international standards for Digital Video Broadcasting*, 2nd ed. Springer, 2005.

[17] S. L. Kota, "Quality of service for broadband satellite internet - ATM and IP services," Ph.D. dissertation, University of Oulu, 2003.

[18] M. Ruiter, "Design of a system for driving heaters on optical ring resonators," 2006.

[19] T. Vrijmoeth, "Implementation of a heater-driving system," 2006.

[20] T. Jansen, "Implementing a heater controller for optical beam forming networks," 2007.

[21] J. W. van 't Klooster, C. Roeloffzen, A. Meijerink, L. Zhuang, D. Marpaung, W. van Etten, R. G. Heideman, A. Leinse, H. Schippers, J. Verpoorte, and M. Wintels, "Design of a ring resonator-based optical beam forming network for phased

array receive antennas," in *Proceedings of the 30th ESA Antenna Workshop on Antennas for Earth Observation, Science, Telecommunication and Navigation Space Missions*, 2008.

[22] Future Technology Devices International Ltd. [Online]. Available: http://www.ftdichip.com

[23] L. Zhuang, C. G. H. Roeloffzen, R. G. Heideman, A. Borreman, A. Meijerink, and W. van Etten, "Single-chip ring resonator-based 1x8 optical beam forming network in CMOS-compatible waveguide technology," *IEEE Photonics Technology Letters*, vol. 19, no. 8, pp. 1130–1132, Aug. 2007.

[24] L. Zhuang, A. Meijerink, C. G. H. Roeloffzen, D. A. I. Marpaung, R. G. Heideman, M. Hoekman, A. Leinse, and W. van Etten, "Novel ring resonator-based optical beamformer for broadband phased array receive antennas," *Proceedings of the LEOS Annual Meeting*, vol. MB, p. 3, 2008.

[25] (2007) SMart Antenna systems for Radio Transceivers (SMART). Telecommunication Engineering. [Online]. Available: http://www.el.utwente.nl/te/research/SMART/SMART.htm

Appendix A

Controller Commands

This Appendix lists the commands which can be sent to the controller. The implementation of the software runned on the controller itself is discussed in Chapter 4.

- **wrch nn,xxxx** "Write Channel." Writes the channel nn with centiVolt[1] value xxxx Example: wrch 3,456 writes 4,56V to channel 3.
- **SNHnn** "Set Number Heaters." Sets the number of heaters (the value represented by nn) to drive, in the FRAM. Also defines the dimensions of vectors and matrix in the algorithm.
- **GNH** "Get Number Heaters." Gets the number of heaters to drive from the FRAM and displays it.
- **SOHnn,ppp.ppp** "Set Offset Heater." Sets the offset of the heater represented by nn with the value represented by ppp.ppp in the FRAM.
- **GOHnn** "Get Offset Heater." Gets the offset of the heater represented by nn from the FRAM and displays it.
- **SOLppp.ppp** "Set Offset Laser." Sets the offset of the laser with the value represented by ppp.ppp in the FRAM.
- **GCO** "Get Complete Offsets." Gets all the offset of the heater from the FRAM and displays it on the PC.
- **GOL** "Get Offset Laser." Gets the offset of the heater represented by nn from the FRAM and displays it.
- **SMNxx,yy,pp.pppp** "Set Matrix Number." Sets the value of matrix entry (xx,yy) with the value represented by pp.pppp to the FRAM.
- **GMNxx,yy** "Get Matrix Number." Gets the value of matrix number (xx,yy) from the FRAM and displays it.
- **GCM** "Get Complete Matrix." Gets the matrix content from the FRAM and displays it on the PC.
- **SAHnn,ppp.ppp** "Set Angle Heater." Sets the desired angle of the heater rep-

[1]The centiVolt measure is used for backwards compability with the old IBM boxes and LabView programs used in the laboratory.

resented by the number nn to the value represented by ppp.ppp. This value is being saved in the RAM of the microprocessor and in the FRAM as well.

- **DAnn** "Display Angle." Displays the angle of the heater represented by the number nn.
- **GCA** "Get Complete Angles." Gets all set angles from the flash memory And displays it on the PC.
- **DVnn** "Display Voltage." Displays the voltage of the heater represented by the number nn.
- **CV** "Calculate Voltages." Calculates the voltages to output as a function of the desired phases.
- **CE** "Check Execution." Control method to check the execution of the method functionality.

Appendix B

SPI Protocol

Thix Appendix gives some background information on the Serial Perhiperal Interface (SPI) protocol. This protocol is used in the controller for both the non-volatile FRAM memory, and the DAC chip. Both are hardware components that communicate with the microprocessor of the controller. This Appendix dicusses the details of the protocol this communication relies on. The wiring and the different configurations possibilities are discussed. Then the implementation regarding SPI in the controller is discussed. Finally, a suggestion for future implementations is given.

B.1 SPI overview

Serial Perhiperal Interface (SPI) is a data link named by Motorola. It is a synchronous link, which can operate in full duplex mode. Hence the behaviour of the communicating partners, either a Master or a Slave, is synchronized, and they may both send and receive at the same time. For a single Slave, the SPI protocol uses four wires: a clock line (SCLK), a chip select line (CS), a data output line and a data input line. The latter two are named as seen from the Master side, hence data out is also called Master Out Slave In (MoSi), and data in is also named Master In Slave Out (MiSo). In SPI, also more Slave devices can be utilized, with either individual addressing lines connected to the Master (the Chip Select (CS) line), or a daisy chain, as explained later on. The wiring for the connection between a Master and a Slave is shown in Figure B.1.

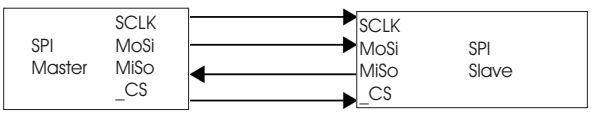

Figure B.1: SPI configuration with a single Slave.

B.1.1 Transmission: chip select

As seen in Figure B.1, the CS has a bar below the 'C'; it is an inverted wire in the sense that its logic value is low or zero when it is active, in contrary to the other data lines (MoSi and MiSo). Therefore it is normally mentioned _CS. When the Master initiates a transmission, it therefore firstly pulls the _CS low to inform the Slave that a transmission will begin. When a transmission ends, the Master pulls the _CS high again. This is shown in Figure B.2, which is the same as Figure 5.2. After the _CS is brought low, the SCLK starts alternating and the data on the MoSi line is interpreted by the Slave.

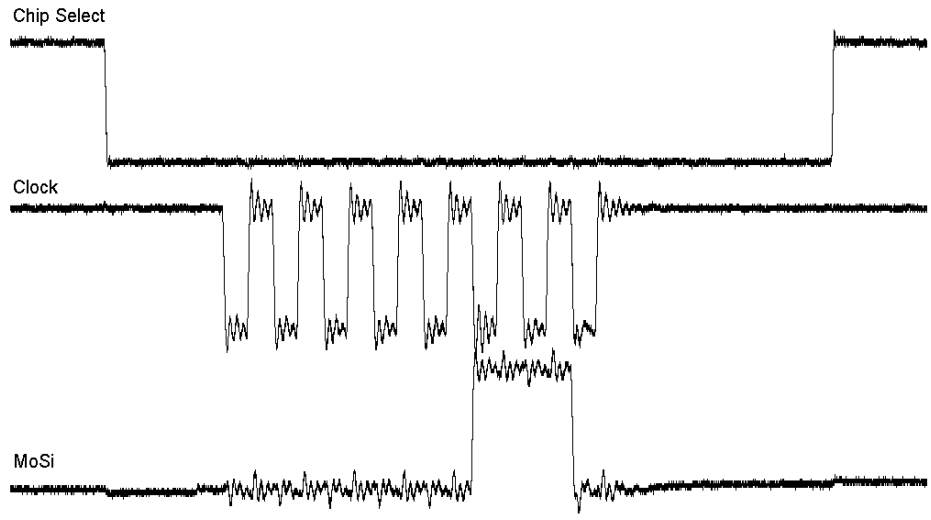

Figure B.2: SPI communication measurement between microcontroller and FRAM. Above: _CS signal. Middle: clock signal. Below: MoSi. After the _CS is brought low, the SCLK starts and the value on the MoSi line is interpreted by the Slave.

B.2 Modes

Regarding the clock wire, a sawtooth pattern can be identified to dictate the speed of the transmission to the Slave. There are 4 modes in which Slaves may operate with respect to this clock. This has to do with the base polarity value of the clock (CPOL), and its falling or rising edge when the clock value goes from 0 to 1 or from 1 to 0 (CPHA). Both CPOL and CPHA have two modes, so in total there are 4 different modes of operation for the SPI protocol:

- Mode 0: CPOL=0, CPHA=0. In mode 0, the base value of the clock is zero. Data is read on the rising edge of the clock (transition from 0 to 1) and data is

changed on a falling edge.

- Mode 1: CPOL=0, CPHA=1. In mode 1, the base value of the clock is zero. Data is read on the falling edge of the clock and data is changed on a rising edge.
- Mode 2: CPOL=1, CPHA=0. In mode 2, the base value of the clock is one. Data is read on the falling edge of the clock and data is changed on a rising edge.
- Mode 3: CPOL=1, CPHA=1. In mode 3, the base value of the clock is one. Data is read on the rising edge of the clock and data is changed on a falling edge.

The 4 modes are summarized in Table B.1.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Table B.1: Modes of the SPI protocol

B.2.1 Different modes on the controller

Since the FRAM and DAC operate in different modes, it was chosen to adapt the driver such that each time the FRAM is addressed, the mode changed to FRAM-compatibility. Whenever the FRAM-related operation is finished, the driver changes back to the DAC-compatible mode. This was done to be able to use both components, with the minimal delay for the DAC operations since they need to be executed as fast as possible, whereas a small delay for the FRAM-related operations is not a very big issue.

B.3 Different chip select lines or daisy chain

When more than one Slave is used, such as in this project, one can either use multiple _CS lines to select a single or multiple Slaves, or employ a daisy chain configuration. Both configurations are shown below in a configuration with three Slaves. The connection with three different select lines is shown in Figure B.3, and the daisy chain configuration is shown in Figure B.4. The advantage of the usage of multiple _CS lines is the possibility to do both singlecasting and multicasting or broadcasting. Also, it is better upgradeable since only a single wire should be added for each new slave. The cost of course is that the number of wires may get to high when many Slaves are needed. In a daisy chain configuration, the Slaves behave as a shift register and therefore it takes more time for the propagation of a command to the last Slave in the chain. It is

also more difficult to address just a single Slave in the daisy chain, since there are no individual select lines.

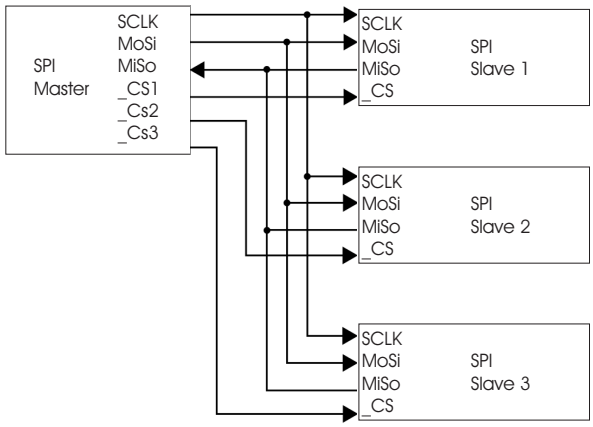

Figure B.3: SPI configuration with 3 Slaves selectable by their corresponding _CS lines.

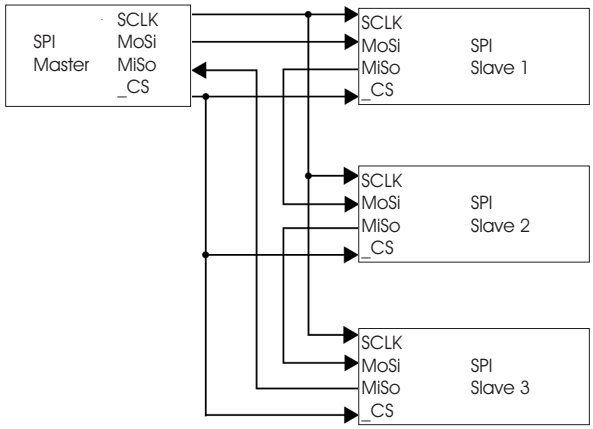

Figure B.4: SPI configuration with 3 Slaves in a daisy chain configuration.

B.3.1 SPI adjustments for three DAC PCBs

In this project both the hardware and the software of the controller have been adjusted to be able to 3 DAC PCBs, or 96 channels. It was chosen to use two extra _CS lines because of the advantages given in the previous Section. Since the other lines are fed through a bus structure to the DAC PCBs, this drew only the need for two extra wires on the mother board, and a connection from the bus structure to the _CS connection of the DAC on the individual PCBs. Hence now these PCBs are not anymore identical to each other (whereas they used to be before the extension).

The code used for this adjustment is given in Listing B.1 and B.2. It consist of statements to (de)select a DAC PCB by alternating its _CS value (Listing B.1), and,

once setting an output channel to a voltage value, some logic to test the channel number and select the according DAC PCB (Listing B.2).

Listing B.1: SPI code for initialising and switching the CS lines for the DAC PCBs

```
// Initialyzes first CS for DAC PCB1
void Initialize_CS2(void)
{
  PINSEL1 &= ~(0x00FFF000); // Initialyzes P0.22 as an output pin
  IO0SET = 1 << 22;         // make pin an active output
  IO0DIR |= 1 << 22;        // Pin 0.22 defined as output
  Initialize_CS2_23();      // also initialize 23 as a GPIO CS
  Initialize_CS2_27();      // also initialize 27 as a GPIO CS
}

// Initialyzes P0.27 (=CS for board 2) as an output pin
void Initialize_CS2_27(void)
{
  PINSEL1 &= ~(0x00FFF000); // Initialyzes GPIO-output pins
  IO0SET = 1 << 27;         // make pin an active output
  IO0DIR |= 1 << 27;        // Pin 0.27 defined as output
}

// Initialyzes P0.23 (=CS for board 3) as a GPIO-output pin
void Initialize_CS2_23(void)
{
  PINSEL1 &= ~(0x00FFF000); // Initialyzes P0.22 and P0.23
  IO0SET = 1 << 23;         // make pin an active output
  IO0DIR |= 1 << 23;        // Pin 0.23 defined as output
}

// function to invert the a CS-value 0 ¿ 1 or 1 ¿ 0 (idem for other CS)
void Pin_CS2_27(unsigned int level)
{ if (level) {
    IO0SET = 1 << 27;       //cs op pin 0.27
  }
  else IO0CLR = 1 << 27;
}
```

Listing B.2: SPI code for selecting the DAC PCB based on the channel number.

```
//Write the given voltage value for the given channel number to DAC 1, 2 or 3
void DAC_spi_write(unsigned int adr, unsigned int value) {
  if (adr < 32) {                          //channel 0..31
    Pin_CS2(0);                            //Select DAC-chip 1
    SPI1_Byte(adr & 0x1F);                 //Write address
    SPI1_Byte((value | 0xC000) >> 8);      // Write value
    SPI1_Byte(value & 0xFF);
    Pin_CS2(1);
  }
  else if (adr > 31 && adr < 64 ) {        //channel 32..63
    Pin_CS2_27(0);                         //Select DAC-chip 2
    SPI1_Byte((adr-32) & 0x1F);            //Write address
    SPI1_Byte((value | 0xC000) >> 8);      // Write value
    SPI1_Byte(value & 0xFF);
    Pin_CS2_27(1);
  }
  else if (adr > 63 && adr < 96 ) {        //channel 64..95
    Pin_CS2_23(0);                         //Select DAC-chip 3
    SPI1_Byte((adr-32-32) & 0x1F);         //Write address
    SPI1_Byte((value | 0xC000) >> 8);      // Write value
    SPI1_Byte(value & 0xFF);
    Pin_CS2_23(1);
  }
}
```

B.4 Suggestion with regard to SPI

Since SPI is a well spread de facto standard for memory extension nowadays, it is suggested to replace the FRAM with for example a Secure Digital (SD) memory slot in future versions of this controller. SD is also SPI-compatible, and its advantages are better market adoption, a better price-quality tradeoff (more storage space for less money), and the possibility to take out the SD card to read or fill log files or other information on a computer.

Appendix C

Code Package

This Appendix shows a UML diagram of the code package developed for the controller in Figure C.1 and describes the files of the code package[1].

C.1 UML diagram

The diagram is complete, in the sense that it shows all C files of the software, except from the libraries and header files being used. Though not specifically for C code in the first place, the UML diagram below shows at least the separation of functionalities and stresses the central role of the main file and the ASI_Exec_handler file, in which the handling of the received commands takes place.

C.2 Files

Below a listing is given of the files that compose the software, followed by the main responsibilities of the files.

- SPI_Mod.c. This files handles the SPI communication between computer and controller.
- System_Functions.c. This files implements various methods to do type conversions.
- Lookup.c. This file is a virtual layer for the data entries in a lookup table. It lists all methods to address these values stored in the FRAM.
- Utils_Mod.c. This file contains utility functions for the controlling all hardware components of the controller.
- Asi_Mod.c. This file respresents the protocol put on top of the SPI protocol. In this file the high level communication between the computer and the controller is conducted.

[1] To obtain a copy of the code package, contact the TE group [25].

- LCD_Mod.c. In this file, the LCD functionality is written.
- main.c. This is the initialisation and main loop of the control cyclus. For compactness, it calls the next file for the control cyclus.
- Asi_Exc_Handler.c. This is the control cyclus for the main loop. It also manages the calculations done by Calculate.c file.
- Calculate.c. In this file the calculations and actuations for the mathematical function are implemented.

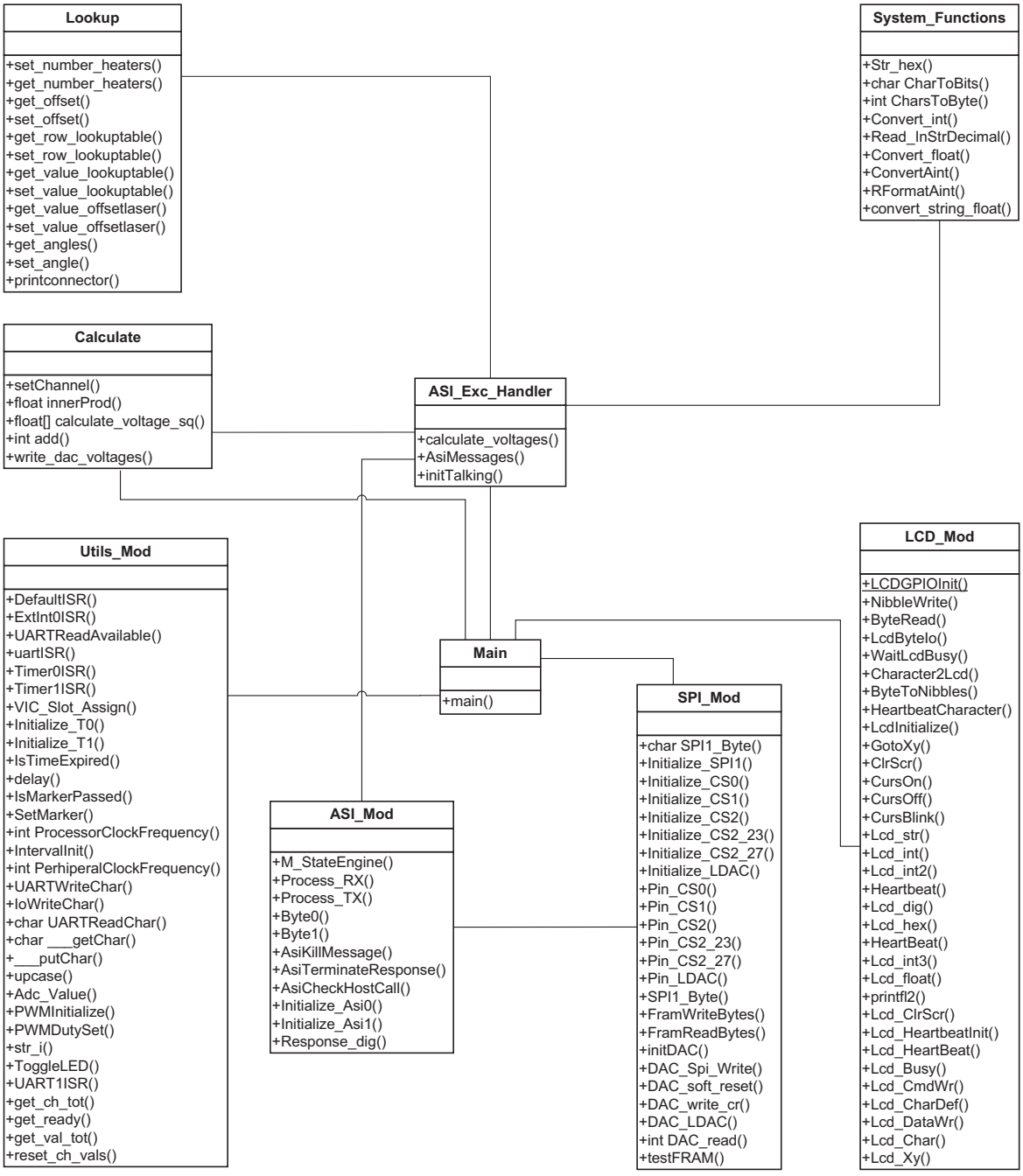

Figure C.1: UML diagram for the source code package for the controller.

Appendix D

Channel Numbering on FlySmart chip

In Subsection 5.4.3, all rings of the OBFN of the FlySmart chip were labeled. In this Appendix, the related Figure is repeated (Figure D.1) and all channel numbers on the controller for each ring are listed.

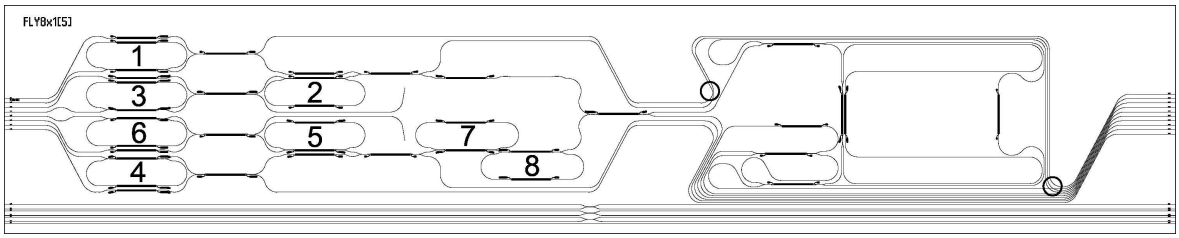

Figure D.1: Numbering of the OBFN ORRs for the crosstalk measurements The thin black lines are the waveguides. The short, thicker black lines are the resistor used for the tuning. The two circles on the right indicate waveguide crossings. The waveguides on the left (input) are numbered 8 (top) to 1 (bottom), on the right there is the output, 1 (top) to 8 (bottom).

Though difficult to see, since the waveguide are very thin in real, the waveguides on the left (input) are numbered 8 (top) to 1 (bottom), on the right there is the output, 1 (top) to 8 (bottom). Normally all 8 eight inputs would be guided to 1 of these outputs in case of a receive scenario (8×1 beamforming). All rings were measured at waveguide output 6, since it is the output to which all input waveguide can be coupled. The input had to be chosen according to a path that passed the to-be-tuned rings.

A 32 channel controller was used for the crosstalk measurements from Chapter 5, hence the measurements are obtained using controller with one DAC PCB. This was sufficient for these measurements, since not all of the 48 heater pins of the Yeti chip were connected.

D.1 Ring, channel and waveguide data

Below an enumeration is given of the voltage sign per ring, their channels for ϕ and κ tuning, and the used input waveguide.

1. +, ϕ channel 1, κ channel 2, input waveguide 7.
2. -, ϕ channel 15, κ channel 16, input waveguide 5. Normally, input waveguide 6 would have been used but this waveguide was damaged.
3. +, ϕ channel 3, κ channel 4, input waveguide 5.
4. +, ϕ channel 7, κ channel 8, input waveguide 2.
5. -, ϕ channel 17, κ channel 18, input waveguide 3.
6. +, ϕ channel 5, κ channel 6, input waveguide 4.
7. -, ϕ channel 21, κ channel 22, input waveguide 2.
8. -, ϕ channel 23, κ channel 24, input waveguide 2.