NTNU
Norwegian University of
Science and Technology

Applying a Digital Verification Approach to an Analog Protocol

Further development of a reusable verification component based on the Universal Verification Methodology

Christoffer Ramstad-Evensen

Master of Science in Electronics
Submission date: June 2016
Supervisor: Kjetil Svarstad, IET
Co-supervisor: Christoffer Amlo, Nordic Semiconductor ASA

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Candidate Name: Christoffer Ramstad-Evensen
Assignment Title: Applying a Digital Verification Approach to an Analog Protocol

Improve the implementation of a Universal Verification Component (UVC) for the NFC-A technology of the Near Field Communication (NFC) protocol. Extend the functionality and verify the requested behavior of the UVC. The UVC shall include models for simulating analog behavior using the SystemVerilog real data type. A minimum stimulus library should be provided with the UVC.

Supervisor: Kjetil Svarstad
Supervisor: Christoffer Amlo

Abstract

Analog and mixed-signal circuit designs are more important now than ever, due to the popularity of wearable and wireless electronic devices. The forecast of the Internet of Things (IoT) suggest that the need for advanced Mixed-Signal System-on-Chips (MSSoC) will be present in many years to come. The increasing functionality required by the analog parts of MSSoCs presents a great challenge in the verification of such systems. Simulation Program with Integrated Circuit Emphasis (SPICE) tools are still the main technique used to verify analog circuits. However, the speed of SPICE simulation becomes an issue with the increasing complexity of analog designs. Other verification strategies are necessary in order to reach sufficient functional verification within the time to market. Real Value Modeling (RVM) is a technique that can be used for digital simulation of analog circuit, yielding a greater speed performance than SPICE tools. By exploiting RVM in combination with the de facto standard for digital verification, the Universal Verification Methodology (UVM), sufficient functional coverage can be achieved.

This thesis presents the design and implementation of a UVM based Universal Verification Component (UVC) for the Near Field Communication (NFC) protocol. The final UVC enables protocol verification of NFC devices using the NFC-A technology. By leveraging Metric Driven Verification (MDV) and RVM models, the UVC achieves digital simulation speed in functional verification. A mathematical model for driving stimulus on an analog interface is presented as well as a model for estimating the frequency of the observed interface. The models enable driving and monitoring of analog characteristics such as amplitude, modulation and frequency.

Verification was performed by the UVC in a testbench with a Device Under Test (DUT) using the NFC-A technology. 100% functional coverage was reached at digital simulation speeds, according to a verification plan. Visual inspection of the driven and monitored interface confirms the UVCs ability to perform correct protocol verification. The results indicate that the UVC can mitigate the speed performance of SPICE simulation to improve functional verification of analog modules.

Sammendrag

Analog og blandet krestdesign er viktigere enn noen gang på grunn av populariteten rundt bærbar og trådløs elekronikk. Prognosen for tingenes internett (IoT) antyder et økende behov for avanserte systemkretser basert på blandet kretsdesign (MSSoC) i mange år fremover. Behovet for økt funksjonalitet i analoge deler av MSSoC-systemer har ført til utfordinger relatert til verifikasjon av slike system. Simuleringsprogram med fokus på integrerte kretser (SPICE) er fremdeles den viktigste måten for å verifisere analog kretser, men i lys av økt kompleksitet i analoge krester kommer simuleringshasigheten til SPICE-verktøy til kort. Andre metoder for å oppnå tilstrekkelig funksjonell verifikasjon er nødvendig for å nå markedet i tide. Ekte verdi modellering (RVM) er en teknikk som kan brukes i digital simulering av analoge kretser for å øke simuleringshastigheten i forhold til SPICE-simuleringer. Ved å utnytte RVM i kombinasjon med den mest brukte metoden for digital verifikasjon, universell verifikasjonsmetodologi (UVM), kan tilstrekkelig funksjonell verifikasjon oppnås.

Denne oppgaven presenterer en UVM basert universell verifikasjonskomponent (UVC) designet og implementert for nærfeltskommunikasjons (NFC) protokollen. Den ferdige UVC′en muliggjør protokollverifikasjon av NFC enheter som dekker NFC-A teknologien. Ved å utnytte målstyrt verifikasjon (MDV) og RVM-modeller oppnår UVC′en digital simuleringshastighet ved funksjonell verifikasjon. En matematisk modell for å påtrykke stimuli på et analogt grensesnitt er presentert i tillegg til en modell for å estimere frekvensen observert på grensesnittet. Modellene muliggjør påtrykking og overvåking av analoge karakteristikker som amplitude, modulering of frekvens.

UVC′en er demonstrert ved å utføre verifikasjon i en testbenk satt opp med en enhet under test (DUT) basert på NFC-A teknologien. 100% funksjonell verifikasjon oppnås ved digital simuleringshastighet, i henhold til en verifikasjonsplan. Visuell inspeksjon av påtrykte og målte verdier på grensesnittet bekrefter at UVC′en verifiserer protokollen riktig. Resultatene indikerer at UVC′en kan gi økt hastighet ved funksjonellverifikasjon av analoge moduler ved å tilføyes SPICE-simuleringer.

Preface

Throughout the work of my thesis I have become well aware of the importance as well as the challenges related to verification in the semiconductor industry. I have had the pleasure of working with the interesting challenges concerning verification of mixed-signal designs and one of the leading verification methodologies for digital simulation, UVM.

I would like to thank Kjetil Svarstad, my NTNU supervisor, for excellent guidance and feedback throughout the work on my thesis. Many thanks goes to Christoffer Amlo, my Nordic Semiconductor ASA supervisor, for being available at all times giving me consultation on verification methodologies and technical aspects of the thesis.

I would also like to thank Ralf Hekmann for being a "go-to" specialist in NFC and providing valuable review of the thesis. Thanks to Vinodh Ravinath for teaching me the concepts of UVM and Yngve Skogseide for great support in creating a working simulation environment.

Contents

List of Figures

List of Tables

List of Abbreviations

AHB	**A**dvanced **H**igh-performance **B**us
AMS	**A**nalog **M**ixed-**S**ignal
ASK	**A**mplitude-**S**hift **K**eying
CDV	**C**overage **D**riven **V**erification
CRV	**C**onstrained **R**andom **V**erification
DUT	**D**evice **U**nder **T**est
EoF	**E**nd-**o**f **F**rame
FDT	**F**rame **D**elay **T**ime
FSM	**F**inite-**S**tate **M**achine
IC	**I**ntegrated **C**ircuit
IoT	**I**nternet **o**f **T**hings
IP	**I**ntelectual **P**roperty
MDV	**M**etric **D**riven **V**erification
MSSoC	**M**ixed-**S**ignal **S**ystem-**o**n-**C**hip
NFC	**N**ear **F**ield **C**ommunication
NFC-A	**NFC** technology type **A**
OOK	**O**n-**O**ff **K**eying
OOP	**O**bject **O**riented **P**rogramming
OSI	**O**pen **S**ystems **I**nterconnection
OVM	**O**pen **V**erificaion **M**ethodology
RF	**R**adio **F**requency
RVM	**R**eal **V**alue **M**odel
SoC	**S**ystem-**o**n-**C**hip
SoF	**S**tart-**o**f **F**rame
SPICE	**S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis
TLM	**T**ransaction-**L**evel **M**odeling
UVC	**U**niversal **V**erification **C**omponent
UVM	**U**niversal **V**erification **M**ethodology
VLSI	**V**ery-**L**arge-**S**cale **I**ntegration

Chapter 1

Introduction

The trend in increased applications for consumer electronics, as well as commercial electronics, does not seem to reach an end anytime soon. Wearable electronics are becoming popular and has a range of applications such as health monitoring, navigation, media, communication and more applications are probably right around the corner. So it seems when hearing about the next big thing of tomorrow, the Internet of Things (IoT). Imagine a world where any physical device could be accessed from anywhere through the internet. Search missing keys on the internet and their location will be given in seconds. Patients with heart disease could have an embedded chip inside the chest that monitors the heart. Information about the patient's heart can be sent to computers at a hospital running powerful algorithms, potentially predicting a stroke and save lives. This is some of the forecast potential of the IoT and it is predicted that within 2020, 50 billion devices will be connected in the internet [1, p. 3].

For the semiconductor business, all these wearables and physical devices connected to the internet, means a high demand for wireless communication. Hence, the need for Mixed-Signal Systems-on-Ship (MSSoC) increase as well as the complexity of these chips, due to the large range of possible applications. The challenge of such systems is that verification becomes a real issue and with the time-to-market requirement it is nearly impossible to perform all the necessary verification in time. According to [2], 70% of the logic design phase of a chip is devoted to verification and 60% of SoC re-spins are due to mixed-signal errors. The increasing functionality of SoCs have been an issue for quite some time. Hence, methodologies for improving the verification process of Integrated Circuits (IC), such as the Universal Verification Methodology (UVM), have appeared. However, UVM is used for verification of digital parts of the chip and analog parts are verified by Simulation Program with Integrated Circuit Emphasis (SPICE) tools. SPICE tools yield accurate simulation of the analog circuits, but they are time consuming. With the increasing complexity of the analog parts of the SoCs, performance becomes an issue, and other approaches are needed. Real Value Modeling (RVM) is a technique based on discrete models of analog parts of a design. These models can be used in digital simulation, yielding a speed-up in the functional verification of analog designs.

In this thesis UVM and RVM is combined to mitigate the poor speed of functional verification in SPICE simulations.

1.1 Main Contributions

A Universal Verification Component (UVC) for the NFC-A technology of the Near Field Communication (NFC) protocol has been designed and tested. The UVC is based on the verification methodology UVM and is designed to be reusable and easily configurable for different testbenches. Due to its modular design, several of the components comprising the UVC is reusable for other UVC designs.

A mathematical model has been designed for estimating the period of a sinusoidal signal. For interfaces similar to the NFC interface, the model can be used to estimate the period of the interface down to a pico second in a simulation with a nano second timescale. Thus, enhancing simulations speed.

The UVC comes with a UVM based stimulus library for NFC-A requests. The library is intended for users of the UVC to easily create interesting stimulus for different testbenches. Its modular structure enables fast creation of new stimulus and it could work as an example for writing stimulus libraries for other UVC designs.

1.2 Methodology

The final design is based on evaluating the ease extending functionality, reusability and meeting requirements of the verification methodology and the protocol. The design consists of modular components and the evaluation is performed separately for each component and for the design as a whole.

Testing is done according to a specified verification plan and targets the functional verification of the protocol. The testbench use Metric Driver Verification (MDV) to quantitatively measure the end of test and to aim the stimulus towards a specific part of the stimulus space. The results are analyzed in terms of the initial requirements of the design, according to the verification plan and by visual inspection.

1.3 Outline of the Report

First a presentation of the verification methodology, UVM, and the analog protocol, NFC, is given in **Chapter 2**. How to set up a verification plan for functional verification is presented as well as a brief study of relevant work. **Chapter 3** describes the result of the design phase of the thesis and arguments for the chosen design. Detailed description of the components comprising the implementation is given in **Chapter 4**. To demonstrate the final implemented design, a testbench is set up to simulate a realistic scenario, which is presented in **Chapter 5**. The results of the testbench is presented in **Chapter 6**. Based on the arguments during the design phase and results from the testbench, a discussion is presented in **Chapter 7** to evaluate the design. **Chapter 8** concludes the thesis, sums up the most important evaluations and presents recommendations for future work.

Chapter 2

Background and Related Work

This chapter is meant to present the necessary information on the central topics and relevant work leading up to the conclusion of this thesis. UVM and the NFC protocol will be covered as well as section about verification planning. The UVM and NFC sections are included from a project report the author did in an earlier study [3], however, the sections are extended and customized for this thesis. A final sections presents a brief study of relevant literature.

2.1 Universal Verification Methodology (UVM)

UVM seems to have taken over as the standard in functional verification of digital designs. It is the successor of other methodologies such as Open Verification Methodology (OVM) and is currently at version 1.2. UVM consist of the industry's best practice in functional verification of digital design [4, p. 1] for over a decade and provides a solid framework for setting up a verification environment. Very importantly UVM provides MDV which means that constrained random stimulus is generated and functional coverage is performed in a self-checking environment [4, p. 2]. This is also known as the "three C's" - Coverage, Checkers and Constraints and is essential in UVM. With its fixed coding style and use of Transaction-Level Modeling (TLM) UVM enables high reusability through what is called the UVC. As System-on-chips (SoC) become more complex, so does the verification of the SoCs. By leveraging these techniques UVM might contribute in making the verification effort less time consuming.

This chapter will describe the important components that makes up the UVC [4, p. 13-15] and how to set up a UVM testbench. For more details on UVM and how create a UVM testbench please refer to [5].

2.1.1 Test Bench

Figure 2.1 shows the overview of an example UVM testbench created according to the recommended guidelines presented by the UVM User's Guide [5]. The testbench typically encapsulates the running tests, instantiates the Device Under Test (DUT), the interface and the connections between them. The tests are dynamically instantiated at run-time which allows the testbench to be compiled once and run different tests.

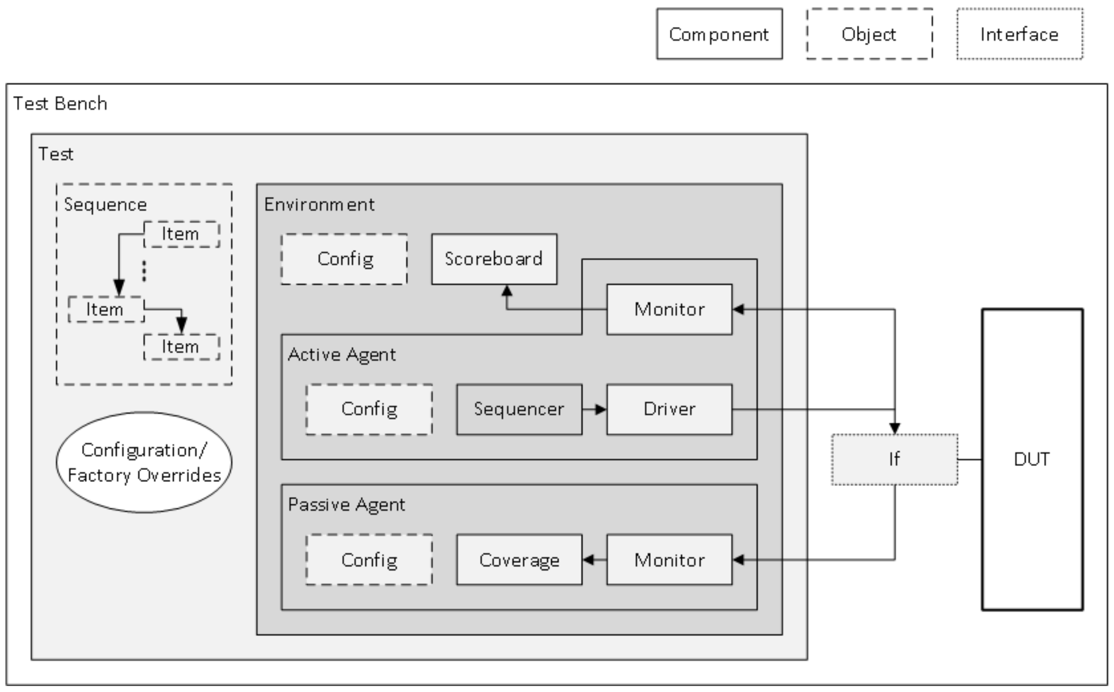


FIGURE 2.1: Typical UVM testbench architecture.

The following chapters will elaborate on the recommended guidelines for setting up the components and objects inside the testbench illustrated in Figure 2.1.

2.1.2 Test

The test is the top-level UVM component of the testbench and is responsible for creating the environment and test stimuli, based on sequences, and passes the test stimuli down the component hierarchy. The test configures the environment by setting configurations in the configuration database.

2.1.3 Configuration Database

The configuration database is a type-specific mechanism that enables the verification engineer to store parameterized configuration information that can be access from anywhere in the component hierarchy. This allows the designer to specify a configuration object in the test component and by storing the object in the configuration database it can be access in the sub components in order to set up the components with the desired configuration. In the example testbench, Figure 2.1, the environment and agents are configured through this mechanism. The configuration object for the environment would contain fields indicating whether to include a scoreboard, the number of different agents and/or other environments. It usually would contain the configuration objects for the agents if changes to these objects are made in the environment. The agent configuration objects would contain parameters for specifying whether the agents are active or passive or if it should have a coverage component.

Other specific parameters used in the configuration are put in the respective configuration objects.

2.1.4 Environment

The environment holds the agents, the scoreboard, coverage collectors, other components, other environments and configuration objects that are interrelated in order to exercise the DUT. The environment reads the configuration from the configuration database and connects all the sub components together accordingly.

2.1.5 Agent

The agent encapsulates the driver, sequencer, monitor, coverage collector and a configuration object that configures the components for the particular verification environment. There is generally one agent per interface making the agent a hierarchical component that groups the verification component dealing with a specific interface to the DUT.

According to UVM the agent has to be configurable as an active or a passive component. The difference is that an active agent is instantiated with components that enables stimulus generation. A passive agent is only configured with the ability to monitor the interface.

2.1.6 Sequencer

Execution of sequences is performed by a sequencer. Upon a driver request for a sequence item, the sequencer grants the request with a randomized sequence item by executing a sequence. The randomization happens after the item has been granted and happens within the constraints of the sequence.

2.1.7 Sequence

When generating stimulus for the DUT it should be random, but it should also be a combination of relevant stimuli. The sequence makes it possible to put together a meaningful sequence of sequence items. The items of this sequence are randomized by executing the sequence. An example of such a sequence could be a burst transfer on an AMBA High-performance Bus (AHB) bus. The sequence would specify the general structure of the transfer. When executed, the sequence would be randomized within a certain set of constraints.

2.1.8 Sequence Item

UVM enables a higher abstraction above the signal representation in a traditional verification environment through TLM. TLM means that the signal representation is replaced with the notion of transactions. An example of such an abstraction is an Ethernet frame consisting of different signals. The sequence item is the fundamental object of this abstraction level in UVM and represents the information passed between components inside the environment. The sequence item holds direct information about the signal interface of the DUT as well as any additional attributes considered to increase the abstraction inside the verification environment. One of the greatest features about the sequence item is the possibility generating random stimulus by randomizing the fields inside the item. One other great feature is the ability to put constraints on the randomization of the fields inside the item. The sequence item is also a container for functions for printing, coping and comparing of sequence items.

2.1.9 Driver

The driver is the component which is responsible of driving the signals to the DUT. This task includes requesting sequence items from the sequencer before translating the items to interface signals. In addition, the driver takes care of the timing in when to proceed with the next item of the sequence.

2.1.10 Monitor

The monitor is similar to the driver in the way that is handles translation between items and signals on the interface. The monitor decides when to create sequence items from the signals on the interface. The items are broadcast to components such as scoreboards, coverage collectors and checkers. The monitor is a fundamental part of the self-checking mechanisms of UVM.

2.1.11 Scoreboard

The scoreboard's main responsibility is to check the behavior of the DUT. The scoreboard receives the input and the output of the DUT through the analysis port of an agent and runs the input through a model to produce the expected output and checks if the actual output matches the predicted output. In essence, the scoreboard takes two inputs and checks if they match according to user defined criteria.

2.1.12 Coverage

Coverage is an essential part of the MDV in UVM. Coverage is used to specify a metric for how well the design is tested. The coverage points are often put inside the monitor or in a separate component that receives broadcast items from the monitor. By coverage it is referred to functional coverage.

2.1.13 Interface

The interface is the connection between the UVC and the DUT. It is typically a standard bus, such as AHB, and consists of all the control and data signals of that particular bus.

2.2 Near Field Communication (NFC)

NFC is a communication technology enabling data transfer through physical proximity. It is coming up as one of the IoT technologies and enables an interactive and secure way of communication. One particular use case is safe pairing of two Bluetooth devices. It is potentially easier and less time consuming when pairing with many devices. The NFC Forum [6] develops specification and test mechanisms that ensure consistent, reliable NFC transaction worldwide.

This chapter will elaborate on the characteristics of the carrier that is essential in the communications between NFC devices using the NFC-A technology. Details about how the carrier is modulated to pass information and how information is represented will be given.

2.2.1 NFC-A Technology

NFC-A technology is one of several modulation techniques and protocols used by NFC devices. When using NFC there is a notion of a sender and a receiver. The sender notifies the receiver, through its physical proximity, and starts communication. Communication is performed over an inductive coupling between the sender and the receiver. Information is modulated on a carrier with a frequency (f_C) of 13,56MHz that passes from the sender to the receiver though the inductive coupling. Throughout this report the device generating the carrier will be referred to as the polling device and the device that receives the carrier will be referred to as the listening device. Figure 2.2 illustrates the polling and the listening device in physical proximity and the direction from which device generates the carrier. The carrier is modulated by both the polling and the

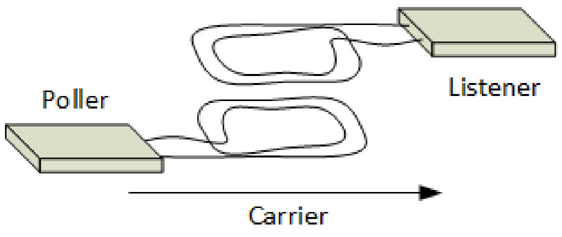


FIGURE 2.2: Illustration of physical between a polling device and a listening device.

listening device to send requests and responses in order to communicate.

2.2.2 Carrier Specification

To ensure reliable communication between the polling and the listening device the carrier characteristics have to guarantee a set of requirements specified by the NFC Forum in [7]. Figure 2.3 depicts the characteristics of the carrier during modulation by a polling device. Table 2.1 depicts the requirements that is extracted from the carrier specification in Figure 2.3. Note that if the rise

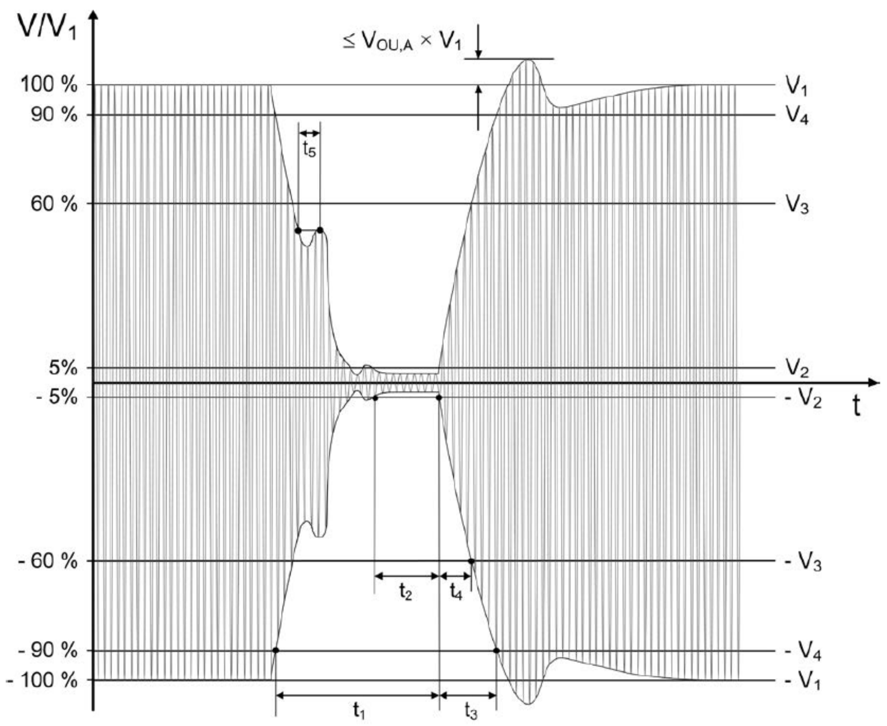


FIGURE 2.3: Modulation polling device to listening device for NFC-A [7, p. 34].

TABLE 2.1: Requirements of the NFC-A carrier.

Requirement	Min Value	Max Value
Pulse width	t_2	t_1 + t_3
Carrier amplitude	V_4	V_1
Modulation amplitude	$\sim$ 0%	V_2
Fall time	0	t_1 - t_2
Rise time	0	t_3

and fall times are max, the pulse width has to be min. The specification also allows overshoots immediately following the rising edge of the modulation which shall remain within (1$\pm$V$_{OU,A}$·V$_1$). V$_1$ is the initial voltage level measured immediately before the first modulation is applied by the polling device. The legal frequency and period of the carrier is given by Equation 2.1.

$$f_C \in [13.553, 13.567]\text{MHz} \Leftrightarrow T_C \approx [73709, 73784]\text{ps} \tag{2.1}$$

2.2.3 Polling $\rightarrow$ Listening Device

Sequence Format and Requirements

Figure 2.4 illustrates the typical sequence format of a polling device. It shows how the analog signal is modulated using the Modified Miller coding with Amplitude-Shift Keying (ASK) 100%. The amplitude of the carrier is modulated either as a 100% V "on" or below 5% V "off". Varying the time the carrier is

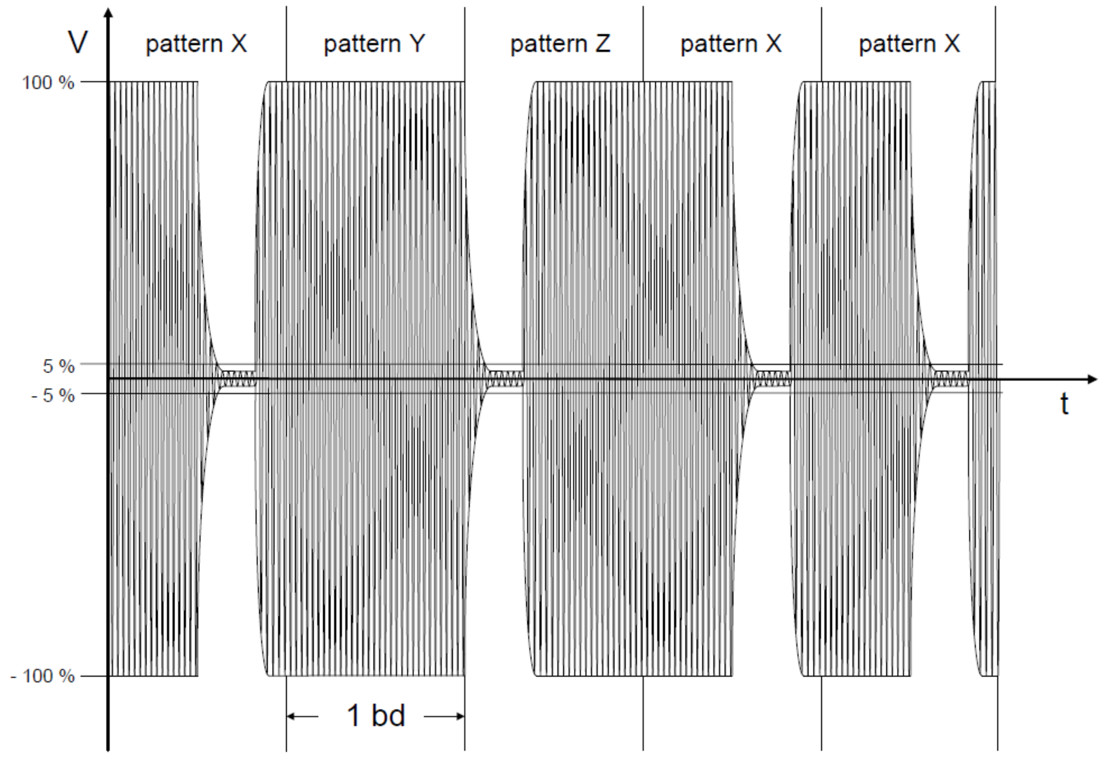


FIGURE 2.4: Modified miller coding with ASK 100% [8, p. 15].

"on" results in patterns X, Y and Z. Figure 2.4 depicts the different patterns and the patterns relation to the bit duration (bd) [8, p. 12]. The requirements for building patterns are listed as follows:

- Pattern X is always "on" in the first half of the bd and "off" in rest of the bd.
- Pattern Y is characterized by the carrier being "on" for the complete bd.
- Pattern Z is always "off" in the first half of the bd and "on" in the rest of the bd.

The listening device should treat all patterns other than X, Y and Z as illegal.

Synchronization and De-synchronization

The NFC-A requires no synchronization in terms of synchronizing a sequence.

De-synchronization is characterized as the pattern Y. Pattern Y can be read by the listening device as the End-of-Frame (EoF) when:

- it detects a pattern Y after a pattern Y
- or it detects a pattern Y after a pattern Z.

Bit Level Coding

The patterns modulated by the polling device are used to represent the digital alphabet. Pattern X shall represent a logic ′1′ and pattern Y a logic ′0′. The exceptions are:

- Start-of-Frame (SoF) logic ′0′ shall be represented as pattern Z.

- All contiguous logic ′0′s except the first are represented by pattern Z.

2.2.4 Listening → Polling Device

Sequence Format and Requirements

Figure 2.5 illustrates the typical sequence format of a response from the listening device. The analog signals is modulated using Manchester coding

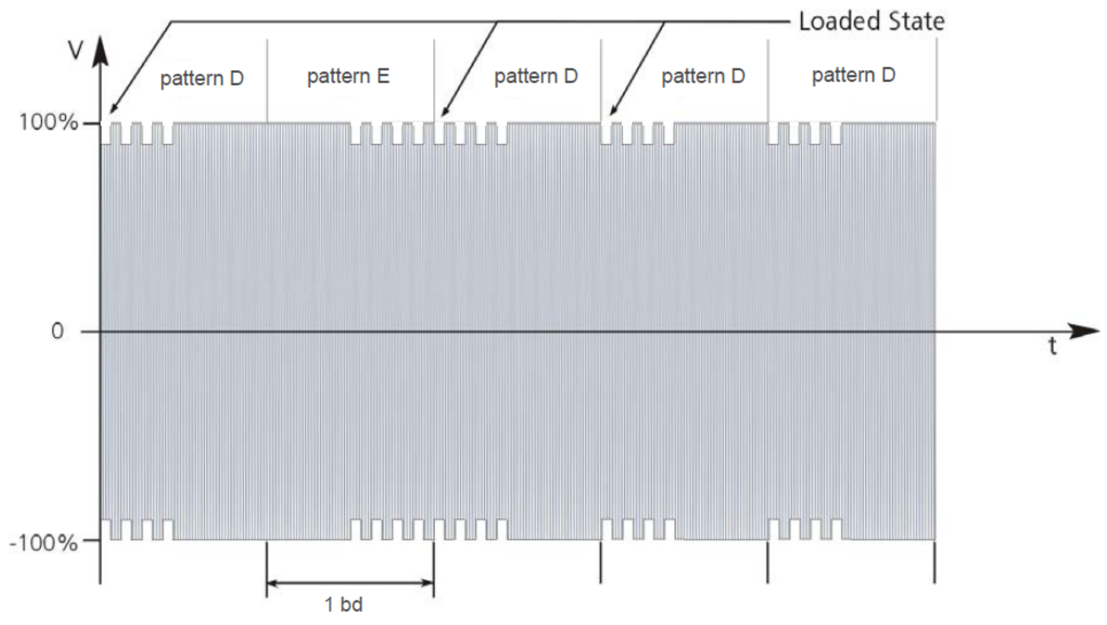


FIGURE 2.5: Manchester coding with OOK [8, p. 17].

with On-Off Keying (OOK). The listening device uses the carrier frequency (f_C) from the polling device to derive a subcarrier ($f_C/16$) to perform OOK. Applying the OOK subcarrier modulation on the carrier defines the two patterns D and E with respect to bd as depicted in Figure 2.5. If no modulation is applied for one bd the patterns is defined as F. The requirements for building patterns are listed as follows:

- If the carrier is modulated by the subcarrier only for the first half of the bd it shall be treated as pattern D.
- If the carrier is modulated by the subcarrier only in the last half of the bd it shall be treated as pattern E.
- If no subcarrier modulation is performed for a complete bd it shall be treated as pattern F.

If the polling device detects patterns different from pattern D, E and F, it should be treated as illegal patterns.

De-synchronization

If the polling device detects a pattern F from the listening device, it should treat it as a EoF.

Bit Level Coding

The patterns modulated by the listening device is used to represent the digital alphabet. Pattern D shall represent a logic ’1’ and pattern E a logic ’0’.

2.2.5 Frame Format

There are three types of frame formats used for NFC devices configured with the NFC-A Technology: Short Frame, Standard Frame and the Bit Oriented SDD Frame. All formats are based on a payload encapsulated with an SoF and EoF. For more details on the structure of each frame format, please refer to [8].

2.2.6 Command Set

The payload exchange between to NFC devices consist of requests and responses. The frames of the NFC-A protocol is described in Table 2.2.

TABLE 2.2: NFC-A Command Set [8, p. 24].

Request	Response	Frame Type
ALL_REQ, SENS_REQ		Short Frame
	SENS_RES	Standard Frame
SDD_REQ		Bit Oriented SDD Frame
	SDD_RES	Bit Oriented SDD Frame
SEL_REQ		Standard Frame
	SEL_RES	Standard Frame
SLP_REQ		Standard Frame

2.2.7 Timing Requirements

The NFC-A Technology poses timing requirements on the communication between the polling and listening devices [8, p. 37-43]. Three Frame Delay Times (FDT) timing requirements are associated with the timing between two consecutive frames. The FDT is described in Table 2.3. The FDT is dependent

TABLE 2.3: Timing requirements between consecutive frames of the NFC-A protocol.

Requirement	Description
FDT$_{A,L}$	Timing between a request and a response. The minimum timing requirement are equal for combinations containing an ALL_REQ, SENS_REQ, SDD_REQ or SEL_REQ. For these combinations, the maximum value of the requirement equals the minimum value.
FDT$_{A,P}$	Timing between a response and a request. The minimum timing requirement is equal for all combinations of responses and requests. No maximum value is defined for this requirement.
FDT$_{A,PP}$	Timing between two consecutive requests. The minimum timing requirement for all combinations of consecutive requests are the same except if the first request is an SLP_REQ. No maximum value is defined for this requirement.

on the last pattern of the first frame and the first pattern of the last frame. Figure 2.6 illustrates the dependency for the timing constraint FDT$_{A,L}$. Similarly, is the

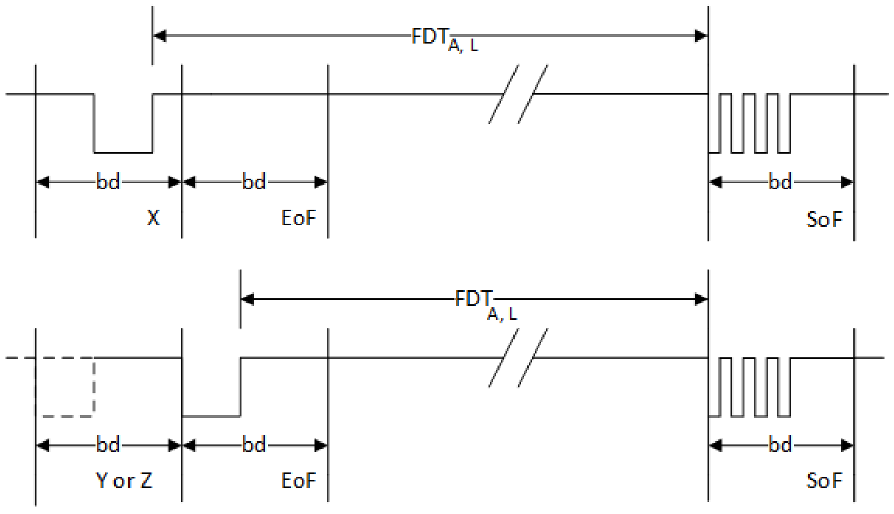


FIGURE 2.6: Illustration of the dependency between the last pattern of the frame and the FDT$_{A,L}$ [8, p. 38].

dependency for the FDT$_{A,P}$ and FDT$_{A,PP}$.

2.3 Verification Planning

Verification of hardware designs makes up a great deal of the time spent during development of a new hardware module. As opposed to software modules the verification phase of hardware is more critical due to the significant cost involved with re-spins. A re-spin involves fixing the erroneous hardware before starting a new fabrication process which is both expensive and time consuming. Erroneous software can often be fixed by releasing a software patch which, compared to a hardware re-spin, is much cheaper. Thus, it is important that the process of verifying hardware is thorough. To ensure that the design is carefully tested it is advisable to have a strategy for testing the design's intent. A verification plan is such a strategy and will be the topic of this chapter. How to compose a verification plan from the design specification and how to answer the important question "Are we done verifying?" will be discussed based on Mentor Graphics' Coverage cookbook [9].

2.3.1 Coverage

The goal of the verification phase is to determine if all the structures of the design have been exercised and that design features and requirements have been verified. Coverage is a metric aimed to help determine this goal and to optimize the test procedure of the design. Controllability and observability are two important concepts of coverage. Controllability refers to the possibility of activating a structure of the design through stimulating the inputs of the design. Stimulating the inputs gives good controllability of the design's model, however this method has low controllability of the internal structures of the design. Consequently, a bug within the design could not be discovered as it did not propagate to the output ports. In contrast to controllability, observability refers to the ability of observing the effect on the internal structures of the design. A good testbench should have good controllability and observability. Issues related to poor controllability and observability can be addressed with code coverage and functional coverage.

In order to explain coverage a classification of coverage based on their method of creation or by their origin of source can be illustrated as seen in Table 2.4. Functional coverage is extracted from the design specification by

TABLE 2.4: Different categories of coverage [9].

Implicit	Code Coverage	Area of Research
Explicit	Assertions	Functional Coverage and Assertions
	Implementation	**Specification**

the designer, thus it is an explicit type of coverage. Assertions are gathered from the internals of the design and contributes to the overall observability of the testbench and is therefore related to the implementation and explicitly defined by the designer. Implicit coverage, such as code coverage, is the metrics extracted by the simulation tools. Different types of code coverage can be extracted by the tools and includes, among others, toggle, line, statement, block, branch and expression coverage. Please refer to [9] for detailed information about different code coverage. Code coverage is used to measure how well the design has been exercised by the testbench. It is an important measure which tells whether or not the complete design has been activated during testing. However, 100% code coverage does not guarantee that a 100% of the design functionality has been tested. Functional coverage was invented as a metric for how well the design functionality has been tested and consists of both data and temporal components. Cover group modeling refers to the data related components of functional verification and is essentially coverage collected by sampling state variable of the design. Often it is important that a sequence in the designs functionality follows a temporal requirement which can be handle by cover property modeling. This type of functional coverage can be addressed with assertions.

Implicit coverage based on specification refers to simulation tools' ability to extract coverage from specification. This is currently still an area of research and is not the concern of this thesis.

The goal of verification is to reach 100% of both code coverage and functional coverage. However, verification is an incomplete process and even for small designs it can be difficult to verify everything in time. Consequently, functional coverage should be prioritized and covered in an iterative manner to address the most critical functions of the design before tape-out.

2.3.2 Test Plan to Functional Coverage

In order to be able to define good functional coverage the behavior of the design must be captured, thus it is necessary with a good test plan. In [9] it is illustrated two approaches on how to set up a good test plan, the bottom-up and the top-down approach. Different conditions favor one or the other approach. The bottom-up approach is suited for small designs and designs with a good design or implementation specification. Coverage is extracted from the low-level requirements which makes it easy to link and close on coverage goals. However, this technique could lead to an explosion of requirements that needs prioritization. In contrast, the top-down approach extracts requirements from high level architectures and works best with large designs. Coverage is not as easily linked to the specification of the design, as for the bottom-up approach, however high level coverage from the top-down approach could be useful for exploration of design trade-offs. The interested reader is referred to [9] for more details on the two approaches.

Once the test plan is created it should include what features have to be tested, under what conditions the features should be tested, how the design's interface should be stimulated and monitored and how the testbench will check that the features work. From the test plan the functional coverage model can be extracted based on covergroup and cover property modeling. The important considerations of covergroup modeling can be summarized as illustrated in Table 2.5. Having these questions in mind during the creation of a functional

TABLE 2.5: Summary of covergroup modeling [9].

Coverage Criterion	**Feature on which data coverage is to be collected**
Which values are important?	Identify the important values to hit
What are the dependencies between the values?	Identify the important cross products between data values
Are there illegal conditions?	Identify values, or combinations of values that should not occur
When is the right time to sample?	Specify a valid sampling point
When is the data invalid?	Identify conditions when the data should not be sampled

coverage model helps closing on testing of the designs functionality.

2.4 Related Work

Examples of testbenches using UVM are given in [10] and [11]. The work in [11] gives a detailed overview of the components of a UVM testbench and how they interrelate in order to generate and monitor stimulus. UVM is adapted in [10] to create a methodology suited for mixed-signal verification such that reusable verification components can be used for analog modules. It describes the concept of an interface UVC through what is referred to as a wire UVC which is similar to the UVC discussed in this thesis. Both [10] and [11] illustrate that UVM is suited for Analog Mixed-Signal (AMS) and RVM simulations and that benefits can be exploited at IP and system level. The work of these to paper aims increase functional coverage of analog modules and points out that SPICE simulations have trouble in achieving that goal in time due to poor performance. In [2] it is stated that using RVM simulations compared to SPICE simulations can increase the performance of the simulation up to 100 times. However, as [10] emphasize, there is an extra cost involved with creating AMS and RVM models as they cannot mitigate the accuracy in SPICE simulations.

The work in [12] discuss an interesting approach designing a layered testbench. It shows how a testbench and its components can be categorized into layers representing different functions and purposes. Arguably, this approach breaks down the complexity of the testbench and components can be created separately in more manageable pieces.

The concept of MDV is verification technique encouraged in [2], [10], [11], [13] and [14] and is a concept based on Constrained Random Verification (CRV) and Coverage Driven Verification (CDV). MDV is originally a technique used in verification of digital design, however these paper encourage the use of MDV in mixed-signal verification as well. In short MDV depicts that stimulus is automatically generated at random and constrained to hit relevant scenarios. Coverage is collected through self-checking mechanisms and used to reach closure on functional coverage.

To mitigate for the poor speed performance of SPICE simulations, modeling techniques based on AMS and RVM can used to increase the performance in mixed-signal verification as described in [2], [10], [13], [14] and [11]. AMS is the most accurate modeling technique, compared to RVM, but requires both digital and analog solvers in a mixed-signal simulation. Better performance can be achieved with RVM as these models are suited for digital solvers as described in [15] and [2], however accuracy is compromised even further with this technique. Both AMS and RVM are models of the analog circuits in the simulation which has an advantage not only increasing performance but the models can be developed in an early stage of the design.

Chapter 3

Design

Verification IP (VIP) is probably a familiar term among hardware design and verification engineers. In the Very-Large-Scale Integration (VLSI) business it refers to a stand-alone reusable module that includes all the necessities for performing verification of a hardware module. This hardware module could be a small IC like a peripheral unit or more commonly a communication bus/interface on a SoC. A VIP could be used in a large testbench with other VIPs or it could be used only to verify its intended IP. The VIP is a testbench component which is more similar to software than actual hardware. Any VIP based on UVM is referred to as a UVC, which is the main topic of this thesis. Naturally, the UVC will make use of all the benefits of UVM such as reusability, object-oriented programming (OOP) and MDV.

The author presented the development of a UVM verification IP for an analog RF protocol in an earlier project [3]. This UVC targeted the NFC interface and more precisely the NFC-A protocol. In accordance with UVM a framework was set up and the UVC was able to both drive and monitor stimulus of the NFC-A protocol. However, the UVC was only able to handle requests of a polling device, as described in Chapter 2.2.3. Thus, it was only able to verify parts of the NFC-A protocol and was not able to verify a NFC-A listening device. This thesis is a continuation of that design and aims to extend and improve the design's functionality such that the UVC can be used in verification of the NFC-A listening device protocol behavior.

This chapter will present the design that is implemented and challenges addressed during the design process. The design process involved many considerations and possible solutions to several design components and arguments for the chosen design will be given. The NFC-A is a large protocol and consequently parts of the protocol is left out of the scope of this thesis. However, as this thesis will discuss, parts that are left out are considered in the design process to facilitate for future extension.

Several components of the design are reused from the project, however, these components have either been slightly modified of completely changed to fit the new design. The design considerations have a slight overlap with the project, but are included here they are regarded important for the understanding of the UVC. A summary of the chosen design and requirements will be given in Chapter 3.6.

3.1 The Design

The design is based on the verification methodology UVM which puts requirements on how the framework of the UVC is set up, but it also provides powerful benefits for verification. As mention in Chapter 2.1 UVM depicts a hierarchy of OOP-based components. The different components serve a small part of the functionality needed to perform MDV. Some components are directly concerned with handling of stimulus. Such components include drivers, monitors, scoreboards, coverage collectors and other components, and some components are responsible for setting up other components and handling their connections. When all the components are connected in a configuration they can perform the intended verification. Figure 3.1 illustrates the final result of the design process and the final UVC.

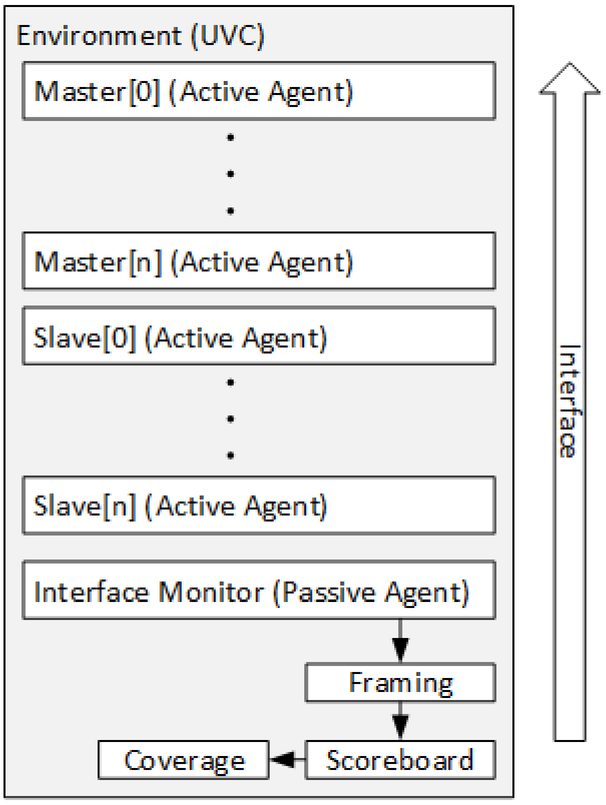


FIGURE 3.1: Design of hierarchical component Environment.

The hierarchical component environment encapsulates all the agent components, scoreboard, coverage and a framing component and is responsible for connecting components to the interface and interrelated components together. As the figure illustrates, the active agents are either configured as a master or a slave and the instance number of these agents are configurable. A passive agent, the interface monitor, surveys the interface and broadcasts sequence items to the framing and coverage component. The important self-checking mechanism required for MDV is provided by the monitor and scoreboard. The scoreboard receives interface information from the monitor through the framing component.

The agents are configurable as either active or passive, as depicted by UVM. Figure 3.2 illustrates the design of the agent. If used as an active agent the

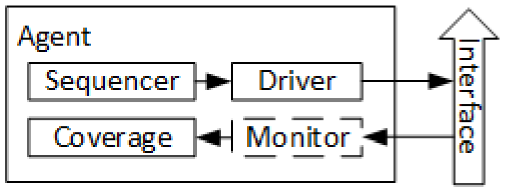


FIGURE 3.2: Design of hierarchical component Agent.

component will be configured with the three components, sequencer, driver and monitor. However, if the agent has a passive configuration the components is only configured with a monitor and the active components are disabled.

3.2 Analog Interface Model

Digital simulations and analog models are in conflict due to the discrete nature of the simulation and the analog nature of the model. However, in order to be able to verify the analog NFC-A protocol a model has to be designed for the interface signal. Otherwise it would not be possible to drive or monitor stimulus going to and from a DUT. Chapter 2.2.2 describes the characteristics and requirements of the NFC interface. Together with information about the DUT, an analog model is designed that enables the required verification. Figure 3.3 illustrates the analog model of the NFC carrier during modulation. Associated with the carrier are a few characteristics such as the amplitude with

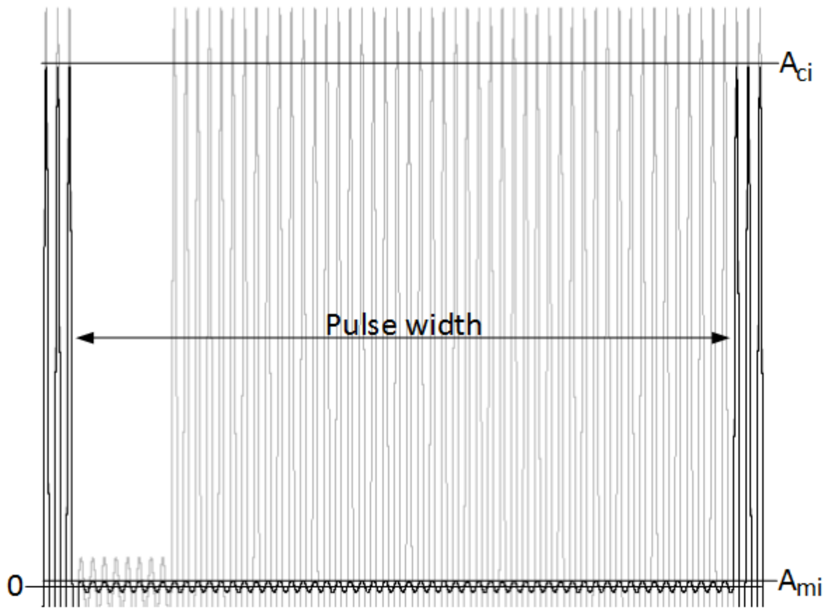


FIGURE 3.3: Illustration of discretized analog model of the NFC interface during modulation.

and without modulation, indicated in Figure 3.3 as A_{mi} and A_{ci}, and the pulse width of a NFC-A pattern. The NFC-A protocol allows these characteristics to vary inside a range which is illustrated in Figure 3.3 as the difference between the black and light gray sinusoidal carrier signals.

The analog model is realized by using the SystemVerilog real data type which is a floating point type variable suited for analog-like level modeling. The carrier model illustrated in Figure 3.3 is similar to the model described in Chapter 2.2.2, however, the model is simplified. Rise and fall times related to start and end of modulation is neglected as it is not a part of the RVM model of the DUT. Overshoots are also neglected from the model as their analog behavior are complex to model discretely. The designed model does however allow NFC-A protocol communication in a basic form.

3.3 Layer-based Design

Layering is a principle found in UVM which facilitates reuse and provide high-level abstraction. The UVM sequence is one of such examples in UVM where interesting stimulus can be comprised by multiple sequence items and later be reused to create new and more complex sequences. Analogous to the NFC-A protocol the sequence item could represent a pattern, a sequence could comprise these patterns into a request and the requests could assemble an initialization routine in another sequence. The technique of layering is advantageous in several ways and is widely used not only in UVM sequences. A good example illustrating layering is the OSI model [16]. All the layers of the OSI model is a logical succession of the lower layers which enable network entities to connect to the different layers. Thus, selective information can be obtained from within the hierarchy of layers. This property could be useful inside the UVC and will be the topic of this chapter.

As mentioned in Chapter 3.1 the UVM environment has a modular structure of reusable components. These components cover different functionality and could be considered to reside in different layers. The UVC is an interface UVC, thus developed for a designated protocol [4, p. 257], and the components typically cover parts of the protocol, but not necessarily. However, the driver and the monitor are the components that deal directly with the interface, thus managing the lowest level of the protocol. From a verification point of view, the monitor will perform verification at this level of the protocol. Other components in the UVC, as the scoreboard, should not have to perform same verification, but instead verify higher level protocol functionality. The monitor broadcasts sequence items, which represent NFC-A patters, to analysis components. The sequence items can be analyzed in terms of frames, which is the next natural layer in the NFC-A protocol, by the receivers of the monitor. Receivers would typically be the scoreboard, but could be any analysis component. Verification of NFC-A frames, by the scoreboard, would involve checking type of requests and responses and timing between consecutive frames. Higher level protocol functionality than NFC-A frames is beyond the scope of this thesis. Figure 3.4 illustrates the components of the UVC and how they relate to the protocol hierarchy. This ordering of components could

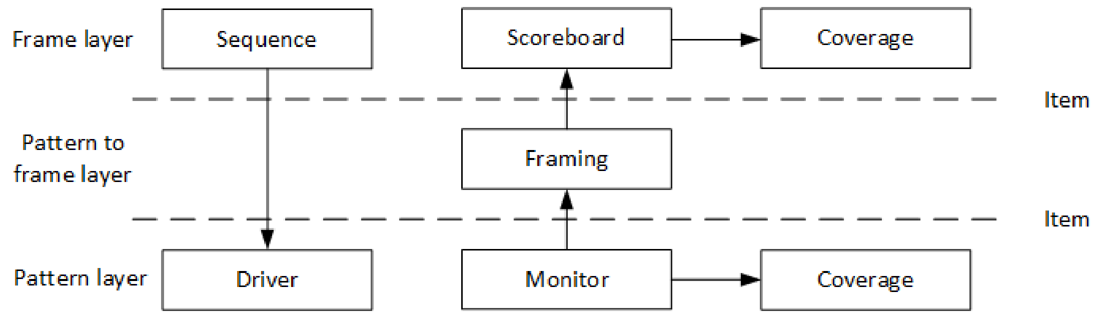


FIGURE 3.4: UVC components illustrated in relation to the NFC-A protocol hierarchy.

increase readability for debug purposes and increase ease of use when adding coverage groups. Having a separate framing component for extracting frame content is relevant for scoreboards and other analysis components.

3.4 Components of the UVC

A presentation of the main UVC components and design considerations will be given in this chapter.

3.4.1 Sequence Item

The sequence item has many purposes in the UVC. Most importantly it holds information about the signals on the interface, but also other attributes that are necessary or convenient for communication between the UVC components. It is necessary that the sequence item holds the characteristics of the NFC-A carrier so that the driver can produce the discrete analog model described in Chapter 3.2. The same characteristics are also used by the monitor when it extracts carrier information and creates a sequence item for analysis by the scoreboard and coverage collectors. Due to the layer-based the design of the UVC, the sequence item has to encapsulate both frame and pattern related information. Thus, the data members of the sequence item are split into these two categories. The sequence item is the lowest level of randomization in the UVC, and the data members related to the carrier characteristics are declared as random variable and constrained according to the NFC-A protocol. A copy and a compare function is designed to be used by components like the scoreboard.

Reuse of the sequence item is limited. A UVC is typically defined by the sequence item, thus this item holds the information that is unique for the test purpose.

Two slightly different designs for the sequence item have been considered. Both designs are based on different time intervals of the NFC interface. The first design held the carrier information related to one single period on the interface. This enabled great flexibility in terms of randomization of the carrier which could be used to create very realistic stimulus. However, the chosen solution represents an NFC-A pattern as described in Chapter 2.2.3 and 2.2.4. This restricts the randomization of stimulus to per pattern, but it is an intuitive representation of the NFC-A stimulus.

3.4.2 Driver

When the UVC is ready to drive new stimulus onto the interface, the driver requests a sequence item from the sequencer. Upon reception of this sequence item the item has to be translated into interface signals. The driver handles this translation by setting up a carrier on the interface and extracting the carrier characteristics contained in the sequence item as well as modulation information. The period, carrier amplitude and modulation amplitude information is fed into a mathematical model which emulates a discrete representation of an analog signal. The sampling frequency of this signal is determined by the simulation timescale and is 1 ns by default, however the driver is easily configured to work with another timescale.

The driver design is based on a previous design provided by the supervisor of this thesis, Christoffer Amlo. The mathematical model has been customized to fit the UVC and new functionality has been added to support all patterns of the NFC-A protocol. In addition, the driven stimulus is synchronized to the start of a new period on the carrier for better control of the stimulus.

3.4.3 Monitor

The purpose of the monitor is to capture information from the interface and reassemble a sequence item as a container of the information. This is a necessary step in the self-checking capability of the UVC. The requirement for the monitor is based on the characteristics of the carrier signal and the verification goal of the UVC. Interesting carrier characteristics are period, carrier amplitude, modulation amplitude and pulse width.

A number of different designs were considered for the monitor and two designs in particular will be described here. One design was based on two monitor implementations, one covering listening device patterns and one covering polling device patterns. The intention was to create agents which drove requests and monitored responses and vice versa. This is an intuitive set up for the agents and it would require only one agent configured as a polling device driver to verify a listening device. The principle of reusability based on class extension could be used in such a design when creating the two monitor implementations. Both monitors could extend a base class containing the similar routines of the monitors. This solution would add complexity to the configuration routine of the UVC when configuring with the correct monitor. However, a problem occurred when trying to facilitate for extending the UVC. If the UVC is configured with several slave agents the environment would be configured with many monitors performing the same task. As the monitor is the most active component of the UVC, monitoring the interface each simulation step, this solution would lead to unnecessary use of resources. Even though handling the resource usage of such a design is simply done by disabling all but one of the slave agent monitors, this solution seems a bit awkward and counter intuitive.

The final design is based on a concept described in [5, p. 174], an interface/bus monitor, which address some of the design issues related to the design discussed above. An interface monitor is a passively configured agent that monitors all interface traffic as opposed to an ordinary monitor which could monitor traffic pertained with the particular agent. An interface monitor solves the resource usage problem with a minimal solution that requires only one monitor implementation. Consequently, the interface monitor will have to implement both monitoring of polling and listening device patterns which will result in a more complicated component.

The monitor is designed to capture all the characteristics of the carrier as well as the start and end of modulation. Based on this information the monitor is able to determine the type of pattern that it represents and broadcasts a sequence item to analysis components such as the scoreboard. The monitor synchronizes on the SoF condition in order to capture the carrier characteristics of each pattern residing inside a frame. An algorithm determines if the monitor should wait for the SoF condition or start recording information of the patterns. The algorithm ensures that the monitor stops recording when the EoF condition occurs.

A simple design has been preferred for the monitor component as recording of pattern information requires a relatively complex algorithm. The monitor has a notion of the SoF and EoF, but it is considered a pattern layer component as it only performs analysis of patterns.

3.4.4 Frequency Calculation

The carrier signal of the analog model in Chapter 3.2 is given as a sine of amplitude A and phase θ as described in Equations 3.1 and 3.2.

$$S = A \sin \theta \tag{3.1}$$

$$\theta = 2\pi f_C t = 2\pi k T_s|_{k\in[0,74]} \tag{3.2}$$

The phase of the signal is determined by the frequency of the carrier f_C at a given time t, which is equal to an amount of simulation timesteps kT_s. The frequency of the carrier can vary inside the range described in Equation 2.1 from Chapter 2.2.2. Equation 2.1 states the approximate resulting period of the carrier signal which determines the range of k in Equation 3.2 with a T_s = 1ns. Refer to Chapter 2.2.2 for information about the carrier signal. Changing Equation 3.1 with respect to the phase, the phase can be described by the amplitude of the signal and the value of signal at a certain timestep, as described in Equation 3.3.

$$\theta = \arcsin S/A \tag{3.3}$$

By sampling the signal at a point T_S and T_E in time, the phase difference between the two sampling points can be calculated, as described in Equation 3.4.

$$\Delta\theta = \theta_E - \theta_S \tag{3.4}$$

Combining Equation 3.4 with Equation 3.2 a time difference between the two sampling points T_S and T_E can be calculated, as described in Equation 3.5.

$$\Delta t = \frac{\Delta\theta}{2\pi f_C} \tag{3.5}$$

If the two sampling points T_S and T_E are approximately one signal period apart, the time difference Δt corresponds to the error in the measured period of the signal. This can be used to mitigate for a frequency estimation error that occurs with a simulation timestep of 1 ns. That is, carrier frequency is either measured to 73 ns or 74 ns. However, Equation 3.5 is dependent on the frequency of the signal in order to calculate the time difference. At the time the frequency is unknown and the frequency is the target result of the calculations. As Equation 3.6 describes, the error in Δt calculated with a f_{AV} = 13,56MHz is never off by more than 0.14% if the actual frequency f_C = f_{WC}. f_{WC} equals the frequency at the boundaries of the legal frequency range.

$$\Delta t_{ERROR} = \frac{\Delta\theta}{2\pi} \cdot \frac{1}{|f_{ACTUAL} - f_{AV}|} = \frac{\Delta\theta}{2\pi} \cdot \frac{1}{|f_{WC} - f_{AV}|} \approx 0.14\% \tag{3.6}$$

This is due to the fact that the time difference is always less than 1 ns and that the worst case difference in frequency f_{WC}, that is, at the boundaries of the legal frequency range, as a result of the approximation is accurate to more than 99.8%. That means a total error in the time difference calculation of less than 0.2 ps. Thus, the frequency of the carrier can be calculated, as described in Equation 3.7.

$$f_C = \frac{1}{T_E - T_S - \Delta t} \tag{3.7}$$

3.4.5 Agent

The agent is described in Chapter 3.1 and is a component only related to the structure of the UVC. By design, this component is completely reusable, if following the guidelines of UVM. It does not directly relate to the layers of the UVC, however, it encapsulates the pattern layer components.

A feature for disabling the monitor instantiation of the agent was design to reduce unnecessary simulations cycles if an interface monitor agent is present in the UVC. The configuration of the agent is set in the coupled agent configuration object.

3.4.6 Framing

The framing component is a custom component designed for the NFC-A protocol. Its purpose is to extract the contents of an NFC-A frame.

Even though the framing component is specific for the UVC it is reusable. It works as a kind of middle man between the frame and pattern layer of the UVC by extracting the frame content of a frame. This is the first step in frame analysis for any NFC-A high-level protocol analysis component, thus analysis component can receive items from the framing component and less code is needed.

3.4.7 Scoreboard

After the monitor, the scoreboard is the second component contributing to the self-checking ability of the UVC. The scoreboard should handle high-level verification of the interface, as the monitor performs verification of patterns.

A scoreboard is typically not a reusable component of a testbench as it performs very specific checks based on the type of interface. However, the components of a UVC provides functionality that can help make the scoreboard reusable. Even though UVM facilitates for reuse of the scoreboard, it cannot be completely reusable as a model describing how to perform checks of the scoreboards inputs must be implemented.

An interesting scoreboard design is proposed in [17, p. 13] where a reusability is the main concern. The design is very modular an is built on four components: the scoreboard, a comparator, a predictor and a model for the predictor. The model for the predictor is the only component that is not reusable in the design, which main task is to predict the item on the other input based on the received item of the first input. The comparator receives the predicted item from the predictor and performs only a check whether they are identical.

However, a minimalistic scoreboard was designed consisting of only one component. The input is compared against the output through the use of the sequence item compare function. Both the design proposed in [17, p. 13] and the chosen design are based on UVM fifos on the inputs.

3.4.8 Environment

As mentioned in Chapter 3.1, the purpose of the environment component is to set up the necessary components required for a specific testbench. The environment is the top-level component of the UVC, thus it is referred to as the UVC, and performs only build and connect configuration of other components. The environment is reusable across different testbenches with a requirement for NFC-A protocol verification. However, the environment could be configured for ether testbenches by disabling the NFC-A specific components. It is configurable through a coupled configuration object to create different UVC configurations suited for different test scenarios. Examples of such scenarios could be only two communicating NFC Devices or a more interesting scenarios with a one polling device and several listening devices. The UVC cannot perform verification of the latter scenario, however, the environment facilitates for setting up the necessary components for this scenario. Having a typical design depicted by UVM improves facilitation of adding new components to the design.

3.5 Stimulus Library

As mentioned in Chapter 2.1.7 stimulus in UVM is based on sequences that describe a combination of sequence items. A library of sequences describing the polling device request given in Table 2.2, except for the SLP_REQ, is designed with the UVC. The purpose of the library is give the user of the UVC a head start in creating stimulus for different testbenches. However, the sequences have limitations. The collision resolution part of the NFC-A protocol is not described by the sequences and consequently a fixed ID for the receiver of the stimulus is embedded in the sequences. The library also works as an example for creating new stimulus.

The sequences in the library is extended from a base sequence. The base sequence instantiates a sequence item and holds functions for generating random sequence items based on the NFC-A patterns described in 2.2.3. The functions are able to produce SoF and EoF patterns as well as X, Y and Z patterns. Higher level protocol sequences are extended from the base sequence to create relevant requests.

The request library is based on one sequence item which holds information about the NFC-A frame and pattern layer. Another interesting method proposed in the UVM 1.2 User's Guide [5, p. 147–148] is to create separate sequence items for the different layers in the protocol. This technique differs from the one used in the request library in that sequences are based on virtual sequences. This means that the sequence is responsible of executing the sequences on the required sequencer. Having two different sequence items would require two sequencers and two monitors in order to drive and monitor the stimulus. It was considered as a solution for the design of the request library, however it required extra complexity which might be more necessary in a more complex protocol such as TCP. For future work, if the UVC should include verification of the NFC-B and NFC-F protocol, this technique could be considered.

3.6 Chosen Design and Requirements

A brief summary of the requirements for the UVC implementation based on the argumentation of this chapter is listed below:

- The monitor should be capable of monitoring:
  - both polling and listening device patterns that reside inside an NFC-A frame
  - the analog characteristics carrier amplitude, modulation amplitude, frequency, pulse width and start and end of modulation
  - pattern-level protocol information only

- The driver should be capable of driving both polling and listening device patterns
- The sequence item should represent all the patterns of the NFC-A protocol
- All communication between components should use the TLM abstraction of the sequence item
- Overshoots and rise- and fall times of the carrier should not be allowed
- A sequence library should come with the UVC. It should include all necessary requests
- The UVC should be configured by a configuration object that includes a default configuration
- The scoreboard should verify timing and frame-layer protocol behavior only

Chapter 4

Implementation

A detailed description of the UVC implementation will be given in this chapter. The implementation is based on the designed developed in the design phase of this thesis. Each component of the UVC will be described separately based on the chosen design and requirements, as described in Chapter 3.6.

4.1 Environment

The environment is the top-level component of the UVC and is designed to be reusable across different testbenches. In order to make the environment generic an array of master and slave agents are declared in the class declaration as well as an interface monitor agent, a framing component and a scoreboard. Algorithm 4.1 depicts how the environment use configuration information to create the agents and other child components. Depending

Algorithm 4.1 Build configuration of the environment class.

```
class environment extends uvm_environment
    Declare agents, scoreboard, coverage and framing components
    Obtain environment configuration object from configuration database
    Set configuration for agents
    if $has\_interface\_monitor$ then
        Create interface monitor agent
    end if
    if $has\_coverage$ then
        Create coverage component
    end if
    if $has\_scoreboard$ then
        Create framing component
        Create scoreboard
    end if
    for $number\ of\ master\ and\ slave\ agents$ do
        Create agents
    end for
end class
```

on the configuration information the components are connected together in order to perform analysis and coverage checking. Figure 4.1 illustrates the four possible configurations of the components. The interface monitor,

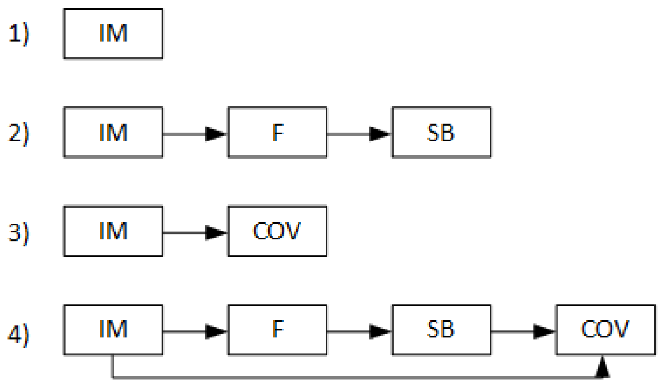


FIGURE 4.1: Possible connect configurations of the environment class components.

scoreboard, coverage and framing component are denoted IM, SB, F and COV, respectively. As an example, setting has_interface_monitor and has_scoreboard in the configuration object achieves connect configuration 2). In order for the environment to be able to compile it is necessary to use the environment configuration object. It is also necessary to set up the agent configuration object for the master, slave and interface_monitor agent as the environment passes the configuration down the hierarchy.

4.2 Agent

The agent is designed to be either an active or a passive component. Depending of its configuration different child component are created and connected. Common in both configurations is the creation of a monitor. If the agent is active, the stimulus component sequencer and driver is created. The agents can also be configured with a coverage component if necessary. Algorithm 4.2 shows how the agent class implementation use the configuration objects passed to the respective agents to set up the necessary components.

4.3 Interface and Sequencer

The NFC interface contains the two SystemVerilog based real data type variables $nfcAnt1$ and $nfcAnt2$. The driver is connected to $nfcAnt1$ and the monitor is connected to $nfcAnt2$.

The sequencer is instantiated from the sequencer class which is extended from the UVM sequencer with no modifications. The UVM sequencer class provides the necessary arbitration and randomization of sequence items that is required.

Algorithm 4.2 Set up configuration of the agent class.

```
class agent extends uvm_agent
    Declare sequencer, driver, monitor and coverage component
    Obtain agent configuration objects from configuration database
    Create monitor
    if $is\_active$ then
        Create sequencer and driver
        Connect sequencer item export to driver item port
        Connect interface to driver virtual interface
    end if
    if $has\_coverage$ then
        Create coverage component
        Connect monitor analysis port to coverage component analysis export
    end if
    Connect monitor analysis port to agent analysis port
end class
```

4.4 Driver

The implementation of the driver is based on four main tasks; ta_generate_carrier_phase, ta_generate_carrier_amplitude, ta_generate_carrier and ta_drive_pattern($seq\_item\ s$). The three tasks ta_generate_carrier_phase, ta_generate_carrier_amplitude and ta_generate_carrier are closely interrelated in order to generate a sinusoidal carrier. The three tasks are synchronized such that the phase θ is calculated before the carrier amplitude A and the carrier amplitude is calculated before the level of the carrier is put on the interface signal $nfcAnt1$. An illustration of the synchronization during each timestep T_s is depicted in Figure 4.2. The ta_drive_pattern(seq_item

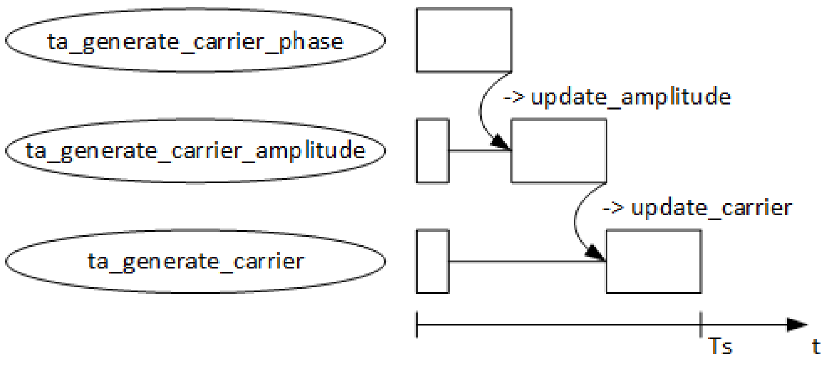


FIGURE 4.2: Synchronization of the three task generating the level of the carrier.

s) task is used to define the frequency f_C, carrier amplitude A and pulse width of the carrier. The task calculates the frequency f_C, carrier amplitude index A_{ci} and the modulation amplitude index A_{mi} from the sequence item information. The modulation bit m of the carrier is toggled using the functions fu_start_modulation and fu_start_modulation. Task ta_generate_carrier_phase generates a carrier clock which is used to synchronize when to drive stimulus. If the sequence item obtained from the sequencer is a SoF, the driver starts driving stimulus at the positive edge of the carrier clock. Algorithm 4.3 depicts how ta_drive_pattern($seq\_item\ s$) modulates and change the characteristics of the carrier based on the sequence item information. Equation 4.1, 4.2

Algorithm 4.3 Task ta_drive_pattern.

```
task ta_drive_pattern($seq\_item\ pattern$)
    Collect pattern characteristics from sequence item
    Update the characteristics of the driven stimulus
    case $pattern\_type$
        Modulate the carrier based on pattern type
    end case
end task
```

and 4.3 shows the mathematical expression used in the calculations of the carrier in ta_generate_carrier_phase, ta_generate_carrier_amplitude and ta_generate_carrier, respectively.

$$\theta = \theta + 2\pi T_s f_C \tag{4.1}$$

$$A = A_{ci}!m + A_{mi}m \tag{4.2}$$

$$nfcAnt1 = Asin(\theta) \tag{4.3}$$

4.5 Monitor

The implementation of the monitor is designed to monitor patterns that either reside inside a frame on the interface or indicates a SoF or EoF. Thus, the monitor is a component representing the pattern layer of the protocol even though it operates with the notion of frames. Monitoring of the interface is carried out by a set of tasks and functions that share information through class objects and events. As a result, the monitor is able to monitor analog characteristics such as carrier amplitude, modulation degree, carrier frequency, pulse width and the start and end of modulation. The main task and their cooperation will be described in this chapter.

ta_sample_carrier samples the carrier continuously, thus detecting any possible change. By analyzing the carrier, this task determines the start of each new period of the carrier and generates a corresponding event, new_period. The new_period event is used to synchronize the **ta_decode_pattern**, **ta_detect_frame** and **ta_sample_characteristics** tasks. A bit, representing modulation of the carrier, is set if the amplitude maximum of the carrier is recorded below the modulation threshold, V2.

ta_sample_characteristics calculates the characteristics of the carrier and store them in a sequence item. The frequency is calculated using the method from Chapter 3.4.4 and the pulse width and modulation start and end is calculated from an array holding the envelope of the carrier.

ta_decode_pattern creates an array of the envelope of the carrier and decodes the current pattern type. This task is initialized at the beginning of each bit duration. On each new period event, during one bit duration, the modulation bit from **ta_sample_carrier** is evaluated to determine the carrier envelope. The information is stored in the bit array where a modulated and unmodulated carrier is represented as a logic one and a logic zero, respectively. Before returning, the bit array is analyzed for pattern type and correct pattern type and logic value is stored in the sequence item.

ta_detect_frame is a two-state Finite-State Machine (FSM) controlling when to monitor the patterns of a frame and is the main task of the monitor. With no modulation present on the interface, the task resides in a SENSE state searching for a potential SoF pattern. If modulation is detected in the SENSE state, thus potentially indicating a SoF pattern, the FSM shifts to a DECODE state. The DECODE state manages the perception of frames and the bit duration. In each execution of the DECODE state a decode of the interface patterns are performed by the **ta_decode_pattern** task and the SoF and EoF conditions are monitored. At the same time, starting the **ta_decode_pattern** task, **ta_sample_characteristics** is initiated to read the characteristics of the carrier. Patterns of the current frame monitored is sent consecutively to an analysis port. Figure 4.3 illustrates the operation of **ta_detect_frame** in detail.

4.6 Sequence Item

The sequence item holds the NFC-A protocol characteristics pulse width, carrier amplitude, modulation amplitude and period. These are the pattern layer data members of the sequence item, as well as attributes like pattern type, logic value of the pattern, SoF and EoF. Frame layer data members of the sequence item include FDT timing, modulation start and end, command type of the frame and whether the sequence item is a frame. The characteristics are constrained according to the requirements in Chapter 2.2.2. Additional constrains ensure that the logic value is set to match the pattern type of the item and for stimulus to hit the corner cases of their respective ranges. The compare function calculates timing based on the modulation start and end information and relates the timing to the command type.

4.7 Framing Component

The framing component implements the necessary step of unpacking frames, extracting information about the start and end of modulation as well as determining the command if the frame. The framing component receives the patterns from the monitor with the SoF and the EoF which encapsulates the command. The information from the frame is comprised in a sequence item that is broadcast to the scoreboard and other analysis components. Algorithm 4.4 depicts the operation of the framing component and how information is extracted from the frame.

Algorithm 4.4 Unpacking algorithm of the framing component.

```
case $framing\_state$
    IDLE:
    Wait for next pattern
    if $SoF$ then
        Save type and modulation start of current pattern
        Jump to FRAMING state
    end if
    FRAMING:
    Wait for next pattern
    if $EoF$ then
        Set send flag
    else
        Append logic value of pattern in frame content array
        Save type and modulation end of current pattern
    end if
    if $send\ flag\ set$ then
        Check type of command in frame content
        Save command and frame modulation start and end in item
        Send item
        Jump to IDLE state
    end if
end case
```

4.8 Scoreboard

The scoreboard receives sequence items through its analysis export from the framing component. The scoreboard waits for the reception of a request and a response before a check is performed whether it has received a valid response to the request according to the NFC-A protocol. The check is performed using the compare function of the sequence item. The number of passed and failed comparisons are reported at the end of simulation. The sequence items are reported to the coverage component.

4.9 Coverage

The coverage component contains two coverage groups, $cov\_pattern$ and cov_frame. The coverage groups encapsulate the user defined functional coverage points specific for the pattern and frame layer, respectively. Coverage is reported to the coverage component from the monitor and the scoreboard.

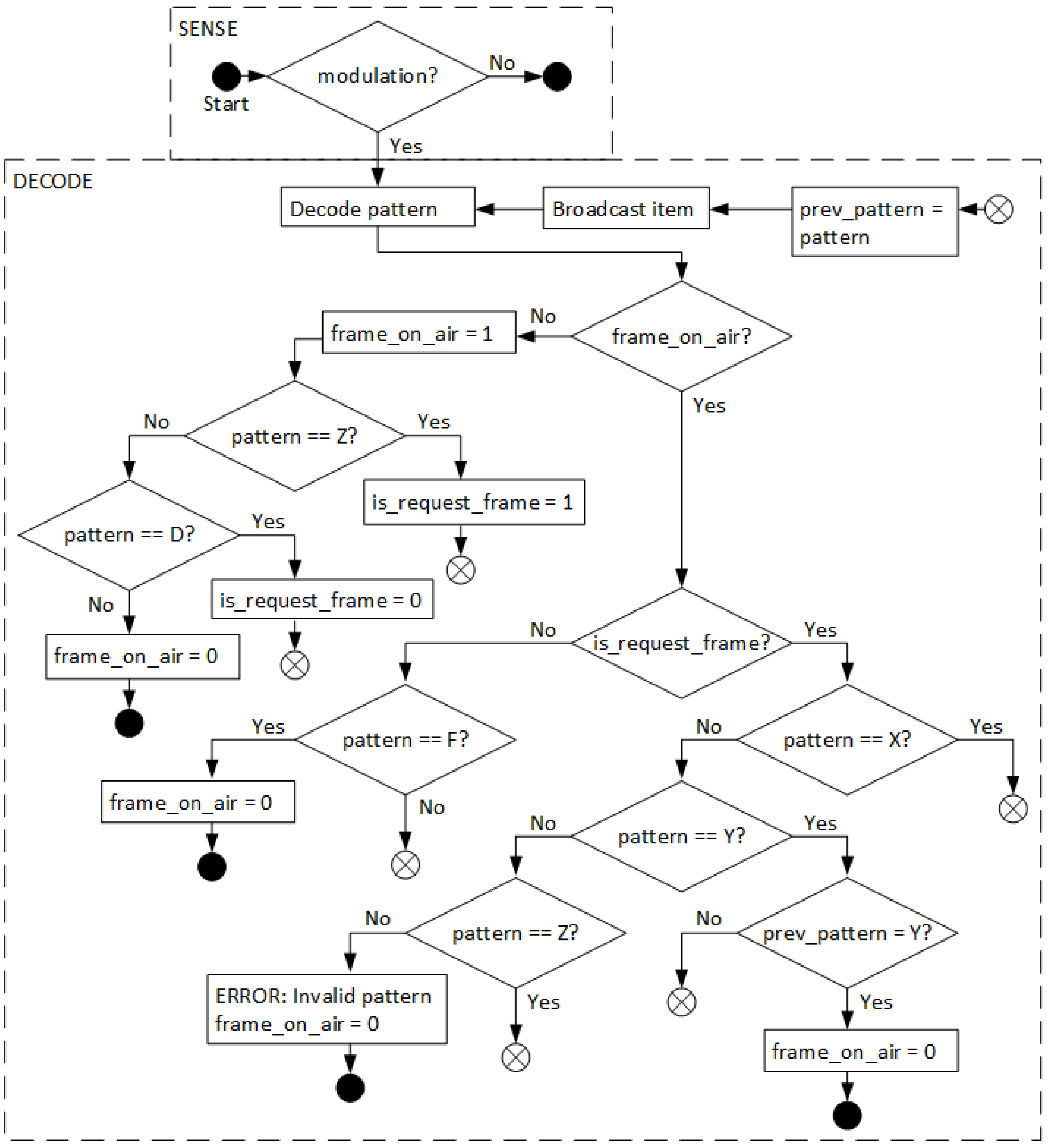


FIGURE 4.3: Operational illustration of the ta_detect_frame FSM.

Chapter 5

Test Setup

A good way to test the UVC is to create a realistic scenario of a testbench set up to perform verification of an NFC-A listening device. From Nordic Semiconductor ASAs point of view rigorous testing of their IPs are important and the UVC is designed to contribute to production of error-free designs. Nordic Semiconductor ASA provided their NFC-A listening device for this thesis as well as their original testbench as a starting point for the testing of the UVC. This chapter will describe the original NFC-A listening device testbench and how it was modified with the UVC. A test designed to verify functionality of the NFC-A listening device will be described. A verification plan was set up for measuring functional coverage of the NFC-A listening device based on the capability of the UVC and the NFC-A protocol. Ultimately the goal is to close on 100% functional coverage of the NFC-A listening device based on the verification plan.

5.1 Testbench

Nordic Semiconductor ASA provided a testbench used to verify the NFC feature of their latest chip nRF52. The testbench is written in SystemVerilog and simulates the NFC-A listening device in a chip environment. The top module of the testbench sets up the power and clock domains, NFC, testbench and register interfaces, the NFC-A listening device and a polling device verification module for stimulus production. The testbench is bound to assertions specific for the NFC-A listening device and starts a SystemVerilog program. The program resets the environment, sets up loggers and holds an interface monitor and coverage groups. After the initialization step various test are run to verify the design in terms of noise, oscillator calibration, register access, timing requirements, pattern recognition and field detection.

5.2 Testbench with the UVC

A couple of modification are necessary when the UVC is included in the testbench provided by Nordic Semiconductor ASA. In the top-level module of the testbench the NFC interface, provided with the UVC, is connected to the NFC-A listening device. The NFC interface and the testbench interfaces is made accessible in the UVC by adding the interfaces to the UVM configuration database. The polling device stimulus driver is disconnected from the NFC-A listening device as the UVC provides this service. The program is modified to perform a custom test with the UVC instantiation and the original tests are removed.

The simulation of the testbench is performed using the Nordic Semiconductor ASA standard for digital simulations. The testbench is compiled and run using Questasim 10.e_4c which has a built-in UVM 1.1 library, uvm-1.1d, enabled for the testbench with the UVC.

5.3 Verification Plan

The NFC-A listening device has a comprehensive set of features that should be verified. However, as the UVC is a protocol UVC, focus is to verify that the NFC-A device behaves according to protocol. In order to ensure sufficient functional coverage from the testbench, a test plan was set up in a bottom-up fashion, as described in Chapter 2.3. The bottom-up approach enables extraction of coverage points directly from a detailed specification. The same specifications used to constrain stimulus generation in the UVC is used for writing coverage points. As mentioned in Chapter 3.2, the UVC produces a simplified model of the NFC carrier, thus functional coverage cannot be measured on unmodeled features of the NFC-A protocol. The rest of this chapter will try to answer the questions in Table 2.5 from Chapter 2.3.2 based on a bottom-up approach on the NFC-A protocol specifications [7] and [8].

Which values are important? The carrier amplitude and modulation amplitude are important metrics to verify as a listening device should be able to interpret the patterns from a polling device where these characteristics vary inside the legal ranges. These amplitude characteristics could in a realistic scenario be related to a varying the distance between the polling and the listening device. Similarly, are variations in period and pulse width, and the DUT should handles all these variations. The pulse width of the modulated patterns on the carrier can vary quite a lot and the DUT must handle the variations. However, the pulse width does not vary much inside a frame as a given NFC Device will produce quite constant pulse width for all patterns.

It is also important that a listening device is able to reply with legal response and that the timing associated with the response is according to protocol. Timing between a response and a request, and between two requests, are of less interest as these timing requirements are controlled by the test.

What are the dependencies between the values? The characteristics of the NFC-A protocol is not particularly dependent on each other. However, the period and the pulse width has a dependency, as the pulse width relates to a number of periods of the carrier. This dependency must be accounted for, which is why cross coverage of the two characteristics should be checked.

Timing has a dependency with the last bit of the frame, thus it has a dependency with the command type.

Are there illegal conditions? Illegal conditions include stimulus or responses that reside outside the range of the values specified in Chapter 2.2.

When is the right time to sample? The carrier characteristics are not all valid until the rising edge of the last modulation of a pattern. The right time to sample is once modulation of the pattern has finished.

When is the data invalid? The carrier characteristics are invalid before the last modulation of a pattern and after the bit duration of a pattern has finished.

The conclusion of the verification plan is that each carrier characteristic should be covered by separate coverage points and the dependency between the pulse width and the period should be cross covered. Table 5.1 depicts the planned functional coverage specification. It is important that all the

TABLE 5.1: Functional coverage plan.

Name	Coverage type	Description
Frame layer		
c_poll_to_listen	**coverpoint**	DUT replies with correct FDT$_{A,P}$ timing
c_commands	**coverpoint**	All requests and responses are exercised and the DUT replies with all responses
Pattern layer		
c_pattern_type	**coverpoint**	All NFC-A patterns have been exercised
c_pulse_width	**coverpoint**	Pulse width is exercised at the boundaries of and inside the legal range
c_carrier_amplitude	**coverpoint**	Carrier amplitude is exercised at the boundaries of and inside the legal range
c_modulation_amplitude	**coverpoint**	Modulation amplitude is exercised at the boundaries of and inside the legal range
c_period	**coverpoint**	Period is exercised at the boundaries of and inside the legal range
c_pulse_width_period	**cross**	Cross of the pulse width and period is exercised at its boundary

characteristics are tested at their boundaries and not as important to cover the characteristics inside the ranges. Thus, only one bin is considered sufficient coverage at the two boundaries and inside the boundaries of each characteristic. This gives a total of three bins per coverpoint.

5.4 Control Unit for DUT

A control unit has been set up a connected to the DUT as the DUT must be told how to respond to request. In hardware, this is typically done by a higher level module inside the NFC-A peripheral. For this particular testbench a limited control unit is designed to read the request, picked up by the DUT, and tell the DUT how to respond to that request. It is a simple control unit based on a subset of the state diagram of a listening device as illustrated in NFC Forum Activity Specification [18, p. 119]. The operation of the FSM in the control unit is depicted in Figure 5.1. The control unit does not have to determine when

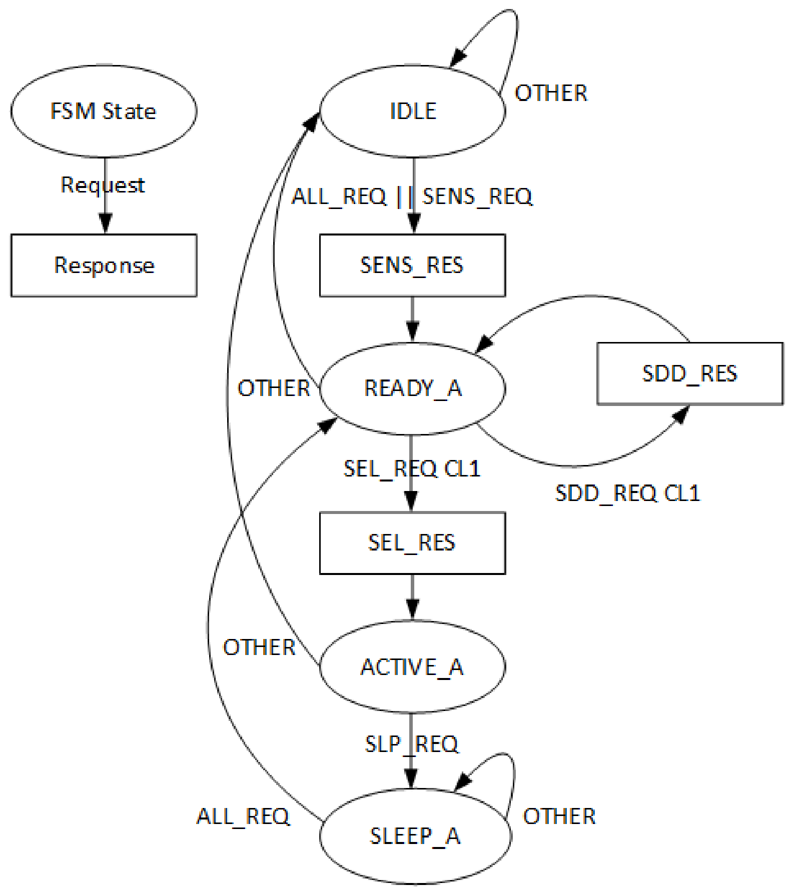


FIGURE 5.1: FSM of the NFC-A listening device control unit.

the response is sent as this is handled by the DUT. However, it makes the DUT respond according to protocol which is necessary in order to do timing analysis between frames. Different responses have different timings associated with them, thus information about the nature of the frame is necessary in order to compare the DUT timing against correct timing requirements.

5.5 Configuration of UVC

The configuration of the UVC is based on the goal of the test. The DUT is a listening device, thus a NFC carrier signal has to be provided by the test. Hence, the UVC is configured with a polling device agent. To verify that the DUT is able to respond correctly to requests the UVC is configured with an interface monitor and a scoreboard. The configuration includes a framing component to provide frame content from the monitor to the scoreboard. Analysis of pattern layer requirements is performed by the monitor and timing requirements are performed by the scoreboard. A coverage collector is configured in the UVC to collect pattern and frame layer coverage from the interface monitor and the scoreboard. The coverage collector records coverage based on the functional coverage specification in Table 5.1 from Chapter 5.3. Figure 5.2 depicts the components of the UVC once configured for the testbench.

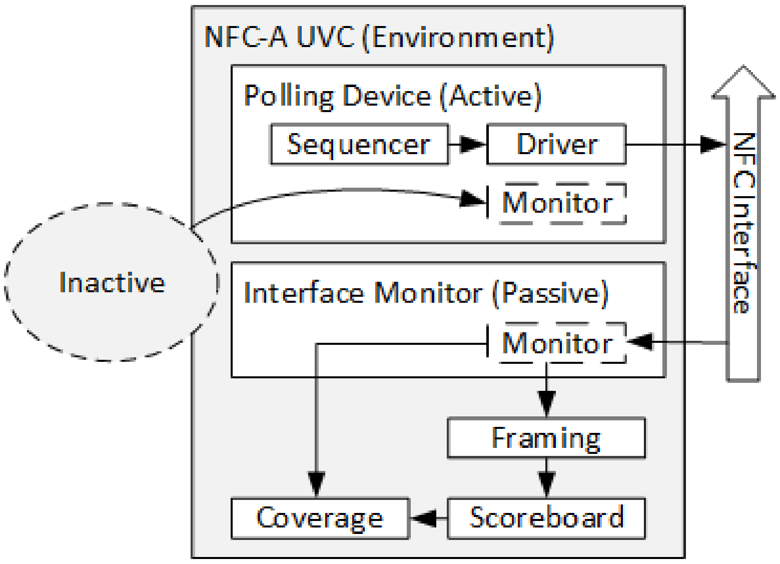


FIGURE 5.2: Testbench configuration of the UVC.

5.6 Test

The test extends a test base class which performs the necessary setup for any testbench using the UVC with the sequence library. The test base instantiates and configures the UVC as described in Chapter 5.5. It instantiates the control unit and connects the serial interface of the DUT to the control unit. A sequence item is declared for the test and added to the configuration database which is necessary in order to generate stimulus with the sequence library.

The test class holds only the description of stimulus generation. Before driving actual stimulus, an initialization routine is performed. This sets up the carrier, indicating the presence of a NFC carrier. Secondly, a sequence from the sequence library is performed until 100% functional coverage of the frame layer coverage points described in Table 5.1 from Chapter 5.3 is reached. This sequence traverses the FSM in Figure 5.1. Finally, another sequence from the sequence library is performed until 100% functional coverage of the pattern layer is reached.

Figure 5.3 illustrates the components of the test. The sequences from the sequence library are referred to as worker sequences.

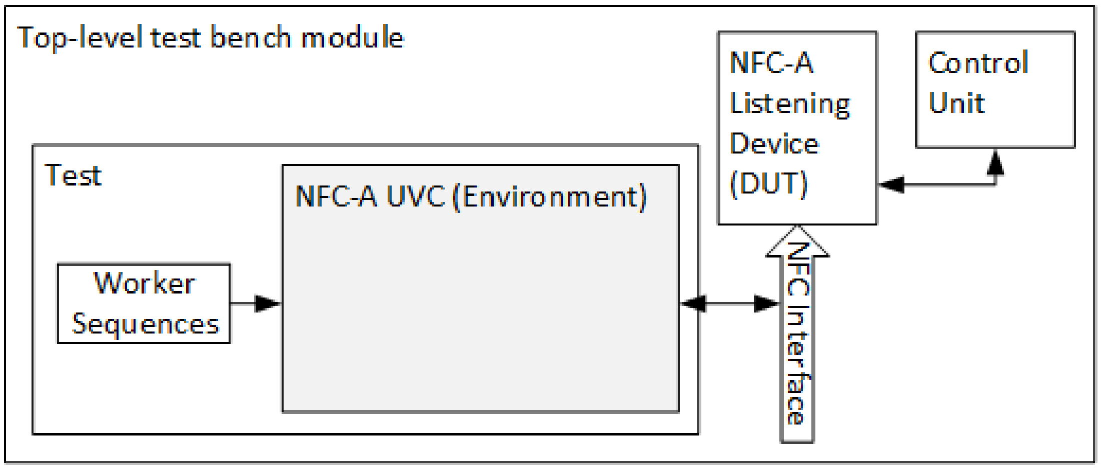


FIGURE 5.3: NFC-A listening device testbench with the UVC.

Chapter 6

Results

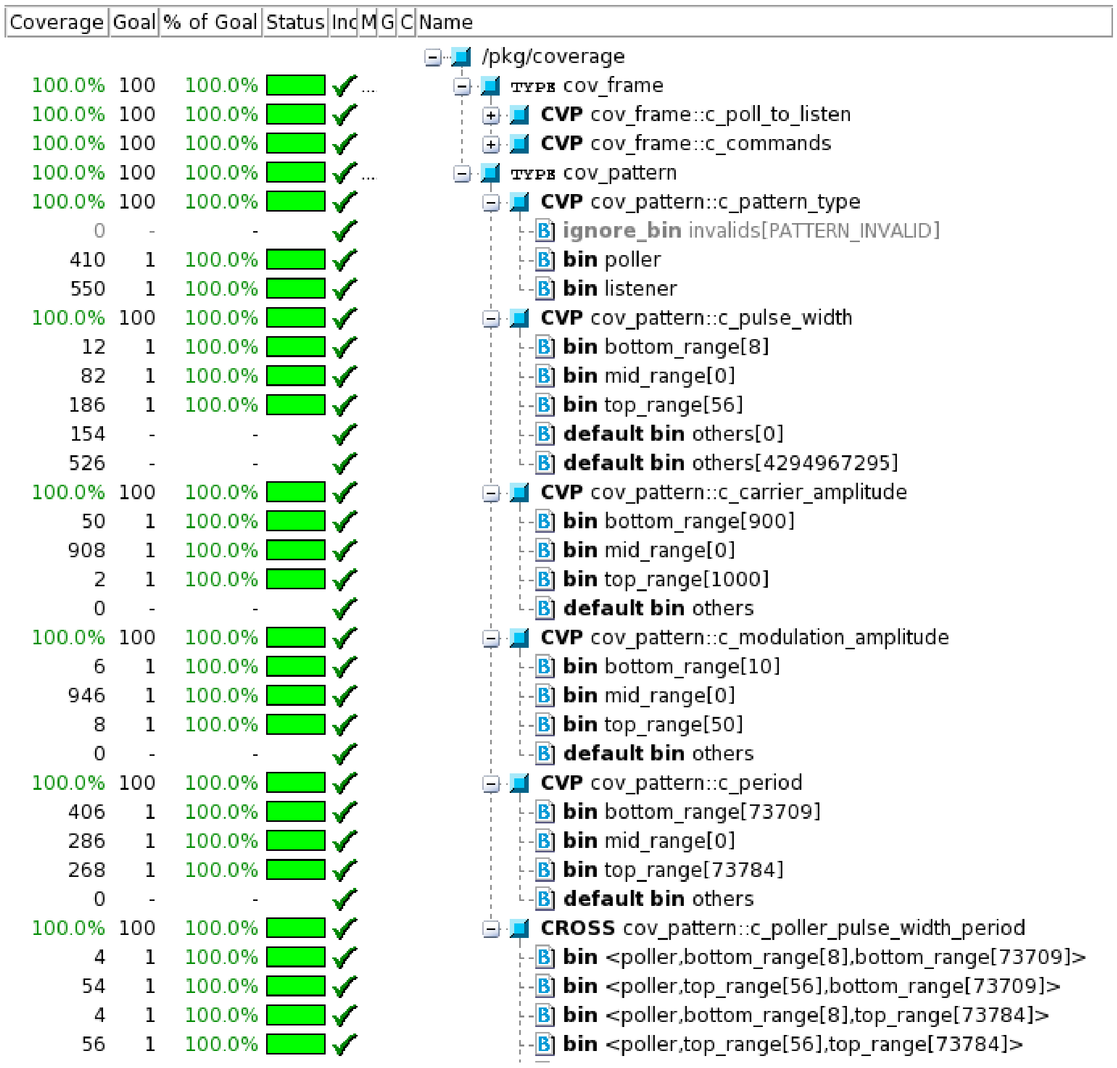
The testbench finished reaching a 100% closure on functional coverage at a simulation time of 55 seconds. Table 6.1 shows the number of passed and failed request sent by the UVC and responses sent by the DUT. Passed requests

TABLE 6.1: Requests and responses monitored during simulation.

Request	**Instances**	**Passed**	**Failed**
ALL_REQ	9	9	0
SENS_REQ	1	1	0
SDD_REQ_CL1	1	1	0
SEL_REQ_CL1	1	1	0
Response	**Instances**	**Passed**	**Failed**
SENS_RES	10	10	0
SDD_RES	1	1	0
SEL_RES	1	1	0

indicate that legal stimulus was exercised and passed responses indicate that the DUT is able to reply according to protocol. The scoreboard confirms that the DUT replies to request within the legal timing requirements of the protocol by reporting no errors.

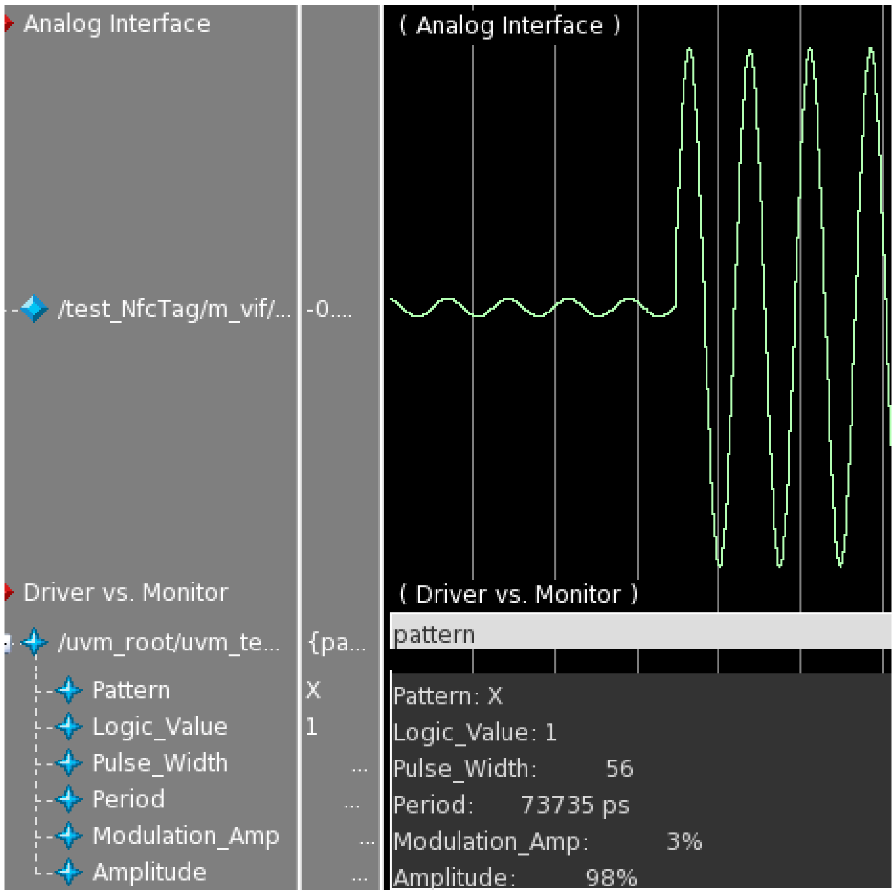
Figure 6.1 shows the coverage report from the Questa simulation tool confirming 100% functional coverage.



Coverage	Goal	% of Goal	Status	Inc	M	G	C	Name
								/pkg/coverage
100.0%	100	100.0%	🟩 ✓	...				TYPE cov_frame
100.0%	100	100.0%	🟩 ✓					**CVP** cov_frame::c_poll_to_listen
100.0%	100	100.0%	🟩 ✓					**CVP** cov_frame::c_commands
100.0%	100	100.0%	🟩 ✓	...				TYPE cov_pattern
100.0%	100	100.0%	🟩 ✓					**CVP** cov_pattern::c_pattern_type
0	-	-	✓					**ignore_bin** invalids[PATTERN_INVALID]
410	1	100.0%	🟩 ✓					**bin** poller
550	1	100.0%	🟩 ✓					**bin** listener
100.0%	100	100.0%	🟩 ✓					**CVP** cov_pattern::c_pulse_width
12	1	100.0%	🟩 ✓					**bin** bottom_range[8]
82	1	100.0%	🟩 ✓					**bin** mid_range[0]
186	1	100.0%	🟩 ✓					**bin** top_range[56]
154	-	-	✓					**default bin** others[0]
526	-	-	✓					**default bin** others[4294967295]
100.0%	100	100.0%	🟩 ✓					**CVP** cov_pattern::c_carrier_amplitude
50	1	100.0%	🟩 ✓					**bin** bottom_range[900]
908	1	100.0%	🟩 ✓					**bin** mid_range[0]
2	1	100.0%	🟩 ✓					**bin** top_range[1000]
0	-	-	✓					**default bin** others
100.0%	100	100.0%	🟩 ✓					**CVP** cov_pattern::c_modulation_amplitude
6	1	100.0%	🟩 ✓					**bin** bottom_range[10]
946	1	100.0%	🟩 ✓					**bin** mid_range[0]
8	1	100.0%	🟩 ✓					**bin** top_range[50]
0	-	-	✓					**default bin** others
100.0%	100	100.0%	🟩 ✓					**CVP** cov_pattern::c_period
406	1	100.0%	🟩 ✓					**bin** bottom_range[73709]
286	1	100.0%	🟩 ✓					**bin** mid_range[0]
268	1	100.0%	🟩 ✓					**bin** top_range[73784]
0	-	-	✓					**default bin** others
100.0%	100	100.0%	🟩 ✓					**CROSS** cov_pattern::c_poller_pulse_width_period
4	1	100.0%	🟩 ✓					**bin** <poller,bottom_range[8],bottom_range[73709]>
54	1	100.0%	🟩 ✓					**bin** <poller,top_range[56],bottom_range[73709]>
4	1	100.0%	🟩 ✓					**bin** <poller,bottom_range[8],top_range[73784]>
56	1	100.0%	🟩 ✓					**bin** <poller,top_range[56],top_range[73784]>

FIGURE 6.1: Functional coverage result from Questa simulation tool.

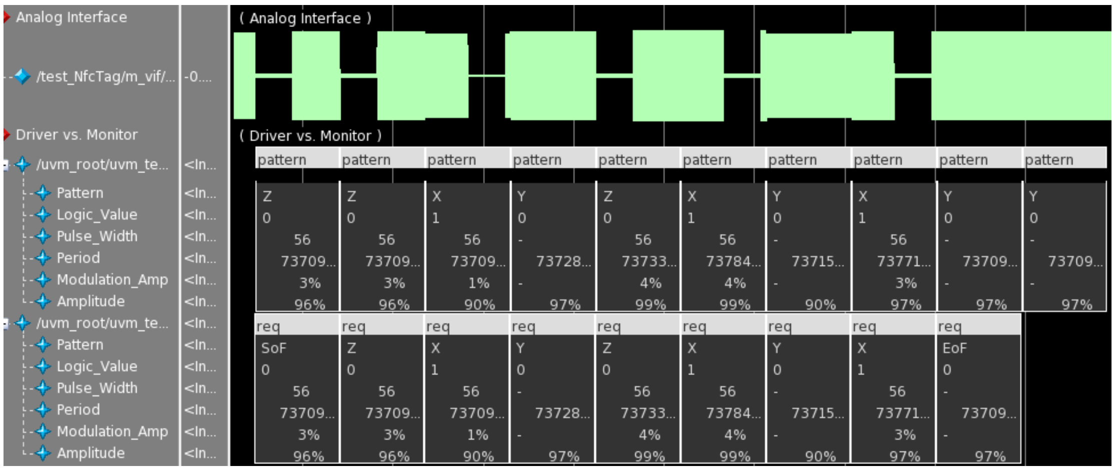
Figure 6.2 illustrate the discrete analog model driven by the UVC depicting a NFC-A pattern X with its associated analog characteristics. Note that Modulation_Amp and Amplitude, indicating the carrier amplitude with and without modulation, respectively, indicates the percentage of a normalized carrier amplitude of 1.



```
Pattern: X
Logic_Value: 1
Pulse_Width:        56
Period:      73735 ps
Modulation_Amp:         3%
Amplitude:         98%
```

FIGURE 6.2: Discrete analog model of NFC carrier driven by the UVC.

Visual inspection of frames confirms that exercised requests and responses are legal according to protocol. Figure 6.3 illustrates the waveform of an exercised ALL_REQ request. The pattern and req boxes holds the characteristics of the monitored and driven stimulus, respectively. Thus, Figure 6.3 confirms that the UVC is able to monitor correct stimulus, by monitoring the exact driven stimulus.



/uvm_root/uvm_te... <In...

	pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern	pattern
Pattern	Z	Z	X	Y	Z	X	Y	X	Y	Y
Logic_Value	0	0	1	0	0	1	0	1	0	0
Pulse_Width	56	56	56	-	56	56	-	56	-	-
Period	73709...	73709...	73709...	73728...	73733...	73784...	73715...	73771...	73709...	73709...
Modulation_Amp	3%	3%	1%	-	4%	4%	-	3%	-	-
Amplitude	96%	96%	90%	97%	99%	99%	90%	97%	97%	97%

/uvm_root/uvm_te... <In...

	req	req	req	req	req	req	req	req	req
Pattern	SoF	Z	X	Y	Z	X	Y	X	EoF
Logic_Value	0	0	1	0	0	1	0	1	0
Pulse_Width	56	56	56	-	56	56	-	56	-
Period	73709...	73709...	73709...	73728...	73733...	73784...	73715...	73771...	73709...
Modulation_Amp	3%	3%	1%	-	4%	4%	-	3%	-
Amplitude	96%	96%	90%	97%	99%	99%	90%	97%	97%

FIGURE 6.3: ALL_REQ request on the NFC interface.

Chapter 7

Analysis and Discussion

Reusability through modular components is one of the benefits of UVM. The components object-oriented nature makes them easily extendable which enhance the reuse of code. UVM provides a convenient configuration mechanism that allows configuration of components both before and during simulation. It is a simple mechanism based on a database that can be accessed from anywhere in a component hierarchy. The UVM factory is another beneficial mechanism where component definitions are stored. This contributes to the overall reusability of the UVM environment as component definitions can be changed in different test without changing the underlying code base.

When designing the UVC it was preferable to register all the components in the factory. The sequence item is often a part of the component definition and by registering the components to the factory they are automatically re-defined if the sequence item is modified or changed. The hierarchical components, environment and agent, dealing with configuration of child components, should obtain configuration from the configuration database. The configuration of these components must be set in advanced, however they could be changed dynamically. Whether the configuration is stored as a separate object in the configuration database or it is set as parameters in a given test is the designer choice. Different approaches suit different environments better, but if no guidelines for the project is set or the design not too comprehensive a separate configuration object could be advisable. A configuration object has the convenient property of gathering the configuration in one location and can keep default configurations.

As the UVC is an interface UVC, the long-term goal of the design is to cover the complete NFC-A protocol and even the complete NFC protocol. Due to the limited time budget of this thesis the focus is put on the NFC-A protocol. Complex parts of the NFC-A protocol, like collision resolution, is also considered to be out of the scope of this thesis. However, the design is designed with this long-term goal in mind and facilitates for extended functionality.

Collision resolution is a part of the NFC-A protocol which only takes effect when multiple listening devices tries to communicate with the same polling device. The immediate goal of this thesis is the verification of only one listening device, thus the UVC is only configured with a polling device agent. However, the environment can be configured with multiple polling devices and/or listening devices when it becomes necessary.

7.1 Analog Interface Model

When dealing with analog modeling in a digital environment it will eventually result in loss of information in the model. The timescale of the simulation might give sufficient sampling to achieve the wanted results, but information is lost between each sample. Analog models often have complex characteristics hard to recreate with a simple digital model. In order to keep the digital model as simple as possible, while still being able to model enough information of the analog model, the most complex characteristics could be neglected.

The carrier of the NFC-A protocol allows a few complex characteristics such as overshoots. Overshoots the rising edge of the modulation, provide no unique high-level protocol information and are mathematically complex. Removing the overshoots from the digital model will remove a possibly interesting analog characteristic, but will allow the same high-level protocol modeling and a much simpler digital model.

At the start and end of modulation a certain amount of time will be related to the transition of the carrier amplitude which could be associated with rise and fall times.

7.2 Components of the UVC

The protocol coverage of the UVC have been improved and extended during this thesis. Capabilities for both driving and monitoring of listening device stimulus have been added and the UVC is able to perform verification of protocol behavior of listening devices. The monitor implementation has a new and more readable design and synchronization issues has been removed due to less communication between internal tasks. The UVC is now able to monitor the frequency of the interface without compromising simulation speed. Configuration objects makes the UVC configurable for different testbenches which improves reusability.

In this chapter discussion related to specific components of the UVC will be presented.

7.2.1 Sequence Item

The sequence item designed for the UVC enables the required communication between components and encapsulates all the information needed to create the interface analog model. However, the disadvantage of the sequence item representation is that representing complex analog behavior inside a NFC-A pattern is a challenge. Complex analog behavior such as overshoots and rise and fall times should be considered for future work of the analog model, but it is not straight forward to implement this behavior with the current sequence item design. A possible solution could be to add data members that holds information about the range of the overshoots, rise and fall times. This would require changes to the driving algorithm of the driver component as well. Considerations regarding improving the analog behavior, work load, reuse of components and facilitation of additional functionality should be considered in the design of a new sequence item. Alternative sequence abstractions should be considered in order to find the best representation. The sequence item considered in Chapter 3.4.1, representing a period of the NFC-A carrier, could give a finer grained stimulus control which might be preferable for complex analog modeling. One challenge with representing a period of the carrier is monitoring. A period of the carrier does not have enough information and consequently the monitor will have to analyze the periods of a pattern. A solution might consider using the principal of layering. A solution with a layered monitor could monitor periods and broadcast the period items to the second layer of the monitor. The second monitor could perform analysis of the pattern layer of the protocol.

7.2.2 Driver

The driver and is not designed to be reusable. However, by extending the interface with methods for writing the driver could be designed reusable by using calls to this method. This requires little work and would basically require the driving routine to be put in the write method of the interface. This could be an interesting point for future work concerning the driver.

By descretizising an analog signal in the digital domain, information will be lost. The sampling frequency of the signal is important in how well the signal is represented and a minimum requirement for sampling a signal is given by the Nyquist Rate [19]. Nyquist state that the sampling frequency of a signal must be equal or greater than twice the frequency of the sampled signal. A simulation timescale of 1 ns will result in a sampling rate of 1 GHz which is well above the minimum sampling frequency of the 13.56 Mhz NFC frequency. However, other unmodeled features of the carrier, such as overshoots and rise and fall times, leads to loss of information.

The design depicted by UVM will be followed in this thesis for the generation and driving of stimulus. The driving algorithms are implemented in the driver as the UVC is one of a kind in an otherwise traditional SystemVerilog testbench. In a larger UVM based testbench, typically a large top-level testbench, it could be wise to design a more generic driver component and have the driving algorithm inside the interface implementation. In such a case, it would be more beneficial with a reusable component.

7.2.3 Monitor

As well as driver, the monitor is not design to be reusable. However, it could similarly be designed reusable by using an interface read method.

Noise is not taken account for by the monitor. In a particular noisy environment any distortion of the carrier signal will not be filtered out as noise, but will be considered a violation of the protocol [8, p. 40]. The NFC-A protocol states that during parts of the time between transmission of frames the polling and listening devices are not allowed to produce any detectable disturbance.

This will be ignored by the monitor, thus it is not able to verify this part of the protocol.

The monitor implementation is specific for the NFC-A protocol and extending the functionality to cover the NFC-B and NFC-F protocol will possibly require completely new structure of the component.

Collision resolution, which is a necessary part of the NFC-A protocol when several listening devices are communicating with a polling device, is not supported by the monitor. Even though the UVC enables configuration of a network of listening devices, it will require collision free communication between the devices. This requirement will lead to a less realistic and interesting test scenario.

7.2.4 Agent

UVM dictates two configurations of the agent, an active and a passive agent, as mention in Chapter 2.1.5. However, in the design of the UVC an extra feature for disabling the monitor of an agent was designed. This is not mention in the guidelines of UVM, but it in a scenario where the UVC contains many agents, thus many monitor, this feature could decrease the consumption of simulations cycles by disabling unnecessary monitor. This is typically suitable for interface UVCs with an interface monitor. In this situation it is most likely sufficient one monitor surveying the interface for protocol behavior. Adding this feature does not compromise the reusability of the agent, thus the agent is still completely reusable in any UVC.

7.2.5 Framing

Specific for the UVC is the framing component which works as a frame content provider for analysis components. Extracting the content of a NFC-A frame is a necessary step for any analysis component that deal with verification of frames. Thus, code reuse of this step is increased as analysis component can either extend or receive frame content from this component.

Analysis components are some of the most important component in a verification IP like the UVC. As mentioned in Chapter 3.1 and 3.3 the UVC is implemented with a framing component. One of the arguments for having this component is to facilitate for new analysis components that might become necessary if new parts of the protocol should be covered. One example where this component's potential is useful is if the UVC should implement the NFC-B and NFC-F protocol.

7.2.6 Scoreboard

A scoreboard is often very specific to a testbench, but UVM enables this component to be reusable. The sequence items are implemented with copy and compare functions which could be used by the scoreboard. Thus, the scoreboard becomes independent of the testbench as compare functionality reside in the sequence item. The proposed design in [17, p. 13] is a very interesting design which should be recommended for future work. Reusability, through modular components, are the main advantage of this design.

7.2.7 Environment

Using the principle of reusability has been a major focus throughout the design of the UVC. The environment class of the UVC refers to the active agents as masters and slaves and generates the number of these components as a configurable number. This makes the implementation less dependent of the protocol, thus it can be used in verification of other master/slave protocols as well. In addition, it enables the UVC to be configured with multiple masters and/or slaves to create a potentially more realistic verification environment. One such example is the NFC where more than one listening device might connect to the polling device.

However, the environment is a component which potentially have different demands on required functionality from testbench to testbench, thus is potentially more difficult to design for reuse. As described in Chapter 2.1.4 the environment can hold a range of components which of course is not necessarily needed in any given testbench. Consequently, when designing the environment, design features that could enhance reusability of the environment was extracted from the purpose of the UVC. The intention of the UVC is to perform verification of the NFC-A protocol which is based on a master/slave relationship between its communicators. This is typical for other protocols as well. By designing the environment with an array, containing master and slave agents, the environment is made reusable for similar protocol testbenches with different demands on number of master and slave agents. A coupled configuration object is designed to hold variables that can be set to enable different configurations of the environment.

7.3 Sequence Library

Reusability has been a major focus when creating the request library. By extending new sequences from a base sequence, which holds the instantiations of the patterns necessary for a polling device request, the sequences become structurally simple and readable. UVM sequences are very well suited for CRV. UVM depicts sequences to be extended from smaller sequences to create hierarchical sequences. This increase reuse of code and is readable. Perhaps the most powerful feature of the UVM sequence is that a hierarchical sequence can be constrained from any level in its hierarchy and the constraints will be passed down to the lowest level. Thus, it is easy to aim stimulus to hit interesting scenarios.

Some of the sequences in the request library use the UVM defined macros for randomizing and executing sequences and sequence items. This provides a readable format of the sequences which has been the motivation behind the use of the macros. However, the macros include, among other, a pre and post-randomize step that is not used. The consequence is excess code evaluation that only slows down the simulation. The macros could be replaced by choosing the required functionality that comprise the macro, but on the compromise of readability. Depending on the amount of calls to the macros the cost in simulation time will increase, however it is such a small amount of extra code that the trade-off between readability versus faster simulation favors readability. The base sequence does not use the UVM macros but provide a similar abstraction trough function. The functions are designed to increase reusability and readability when creating higher level sequences.

Some of the sequences represents potential parts of the collision resolution, but in a simple setup with one polling and one listening device the definition of these sequences become constant. When the testbench increase in complexity and collision resolution must be implemented, it can be implemented with minimum effort.

7.4 Test and Results

The main focus of the testbench set up with the UVC is not to verify the DUT, but to confirm that the UVC is able to perform protocol verification of a NFC-A listening device. This focus makes the UVC testbench difficult to compare quantitatively with the NFC-A listening device testbench provided by Nordic Semiconductor ASA. As mentioned in Chapter 5.1, the testbench provided by Nordic Semiconductor ASA test a lot more functionality of the DUT than the UVC testbench. However, the UVC testbench performs a more complete verification of the protocol behavior of the DUT. In addition, the UVC testbench is able to verify the analog characteristics of the interface which it provided testbench is not capable of.

The stimulus of the test performed by the testbench targets the functionality of the protocol and the characteristics of the interface. The test exploits the principal of MDV by first running stimulus targeting high-level protocol features such as frames and timing between frames. Once sufficient coverage is reached at this level of the protocol, the test aims the stimulus hit the characteristics of the interface. This method decreases the potential stimulus scenarios, yielding greater simulation speed. Constraining the stimulus is main technique used to aim the stimulus at specific parts of the stimulus space. The boundaries are described, in Chapter 5.3, as the most important verification criterion of the analog characteristics. The reason behind this criterion is that the DUT measures the analog characteristics by level and timing sensitive circuits. Thus, if the DUT handles the characteristics at the boundaries, it is reasonable to assume that the DUT handles the characteristics inside the boundaries.

The results suggest that the UVC is able to perform verification of a NFC-A listening device. 100% functional coverage is reached much faster than is possible in a SPICE simulation. This is due to the use of digital simulation solvers and using a MDV technique on a RVM model. Code coverage is not considered important as the testbench targets the protocol behavior of the DUT, not the functionality of the complete DUT. Otherwise, the it would be important with 100% code coverage to ensure that the DUT's functionality was fully tested.

Figure 6.2 and 6.3 indicate that the UVC models the interface according to the analog model described in Chapter 3.2.

7.5 RVM

SPICE simulations are the most used type of simulation when verifying an analog IP. The benefit of SPICE simulation is the accuracy with which it can verify a circuit, however the performance of the simulation is compromised by its accuracy. This problem becomes apparent if the circuit is large and complex which could result in a long tedious simulation if all functionality is to be tested. The time-to-market requirement of the VLSI business will most likely make it impossible to verify all the functionality of such analog circuits with SPICE simulation accuracy. Alternatives to the SPICE simulations exists on the market, such as AMS and RVM simulations, however they tend to compromise the accuracy of the simulations to gain performance. AMS is the most accurate simulation type of the two and uses both digital and analog solvers in the simulation of a design. Unfortunately, this leads switching activity between the solvers based on when a digital or an analog portion of the circuit is simulated. This in addition to using the analog solvers decrease the performance of the simulation, though it is faster than SPICE simulations. RVM trades even more of the accuracy for more performance by using only the digital solvers. This requires discrete analog models of the analog IPs which significantly degrades accuracy. Not only is the convergence analysis of the circuit lost, due to not using SPICE, but even more information is lost due to the discretization of analog behavior. The importance of such circuit analysis cannot be mitigate for by digital solvers and consequently RVM simulation cannot replace SPICE simulations. However, RVM simulation can still be beneficial in the verification of mixed-signal circuits. Especially for large analog IPs a discrete model simulated in a RVM simulation could be used increase functional verification of the IP. As well as for large analog IPs, even for small analog IPs, RVM simulations is useful in top-level testbenches of a chip. It allows functional verification of the complete cooperation of the chip. A less attractive alternative is black-boxing of analog IPs in the top-level simulation and only perform verification of the interconnects between the digital and analog circuits. For RVM simulation to be beneficial it is important to notice that the testbench can only be as good as the model of the analog circuit itself. Thus, a good model that allows verification of the intended functionality is important.

7.6 UVM

UVM has a lot to offer in terms of digital simulation. Based on the SystemVerilog language it has the ability to create constrained random variables in an OOP environment. CRV is not only a benefit found in UVM, but CRV is very powerful when the stimulus space is large. The time and effort related to creating stimulus is greatly reduced, compared to directed tests, by letting the simulation solver create the stimulus inside given constraints. Thus, providing better control in the process of ensuring that all possible stimulus scenarios within the legal stimulus space is hit. Reusability based on modular components is one of the key features of UVM which is a structure UVM itself is based on. The larger the testbench, the more this benefit becomes significant and the UVC can be designed to be almost completely reusable. There is a cost related to learning UVM and setting up a UVM environment, however once it is set up for one IP it takes less effort setting up another. From a design and/or verification engineer's point of view the methodology provides a fixed structure of the verification modules which makes it easier for the engineer to migrate to other UVM testbenches. An interesting aspect of UVM is the notion of a developer and a user of UVCs. What this means is that professional UVM developers can set up the testbench environment and function as the moderator of the simulation. However, the user does not have to concern with this issue as much as only defining stimulus sequences and including the UVCs in the testbench. This allows the user to focus on the actual problem of defining good stimulus and in turn spend less time verifying the design.

UVM is a dynamic simulation environment that allows testbench to change tests during simulation. In an MDV simulation this is beneficial as certain tests can be stopped based on coverage metrics and other test can be initiated to hit other parts of the design functionality. The configuration database in UVM allows UVCs to easily be configured to suit different testbenches which can be a time consuming process.

As this thesis shows, UVM is not only suited for the traditional digital designs, but also for RVM simulations. Using UVM on analog behavior models enables the all the benefits of UVM in this type of simulation. However, analog behavior models are typically level sensitive and often requires floating point precision. The SystemVerilog real data type provides this precision, but it is not possible to randomize such variables opposed to integers and other similar data types. The solution is to create a model for converting a random variable into a real data type representation. This works well and is a technique used in the UVC where the frequency of the NFC carrier signal is based on a randomized period represented as an integer.

As the RVM simulation is used to mitigate for SPICE simulation performance the focus of the simulation becomes functionality. UVM sequences allows functionality to be exercised in a convenient are readable manner.

Chapter 8

Conclusion

The work of this thesis presents the successful design and implementation of a UVC for the NFC-A protocol. The UVC has been simulated in a testbench with a NFC device and was able to close on 100% functional verification of the device according to a verification plan. Reusability has been one of the main focuses in the creation of the UVC, thus components of the UVC can be reused for other UVC projects. Parts of the NFC-A protocol that is not covered by the UVC could be implemented with minimal effort due to the focus on facilitation of additional functionality.

The UVC illustrates how UVM can be used with RVM to achieve digital simulations speeds in functional verification of analog IPs.

A sequence library for stimulus generation comes with the UVC to provide users of the UVC with examples and a starting point for creating relevant testbench stimulus.

8.1 Recommendations for Future Work

- Design the driver and the monitor components reusable by replacing the algorithms with calls to interface methods.
- Extend the monitor component with the ability to detect noise on the interface.
- Extend the stimulus library with responses for listening devices.
- Create more reusable scoreboard design.
- Extend the driving and monitoring algorithms with the ability to perform collision resolution of the NFC protocol.
- Extend the UVC to cover verification of the complete NFC protocol.

Bibliography

[1] Dave Evans. "The Internet of Things How the Next Evolution of the Internet Is Changing Everything". In: (Apr. 2011).

[2] Sathishkumar Balasubramanian and Pete Hardee. "Solutions for Mixed-Signal SoC Verification Using Real Number Models". In: (2013).

[3] Christoffer Ramstad-Evensen. "Develop a UVM based Verification IP for an Analog RF Protocol". 2015.

[4] K. A Meade and s. Rosenberg. *A Pratical Guide to Adopting the Universal Verification Methodology (UVM) Second Edition*. Cadence Design Systems, Inc, 2013, pp. 1–2,13–15, 257.

[5] Accelera. "Universal Verification Methodology (UVM) 1.2 User's Guide". In: (Oct. 2015), pp. 1–5, 29, 144–154, 174.

[6] NFC Forum. *What Is NFC - What It Does*. 2015. URL: http://nfc-forum.org/what-is-nfc/what-it-does/.

[7] NFC Forum. "NFC Analog Specification Analog 1.0". In: (July 2012), pp. 33–34, 40, 52.

[8] NFC Forum. "NFC Digital Protocol Version 1.1". In: (May 2014), pp. 12,15–20, 37–43.

[9] Mentor Graphics Verification Academy. "Coverage Cookbook". In: (May 2016). URL: https://verificationacademy.com/cookbook/coverage.

[10] Yaron Kashai Neyaz Khan and Hao Fang. "Metric Driven Verification of Mixed-Signal Designs". In: (2011).

[11] Lukasz Kotynia Muralikrishna Sathyamurthy Felix Neumann and Eckhard Hennig. "UVM-based verification methodology for RFID-enabled smart-sensor system". In: (Jan. 2013).

[12] "www.testbench.in". In: (). URL: www.testbench.in.

[13] Jeffrey Fan Xiaokun Yang Xinwei Niu and Chiu Choi. "Mixed-Signal System-on-a-Chip (SoC) Verification Based on SystemVerilog Model". In: (Mar. 2013).

[14] Cristian Ţepuş Ioana Iliuţă. "Constraint random stimuli and functional coverage on mixed signal verification". In: (2014).

[15] Pallavi Das. "Exploration of Real Value Modelling for Complex Mixed Signal Verification". MA thesis. Indraprastha Institute of Information Technology New Delhi, June 2015.

[16] Hubert Zimmermann. "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection". In: (1980).

[17] Clifford E. Cummings. "OVM/UVM Scoreboards - Fundamental Architectures". In: (2013).

[18] NFC Forum. "Activity Version 1.1". In: (Mar. 2014), p. 119.

[19] H. J. Landau. "Sampling, Data Transmission, and the Nyquist Rate". In: (Oct. 1967).