On the Verification of a WiMax Design Using Symbolic Simulation

Salim Ismail Al-Akhras

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical & Computer Engineering)

at

Concordia University

Montréal, Québec, Canada

April 2012

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Salim Ismail Al-Akhras**

Entitled: **On the Verification of a WiMax Design Using Symbolic Simulation**

and submitted in partial fulfilment of the requirements for the degree of

Master of Applied Science (Electrical & Computer Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. Samar Abdi (ECE)

_____ Dr. Nizar Bouguila (CIISE)

_____ Dr. Gabriela Nicolescu (École Polytechnique de Montréal)

_____ Dr. Sofiène Tahar

Approved by _____

Chair of the ECE Department

_____ 2012 _____

Dean of Engineering

ABSTRACT

On the Verification of a WiMax Design Using Symbolic Simulation

Salim Ismail Al-Akhras

The system-On-Chip design process is continuously increasing in terms of cost and complexity. This imposes new modeling and verification challenges. A particular example is heavy computational applications and functionality, such as digital signal processing and telecommunication applications, which are increasingly integrated in embedded systems nowadays. To meet these challenges, designers use a multilevel model based approach, which is a top-down design methodology where the behavior of the system is first modeled at a higher level of abstraction. Then, design decisions are made to refine those models in a number of transformations until the final product is realized. In this thesis we verify an implementation of a WiMax modem physical layer that has been designed according to the multilevel design approach. This implementation is provided by STMicroelectronics. We propose the utilization of two verification methodologies to verify designs at higher levels of abstraction. The first one is an equivalence checking methodology that is based on symbolic simulation, which provides high speed and computational capabilities. The main purpose of this methodology is to verify the functional equivalence of refined system models in the design process. The second methodology is a property checking approach, which is also based on symbolic simulation. It verifies the conformance of models at different levels of abstraction with the system specification. We verified the equivalence of three models of the WiMax system at different levels of abstraction, and we verified the correctness of various system properties on those models.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my great thanks to the Almighty God, Who gave me the strength and patience to complete this work.

I would like to extend my sincerest gratitude for my supervisor Dr. Sofiène Tahar, who has been a constant source of thoughtful guidance in pursuing this work. Because of his input, advice, and challenge, I have matured as a researcher and as a graduate student. I am very thankful for the verification team at STMicroelectronics Montreal Canada and Dr. Gabriela Nicolescu from École Polytechnique de Montréal for giving me an opportunity to do my Master's project in collaboration with STMicroelectronics Canada in Ottawa. I would also like to acknowledge my thesis committee: Dr. Samar Abdi and Dr. Nizar Bouguila for their valuable feedback on the thesis.

I would also like to take this opportunity to express my sincere thanks to my colleagues in the Hardware Verification Group (HVG) of Concordia University for their motivation and constructive suggestions. In particular, I would like to thank Dr. Naeem Abbasi for his time and valuable help during my thesis writing.

During my stay in Montreal, I became acquainted with some great people who shared with me both difficult and easy times throughout this project. I would like to thank my friends, Maher Alhalabi, Suliman Albashir, Hani and Haitham Altelbani, for their huge support and motivation. My brother and very special friend Zaid Abdulhadi whom I relied on for many matters deserve also great thanks for those years of friendship.

I am very very grateful for my wife, Hanaa, for my sisters, Rasha, Rola and Ola and for my brothers, Mohammad and Ahmad for the support and happiness they always provide me with. Finally, my sincere thanks and deepest appreciation go out to my parents, Khawla and Ismail for their affection, love, support, encouragement, and prayers to success in my missions.

This thesis is dedicated to

My Father

Ismail Alakhras

My Mother

Khawla Almasri

My Wife

Hanaa Almasri

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AMS	Analog and Mixed Signal
AP	Access Point
API	Application Programming Interface
ARQ	Automatic Repeat Request
BDD	Binary Decision Diagram
BS	Base Station
CC	Convolution Coding
DAMA	Demand Assigned Multiple Access
DSL	Digital Subscriber Line
DSP	Digital Signal Processing
DUT	Design Under Test
FDD	Frequency Division Duplexing
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIFO	First In First Out Structure
FPGA	Field-Programmable Gate Array
HARQ	Hybrid Automatic Repeat Request
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
ILP	Instruction Level Parallelism
LOS	Line of Sight
MA	Model Algebra
MAC	Medium Access Control Layer
MAN	Metropolitan Area Networks
NLOS	Non-Line of Sight
OO-VHDL	Object Oriented VHDL

OFDMA	Orthogonal Frequency-Division Multiple Access
PDU	Protocol Data unit
PDU	Protocol Data Units
PSL	Property Specification Language
PHY	Physical Layer
PSL	Property Specification Language
PTL	Process Transfer Level
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
RS	Reed Solomon
RTL	Register Transfer Level
SLDL	System Level Description Language
SC	Single Carrier Modulation
SoC	System On Chip
SRE	Sequence Recurrence Equations
SS	Subscriber Stations
ST	STMicroelectronics
SW	Software
SWSR	Single Write Single Read
TC	Transmission Convergence
TDD	Time Division Duplexing
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TLM	Transaction Level Modeling
UL-MAP	Uplink Map
VHDL	VHSIC Hardware Description Language

VHSIC	Very High-Speed Integrated Circuit
WEP	Wireless Equivalent Privacy
WiMax	Worldwide Interoperability for Microwave Access
WPA	WI-FI Protected Access

Chapter 1

Introduction

1.1 Motivation

In today's electronics technology, a single chip can accommodate a large and complex system that has a wide variety of components and functionalities; called System-On-Chip (SoC). SoCs have a wide range of applications in consumer electronics and embedded systems like telecommunication devices. A single SoC can contain more than 10 million gates which make their design a very complicated and costly process. As a result, it becomes increasingly complex to identify design bugs in SoCs before the chip manufacturing stage.

On the other hand, if design bugs are discovered after chip fabrication, a complete and expensive system redesign may be required to fix this bug. Nowadays, design verification takes 80% or more of the whole design process [6]. Therefore, performing system verification at each level of the design process is extremely important, especially in earlier design stages, since the cost of fixing bugs at later stages is very high.

To increase system verification efficiency, it is crucial to verify designs at a high level of abstraction. For state-of-the-art SoCs and embedded systems, designs at levels higher than register transfer level (RTL) are described in high level languages

like C/C++. High level design support is still in its infancy and the process is mainly manual or interactive. So developing verification techniques at this level becomes a more critical issue.

One very important issue in high level design methodology is to maintain the correctness of the design descriptions while they are refined into more detailed models and finally into RTL and implementation descriptions. This is basically an equivalence checking problem between two high level descriptions of the system design. Due to this nature of high level design processes of incremental refinement steps, equivalence checking is a very efficient verification methodology in this design paradigm.

Simulation techniques are widely used to verify systems at various levels of abstraction. Those techniques use simulation models to compute output values for given input patterns (test cases) and then compare them with expected correct values. The number of test patterns exponentially increases when the number of state variables in the system increases. So, it is infeasible to perform an overall design verification using simulation alone. In addition, because the quality of the simulation depends on the quality of the chosen test patterns, it is possible that some design bugs are not discovered during the simulation process. To compensate for those weaknesses, formal verification techniques [28] have been investigated and developed.

In formal verification, system specification and design are translated into mathematical models. Then, mathematical reasoning is used to verify the correctness of the design according to the specifications. Verification using formal techniques is exhaustive by nature because it explores all cases in the mathematical representation. This solves the test coverage problems of simulation techniques. The mathematical models used in formal verification include Boolean functions and expressions, first-order logic and others. Due to recent advances in mathematical modeling and reasoning techniques, formal verification techniques can now deal with larger and

more complex designs.

High level design descriptions in C/C++ are very easy to simulate. Hence verification by simulation techniques is the first thing to do. However, corner-case bugs coverage and performance problems reduce the efficiency of those techniques. Formal verification, on the other hand, can provide adequate coverage. However, it cannot efficiently handle the complexity and size of system descriptions at high levels of abstraction. Thus, semi-formal verification techniques have advanced a lot during the last decade.

Semi-formal verification tries to provide the coverage of formal methods and the scalability of simulation methods. Symbolic simulation [22] is a very important example of semi-formal verification techniques. Symbolic simulation is a technique first used to evaluate behavior under multiple input values or scenarios by encoding and evaluating them as symbols rather than numerical values. In the context of system verification, mathematical reasoning is used in conjunction with the results of symbolic simulation to answer the verification questions.

In this thesis we verify high level designs of the STMicroelectronics WiMax modem. To perform this system verification, we propose a complete semi-formal verification methodology to verify high level system designs. The methodology is based on modeling the system under verification using a mathematical description, then, utilizing an equivalence checking technique and an assertion based verification technique to address the verification problem.

The modeling technique uses system of recurrence equations (SRE) [1] to write the mathematical description of the system at various levels of abstraction. SRE is a flexible and easy tool to describe the functionality of systems at higher levels of abstraction [1]. Our proposed verification techniques use symbolic simulation of SRE descriptions of the system in conjunction with mathematical reasoning to perform the verification tasks. This methodology was successfully applied on the STMicroelectronics WiMax modem.

WiMax, (Worldwide Interoperability for Microwave Access) is a telecommunications technology that is grabbing increasing attention. 78 % of North America's telecom operators anticipate an investment in the WiMax deployment by the end of 2014 [5]. This popularity is driven by developments in worldwide spectrum allocation and standardization. As a result, the ability of WiMax to take advantage of emerging market opportunities is tied to the ability to test its products for regulatory and standards compliance. The main objective of this thesis is to verify a modem implementation provided by STMicroelectrics [19]. We focus on the verification problem for high level design descriptions of the WiMax modem.

In the following section we will discuss system verification techniques. We also focus on high level design in multi-level design methodologies.

1.2 System Verification Techniques

Verification consumes more than 80 % of the system design process. Most of this effort is dedicated to functional logic verification. The rest is used in performance verification, where the designer tries to check if the final manufactured device will meet the specified constraints of timing, area and power [11]. In this section, we will give a brief overview of the state-of-the-art in design verification techniques and their application in high level of abstraction.

1.2.1 Simulation Based Methods

Simulation is the most commonly used method for validation of models in industry. A typical simulation environment is shown in Figure 1.1. First, the design to be tested is described in some modeling language and is referred to as *Design Under Test* (DUT). The design specification is then used to generate the input and the expected output test vectors. The stimulus routine applies the input vectors to the DUT. The inputs are propagated through the model by a simulation tool and

finally the outputs are generated. Then, a monitor routine checks the output of the DUT against the expected outputs for each input test vector. If a mismatch is found, the designer can use debugging tools to trace back and find the source of the problem. The problem arises from either incorrect design or incorrect timing. Once the problem source is identified, the designer can fix it and simulate the new model.

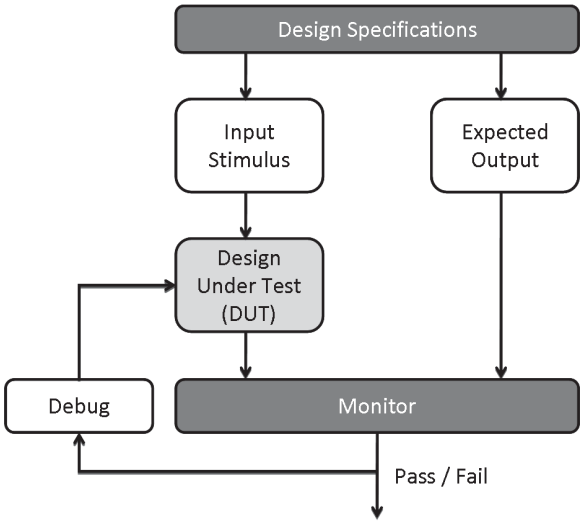

Figure 1.1: A Typical Simulation Environment.

Ideally, the designer should test the model for all possible scenarios. However, this would require an unreasonable number of test vectors and simulation time. So, the designer must use only a limited number of test vectors that are useful. The usefulness of a test case is usually defined by the number of components and connections it can excite in the simulation. Therefore, several coverage metrics have been suggested to quantify the usefulness of a test case.

One approach to reduce functional verification time is by modeling the system at higher abstraction levels. By abstracting away unnecessary implementation details, the model not only becomes more understandable, but also simulates faster. For this purpose, System Level Description Languages (SLDL) are used [31]. SLDL provide tools that help the designer to describe the concurrency, communications,

and synchronization required for the modeling. In addition, those languages provide sufficient abstraction support to hide low level implementation details in models of higher levels of abstraction [31].

In general SLDLs can be categorized into two main categories. The first group of languages builds upon C++, and adds libraries and templates to model concurrency and temporal characteristics. SystemC [29] is the most commonly used language that belongs to this category. The second group of languages builds upon hardware description languages such as VHDL and Verilog and extends them by adding mechanisms for modeling abstract data. OO-VHDL [30] and SystemVerilog [36] belong to this category of SLDLs.

1.2.2 Formal Verification Methods

Formal verification techniques use mathematical formulations to verify designs. In this subsection, we will give a brief overview about three basic techniques of formal verification, namely equivalence checking [33], model checking [20] and theorem proving [21]. Theorem proving takes formal representations of both the specification and the implementation using mathematical logic and proves their equivalence using the axioms and inference rules of the logic. Model checking, on the other hand, takes a formal representation of both the model and a given property, and checks if the property is satisfied by the model. Equivalence checking can be used to check for correctness of model refinements in high level designs or to verify models synthesis and optimization for processes. Some notion of equivalence such as logic equivalence or state machine equivalence should be defined and the equivalence checker proves or disproves the equivalence of the models at different stages of the design process.

Theorem Proving

Theorem proving is a method that provides mathematical proofs for given systems interactively. Due to its generality and mathematical basis, theorem proving can

be applied to almost any verification problem in any domain. The theorem proving problem from system verification point of view can be described as follows: Given two expressions in some logic (usually first or higher-order logic) [21], derive a proof for the equivalence or non equivalence of some expressions using the rules of that logic and the problem domain. Those expressions can be descriptions of specific functionality of a design at different levels of abstraction. Due to the underlying logic, this process is interactive where the user is the one who actually develops the proofs. So, theorem proving provides expressive and powerful reasoning. But, it is very difficult to automate the process and it is required to have expert-level mathematical knowledge to use this system effectively.

The focus of our thesis is to solve the verification problem at higher levels of abstraction. Theorem proving, in principle, can be used to verify the correctness of the design refinements at those levels. Each one of the models can be described in a specific formalism. Then, theorem proving tools can be used to prove the functional equivalence of those models. However, due to the large size and complex functionality of todays systems and the large number of design refinements, an interactive verification technique is very expensive. It is not efficient to use a pure theorem proving methodology to verify the whole system design process. However, theorem proving is a very practical solution to verify complex functional sections of the design where other verification techniques fail.

Model Checking

Model checking [20] is a formal technique for property verification. A typical model checking system is illustrated in Figure 1.2. First, the model is represented as a state transition system (also called a Kripke structure [34]), which consists of a finite set of states, transitions between states and labels on each state. The state labels are atomic properties that hold true in that state. The model checking problem is simply defined as a verification that a temporal property P is satisfied by the model defined

as a state transition system (M,s), where s is the start state for the model execution.

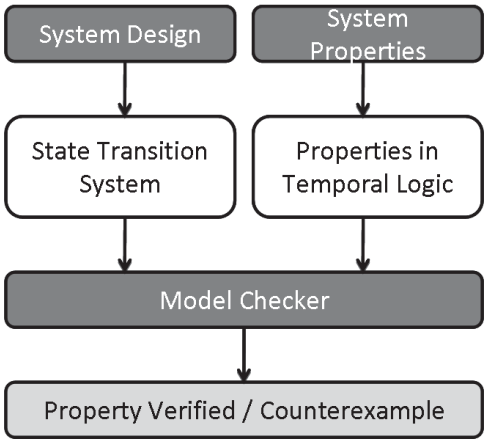

Figure 1.2: A Typical Model Checking System.

The major problem with model checking is the state space explosion. The state transition system grows exponentially with the number of state variables. Therefore, memory for storing the state transition system becomes insufficient as the design size grows. So, model checking can scale only to small sizes of high level design descriptions. However, some techniques are proposed to abstract large designs and focus on communication and synchronization properties of the system [6]. In this case, model checking is a practical solution to verify communication and synchronization components of the system, but not the datapath and computational components.

Equivalence Checking

In digital system design and during system design refinements of logic circuits, the design encounters a number of design transformations. The designer is responsible for the logical correctness of any such transformation. A logic equivalence checker verifies that the result of the design refinements is equivalent to the original design. This is achieved by dividing the model into logic areas separated by registers, latches or black-boxes. Those logic areas are called logic cones, as shown in Figure 1.3. The corresponding logic cones in the original and refined models are then compared.

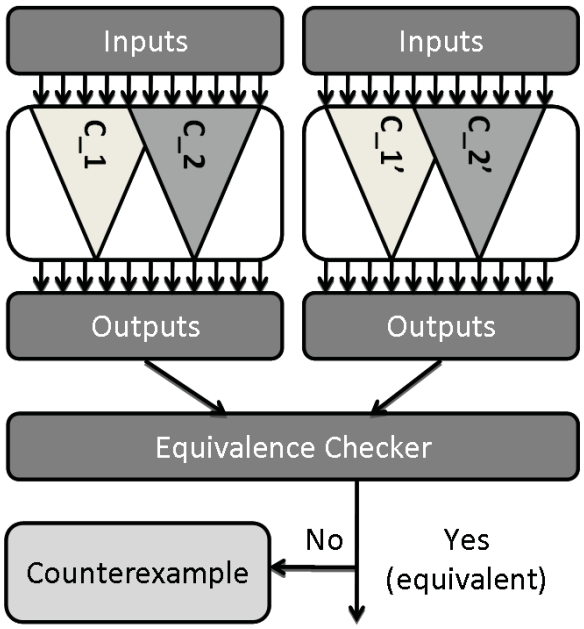

Figure 1.3: A Typical Equivalence Checking System.

Equivalence checking techniques are well developed and widely used in industry today due to their efficiency and scalability. For example, those techniques are used to prove the functional equivalence of system models at different levels of abstraction. Because of that, equivalence checking is a suitable formal technique to verify the design refinement decisions in a multi-level design approach.

1.2.3 Semi-Formal Verification Technique

It is a well-established fact that classical simulation techniques, while scalable, are unable to discover bugs in the hard to reach corner areas of nowadays complex design space. Formal verification, on the other hand, provides the coverage, but does not efficiently scale to large designs. Due to this complementary nature of available verification techniques, hybrid, semi-formal, techniques were developed to merge the advantages of simulation and formal method-based system verification techniques. Nowadays, the most widely used semi-formal verification technique is symbolic simulation [22].

Symbolic Simulation

In digital system design, symbolic simulation encodes and evaluates system signals as symbolic expressions. For example, considering the circuit in Figure 1.4, to simulate the circuit, we need to evaluate 23 concrete sets of inputs. However, using symbolic simulation by replacing inputs by the symbols a, b and c, the output symbolic expression can be evaluated in a single run of the simulation engine. The symbolic expression (a.b+b'.c) represents the value of the output for all 8 sets of inputs.

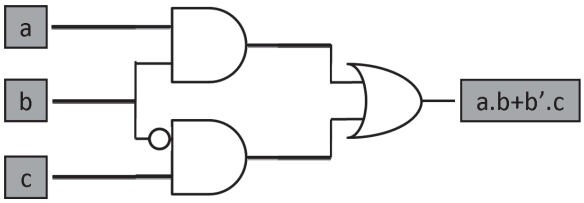

Figure 1.4: Symbolic Simulation Illustration Circuit.

From the verification point of view, replacing the expensive numerical simulations with symbolic simulation is another hybrid verification technique. The idea behind symbolic simulation is to significantly minimize the number of simulation test vectors, for the same coverage, by using symbols instead of explicit test vectors. In symbolic simulation, the stimulus applies symbols as inputs to the simulation model. During simulation, the internal variables and outputs are computed as symbolic expressions of the input variables. In order to check for correctness, the output expression is compared with the expected output expression for logic equivalence.

Due to its efficiency, symbolic simulation techniques can be used to compare high level design descriptions at different levels of abstraction. This is called *symbolic equivalence checking*. Figure 1.5 shows a typical symbolic equivalence checking system. This system, in principle, is similar to the classical equivalence checking system. Symbolic inputs and outputs are used to evaluate the corresponding functional area of each of the system descriptions (shown in the figure as C_1, C_2, C_1' and C_2'). Then advanced decision procedures are used to decide about the equivalence

of the generated equivalence traces, hence, the functional equivalence of the models.

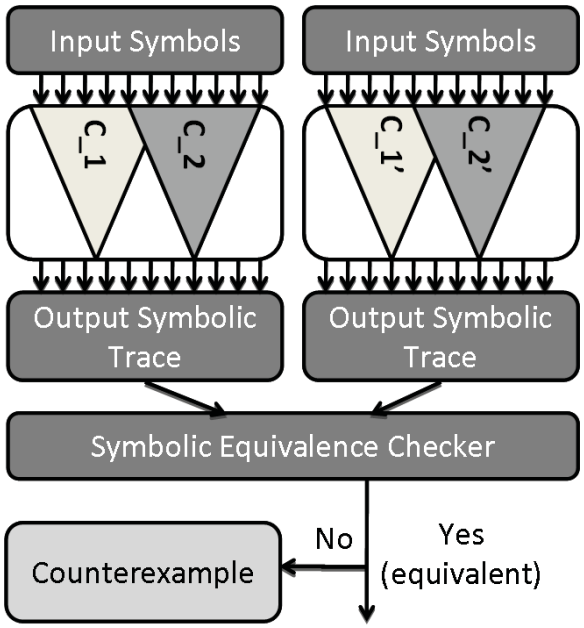

Figure 1.5: A Typical Symbolic Equivalence Checking System.

1.3 Related Work

Due to the increase in size and complexity of telecommunication hardware applications, major advances in design methodologies and automation occurred. The main trend to face this challenge is to use a top down multilevel design approach. In [12], Deb *et al* proposed a Transaction Level Modeling (TLM) based design methodology. In this work, the design process starts by writing a system level description of the system in C or Matlab. Then a series of design decisions are taken to refine the model into more realistic ones in terms of computation and communication structures. The authors of [12] defined systematically three transaction level models, which reside at different levels of abstraction between the functional and the implementation model of a digital signal processing (DSP) system. They are Process Transaction Model, System Transaction Model and Bus Transaction Model.

In [6], Fujita *et al* presented a similar multi-level system design methodology. They discussed the proposed methodology in the context of C/C++ based design and other system design languages like SpecC [35] and SystemC [29]. The major levels specified in this work are Functional Model, Architecture Model and Communication Model.

In industry, various leaders in telecommunications follow a multilevel design process that starts with highly abstracted functional models of the system. In [23], Altera presented a framework called "Altera DSP Builder" to transfer functional description in Matlab/Simulink of DSP systems into Alteraś FPGA Hardware implementation. Altera shows the application guidelines of this framework to physical layer implementation of the WiMax OFDMA modem.

In [17] [18] STMicroelectronics presented a complete framework for designing DSP systems using a set of design refinements that are applied on C/C++ descriptions of the target system. The WiMax OFDMA modem PHY layer implementation [19], our case study in this thesis, is a direct application of the multi-level system design framework.

There are many proposed system level models of the WiMax modem. In [13], the authors presented a high level model of the WiMax modem using Transaction Level modeling (TLM) in SystemC. Jie He *et al* provided in [14] a system-level time domain behavioral model for a mobile WiMax transceiver. Their model was implemented in Matlab/Simulink. The main purpose of this model is to be used for evaluating the functionality and the performance of design alternatives at higher levels of abstraction.

This literature survey shows that multi-level design and high-level (system-level) design approaches are widely accepted in VLSI and SoC designs. Thus, functional verification in those levels is very important since designers sometimes need to go back to high-level or system-level when functional bugs are found in the later design phases such as RTL or gate-level. This implies that efficient verification

methods for high-level designs can significantly improve the design productivity by detecting more bugs in high-level.

1.3.1 WiMax and OFDM PHY Verification

WiMax physical layer designers chose to use the Orthogonal Frequency Division Multiplexing (OFDM) technology. This technology is also used in designing the physical layer of Wi-Fi modems. There have been some active industrial and research contributions to address the OFDM physical layer verification problem.

In [26], Agilent Technologies present their WiMax design and verification kit. This kit includes the reference values and measurements to which the final product should conform in order to be a WiMax certified product. This verification kit depends on pure simulation techniques to verify the correctness of the system.

Chiang *et al* used SystemVerilog to validate the physical access layer of WiMax systems [25]. Again, their methodology proposes simulation as the verification technology. However they improved the simulation process by using SystemVerilog to generate random and valid test frames for the WiMax modem.

In [27], Nasser formally specified and verified an implementation of the Orthogonal Frequency Division Multiplexing (OFDM) Physical Layer using theorem proving techniques based on the HOL (Higher Order Logic) system. In his work he followed a framework, developed at Concordia University, incorporating formal methods in the design flow of digital signal processing (DSP) systems in a rigorous way. This methodology addressed the verification problem at the register transfer level (RTL).

The verification of the WiMax modem design at high levels of abstraction is still an open problem. Due to the complexity of the design of the WiMax physical layer and its dynamic operation modes, it is a challenging task to formally specify and verify the design efficiently at all levels of abstraction. We focus on the verification problem of the functional and architecture levels of the WiMax design.

1.3.2 Equivalence Checking

In [3], Ritter *et al* proposed a new symbolic simulation based methodology to verify the equivalence of design descriptions at register transfer level. In this methodology, information on internal equivalent point pairs (verification objectives) is collected to verify the output equivalence. Although this methodology is efficient, it requires more knowledge about the internal implementation of the system in order to identify the equivalence points. In addition, this method focuses on designs at lower levels of abstraction and relies on binary expressions which are not adequate to reason about word-level variables in high level designs.

In the field of formal software verification, Matsumoto *et al* presented in [2] an equivalence checking method for two C descriptions. Their method uses symbolic simulation to prove the equivalence of all variables in the descriptions. In addition, the method identifies textual differences between descriptions to reduce the number of equivalence checking tests among variables and increase the speed of the process. We use a similar concept to improve the performance of our methodology to compare high level descriptions of hardware designs.

In [11], Abdi *et al* presented a formal method to verify system model level transformations in a multilevel design methodology paradigm. In this work, a model algebra (MA) is defined and used to describe the systems to be checked for equivalence. Then (MA) expressions of the system are manipulated to realize the model transformations. Finally, the equivalence of the models is checked by proving the correctness of the encountered model transformations. The only limitation of this methodology is that it defines a set of rules (laws) that are used to transform one model into a functionally equivalent one. If the logical transformations between two models does not use those predefined rules, then the correctness of this transformation cannot be proven. This bounds the application to the set of supported design methodologies and model transformations. This work focuses on the correctness of the transitions rather than the functional correctness of the transformed models

themselves.

Fujita *et al* [6] proposed an algorithm to verify the equivalence of high level design description in C/C++ using symbolic simulation. Their algorithm identifies functional differences between descriptions by looking at the corresponding input, output and internal signals of the two models. It also uses identification of textual differences in symbolic traces to reduce the number of equivalence checking operations.

All this research relies on symbolic simulation to compare similarities between different descriptions of the system under test. We also follow a symbolic simulation paradigm. However, we use a higher level of symbolic simulation that is based on sequence of recurrence equations (SRE) and pattern matching. In [1], the notion of recurrence equation is extended to describe digital systems for formal verification purposes. Due to the flexibility and ease of describing the functionality of systems at higher levels of abstraction using SRE, it is more adequate for the multilevel design methodology. In addition, the powerful mathematical reasoning developed in the symbolic algebra and recurrence equations supports the verification process and decision procedures.

1.3.3 Property Checking

In [24], the authors proposed a symbolic simulation methodology to verify Property Specification Language (PSL) [37] properties in VHDL system descriptions. The advantage in their proposed methodology is that it uses word-level variables and it does not depend on binary decomposition of variables and BDD representation of assertions. This increases the scalability of the verification technique to larger and more complex designs. In fact, it is one of the strong points of symbolic simulation that it supports word-level simulation and reasoning.

In [6], the authors also proposed a methodology to verify synchronization and

concurrency properties in high level design descriptions. In their methodology, timing constraints (properties) are written in SpecC [35] as equalities and inequalities. Then reachability analysis of the model is performed to find error states. Finally, integer linear programming (ILP) solvers [6] are used to evaluate timing constraints. This methodology can also be applied to other liveness and safety properties of the system.

In [15] and [16], Zaki *et al* used symbolic simulation and SRE to verify properties in continuous systems and Analog and Mixed Signal (AMS) systems in particular. This methodology showed great results in terms of speed of system modeling for verification, speed of the verification and the coverage of the verification. This success in the AMS field inspired us in the field of system level verification of digital systems.

1.4 Proposed Verification Methodology

The main objective of this thesis is to verify STMicroelectronics WiMax modem physical layer design at higher levels of abstraction. The verification problem of WiMax is a challenging task due to the following reasons:

- The WiMax specification is complex. The IEEE 802.16 standard [38] that specifies the physical and MAC layers of the WiMax OFDM modem is 900 pages long!
- The WiMax modem implementation is left totally to the designer which may introduce unexpected bugs.
- The WiMax modem operation is highly dynamic and complex due to various operation modes supported by the modem.

The WiMax modem implementation follows a multi-level design paradigm. An efficient verification methodology that integrates with today's multi-level design

process of complex SoC is needed. And this verification has to meet the following characteristics and challenges:

- Efficient internal modeling technique that can scale to large system functionality.
- Efficient verification methodology that supports the large number of design refinements and backtracking at different levels of abstraction.
- Powerful mathematical reasoning that supports the word-level signal verification process at high levels of abstraction.
- Exhaustive methodology that provides enough coverage of increasingly complex designs.

To address those issues we propose the utilization of a multi-level semi-formal methodology. Figure 1.6 shows a basic block diagram of the proposed methodology. First, key system specifications (properties) of the system under verification are written using Sequences of Recurrence Equations (SRE)s. Then, model descriptions of the design under verification at each level of abstraction are translated into SREs as well. Finally, two complementary verification processes are used on those models. The first one uses symbolic simulation to verify the conformance of SRE models to key features of the system, written in the first step. The second verification process also uses symbolic simulation to prove the functional equivalence of SRE models, hence the correctness of incremental design refinement. This methodology guarantees the verification coverage of the design refinements at each level of abstraction and corner specification points in the design.

In this thesis, we present the application of our proposed methodology to the functional and architectural models of STMicroelectronics WiMax OFDM modem [19].

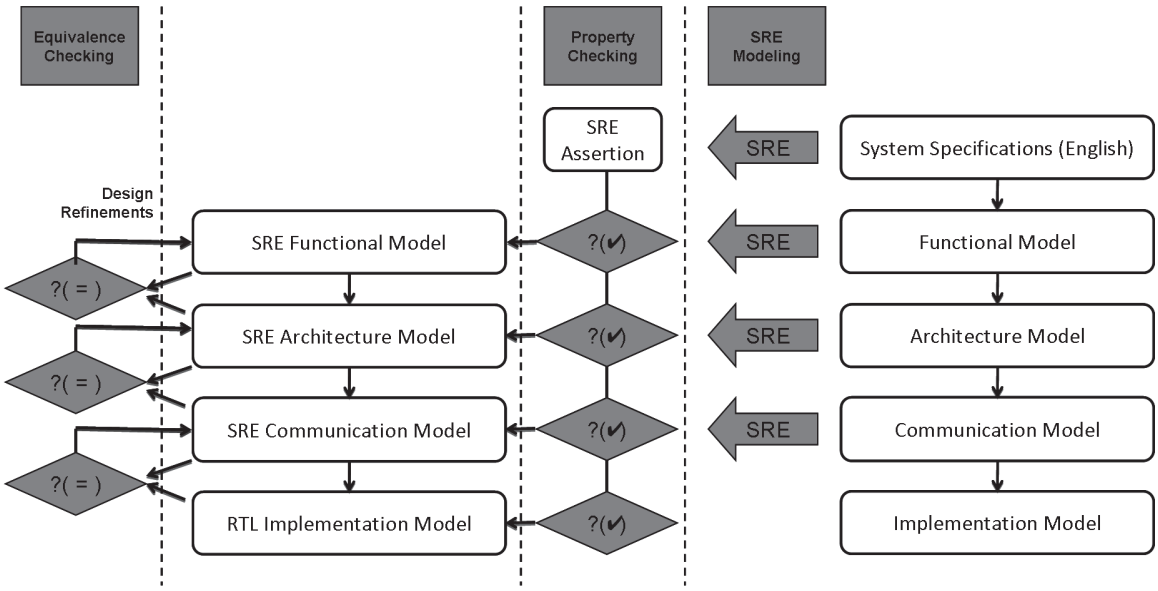

Figure 1.6: Proposed Verification Methodology Framework.

1.5 Thesis Contributions

In this thesis, we present the utilization of semi-formal functional verification methodologies to verify high level descriptions of WiMax system design in the multi-level design paradigm. We applied this methodology on the design of STMicroelectronics WiMax modem physical layer. Specifically, we focused on the verification of three design models of this system, one functional level description (executable specification) and two design refinements of architectural level descriptions. The contributions of this thesis can be summarized as follows:

- A symbolic simulation based property checking methodology to verify the conformance of system models at the functional and architecture levels of abstraction with key system properties. We successfully utilized this methodology to verify the conformance of a WiMax PHY layer implementation provided by STMicroelectronics with key WiMax system properties, based on the IEEE 802.16 (WiMax) standard [38]. We applied this methodology on the three functional and architectural level models.

- A symbolic simulation based equivalence verification methodology to verify the functional equivalence of two system models at the functional and architecture levels of abstraction. We successfully utilized this methodology to verify the functional equivalence of WiMax PHY system models at different levels of abstraction. We verified the equivalence of the functional model and the first architectural model. We also verified the equivalence of the two architectural models.

1.6 Thesis Outline

The rest of the thesis is organized as follows: Background information about WiMax IEEE 802.16 standard and a description of the basic components of WiMax modem physical layer is presented in Chapter 2. In this chapter we also discuss the verification goals based on the WiMax standard. In addition, an overview of STMicroelectronics WiMax modem design process and main functional blocks are presented. Chapter 3 explains in detail our proposed methodologies based on equivalence and property checking. The chapter also includes some preliminary mathematical background about symbolic simulation and sequence of recurrence equations. In Chapter 4, we describe the modeling and verification of STMicroelectronics WiMax modem using the proposed methodology. Experimental results of the process are also discussed in this chapter. Finally, Chapter 5 concludes the thesis and discusses future directions of our work.

Chapter 2

IEEE 802.16 Standard and ST WiMax Design Overview

This chapter will give a brief overview of the WiMax IEEE 802.16 standard and the capabilities that this technology enables. It will also provide some important background information about the main components in the physical layer of an IEEE 802.16 modem. This chapter will also provide an overview of STMicroelectronics (ST) WiMax modem design methodology, verification goals and main components.

2.1 WiMax IEEE 802.16 Standard Overview

The IEEE 802.16 standard was designed for fixed broadband wireless access to the local and metropolitan area networks (MAN). Although the 802.16 family of standards is officially called WirelessMAN in IEEE, it has been commercialized under the name "WiMAX" (for "Worldwide Interoperability for Microwave Access") by the WiMAX Forum industry alliance. The Forum promotes and certifies compatibility and interoperability of products based on the IEEE 802.16 standards.

IEEE802.16 systems are capable of transmitting and receiving shared data at rates up to 120 Mbps for Line of Sight (LOS) transmission and 70 Mbps Non-Line

of Sight NLOS. This puts the performance of the air interface comparable to that of cable, DSL or T1 systems. For example it can simultaneously support more than 60 businesses at T1 level and hundreds of homes with DSL rate connectivity at 20 MHz bandwidth [5]. In addition to those very high data rates, IEEE 802.16 systems is capable of providing:

- Long range operation: radius up to 30 miles
- Guaranteed service levels and Quality of Service (QoS) control.
- Superior scalability due to advanced multiplexing and link sharing techniques.
- Routable networks within existing wireless solutions such as WiFi.
- Cost savings and quick network setup compared to wired solutions.

Typical commercial sector applications for IEEE 802.16 include a cellular backbone. The robust bandwidth management schemes of the IEEE 802.16 makes it a good replacement of leased wire lines or microwave links as a cellular backbone. This technology also provides a very good solution for businesses that relocate frequently within a metropolitan area, such as construction companies. The broadband wireless service can be very quickly provided to these companies in a very short time as they move from one location to another without the need to re-wire. Current research in the WiMax applications area focuses on higher reliability networks and on Enhancements to Support Machine-to-Machine Applications [5].

This long list of advantages in the WiMax technology imposes challenges and complications on both network deployment and hardware system design. In this thesis we will focus on the hardware design and verification part of the story. Specifically, we will address the issue of verifying the design correctness of the physical layer (PHY) of the WiMax modem.

2.2 WiMax Modem Physical Layer (PHY)

The IEEE 802.16 standard essentially standardizes two aspects of the air interface - the physical layer (PHY) and the Media Access Control layer (MAC). This section provides an overview of the technology employed in the physical layer in the specification.

WiMax uses Scalable OFDMA (Orthogonal Frequency Division Multiple Access) technique to carry data, supporting channel bandwidths between 1.25 MHz and 20 MHz, with up to 2048 sub-carriers. This technique distributes data over a large number of carriers separated at precise frequencies. The orthogonality provided by the spacing in the OFDMA technique prevents the demodulators from seeing frequencies other than their own. The benefits of OFDM are the high spectral efficiency, resiliency to Radio Frequency (RF) interference, and lower multi-path distortion [7].

The IEEE 802.16 standard utilizes modulation using either:

- Quadrature phase shift keying (QPSK).
- 16-bit quadrature amplitude modulation (16 QAM).
- 64-bit quadrature amplitude modulation (64 QAM).

A unique feature of the IEEE 802.16 standard is its use of adaptive burst profiling. Adaptive burst profiling allows the radio to make adjustments to the modulation and coding schemes being used in response to changing environmental conditions and the resulting signal quality [8]. Systems using adaptive burst profiling constantly monitor signal quality. Adaptive adjustments on a frame by frame basis can then be made by shifting between more efficient and less robust QAM to less efficient but more robust QPSK when needed.

In downlink communication, that is when one base station (BS) talks to multiple subscriber stations (SS)s, time division multiplexing (TDM) is used [8]. Each

uplink channel is divided into several time slots and these slots are dynamically assigned based on the moment to moment needs of the systems by the MAC layer of the BS.

Both time division duplexing (TDD) and frequency division duplexing (FDD) are allowed in the IEEE 802.16 standard. In TDD, the uplink and downlink take turns transmitting on a shared channel, whereas FDD allocates separate channels for uplink and downlink. The standard also allows half duplex FDD where the uplink and the downlink use one channel similar to TDD.

2.2.1 Error Control

Error control techniques are very important in the design and verification process of the WiMax modem. Error control blocks constitute the largest portion of the physical layer. The IEEE 802.16 standard uses two main methods of error control in the PHY layer design. They are: Forward Error Correction (FEC) and Automatic Retransmission Request (ARQ).

Forward Error Correction

FEC techniques typically use error-correcting codes that can detect with high probability the error location. The most commonly used FEC scheme in IEEE 802.16 utilizes Reed-Solomon (RS) based on the Galois field (GF) 256 code. Turbo code (TC) is another more robust scheme that can be used to either increase the range of the BS or to increase the throughput [8]. In Reed-Solomon error correction codes, a polynomial is first constructed from the data symbols. Then an over-sampled plot of this polynomial is transmitted rather than the original symbols themselves. Because of the redundant information contained in the over-sampled data, it is possible to reconstruct the original polynomial and thus the data symbols. The degree of error tolerated by an error correcting code varies from code to code [9]. After applying the error-correcting codes, the stream of Reed Solomon encoded data blocks is passed

through a rate compatible Convolutional Code (CC) block to interleave the data. Here a code rate can be defined for convolutional codes as well. If there are k bits per second input to the convolutional encoder and the output is n bits per second, the code rate is k/n. more details about error correction blocks will be discussed in the next section when we discuss STMicroelectronics WiMax modem design.

Hybrid Automatic Retransmission Request (HARQ)

The HARQ technique allows retransmission of individual bits of data that may have been lost in the original transmission. This allows for a possibility to correct errors before the data is sent to a higher layer for processing [38].

2.2.2 Framing

Frame durations of 0.5, 1 or 2 milliseconds are specified in the physical layer of the IEEE 802.16. Each frame is further divided into physical slots each of which is 4-QAM symbols long. The details of the frame structure is out of the scope of this thesis. However, it is very important to note that the data frame in the physical layer in IEEE 802.16 is divided into control section and payload. The control section of the frame provides SSs with the characteristics of the downlink channel and provides BSs with the characteristics of the uplink channel [8].

2.2.3 Transmission Convergence (TC) Sublayer

The TC sublayer exists between the PHY and the MAC. The TC sublayer takes variable length MAC protocol data units (PDU) and organizes them within fixed length FEC blocks prior to transmission. A 1-byte pointer is then added at the beginning of the TC PDU to indicate the first byte of the next MAC PDU within the TC PDU. In the event of lost data transmissions, this pointer allows for resynchronization between the SS and the BS [8].

2.3 STMicroelectronics WiMax OFDMA Transmitter Overview

In this section we will provide an overview of ST WiMax modem. An overview of the WiMax Wireless-MAN transmitter architecture is shown in Figure 2.1 [10]. In summary the different data processing steps are:

1. Randomization: scrambling is used to avoid long sequence of 0s or 1s. This improves the coding performance in the next steps.

2. Convolution Coding: first part of the FEC processing. It adds redundancy to the signal to help identify error locations in the transmitted data. This coder is implemented as a Reed-Solomon Coder and a Convolutional Coder.

3. Puncturing: Second part of the FEC processing. It is used to reduce the number of bits to be transmitted based on coding rate.

4. Interleaving: changes bit ordering to minimize impact of burst transfer error impact (e.g., fading, signal level drop or other RF conditions).

5. Modulation: mapping bit on carrier amplitude, depending on the QPSK, 16QAM or 64QAM mode.

6. Burst mapping: maps modulated amplitude on specific subcarrier used for data, depending on the subframe zone defined by the OFDMA burst mapper.

7. Pilot insertion map pilot amplitude on subcarrier used for pilot.

8. Inverse FFT: generate signal in time domain from the subcarrier amplitude, for each OFDMA symbol.

9. Insert guard period: add a guard time to each OFDMA symbol.

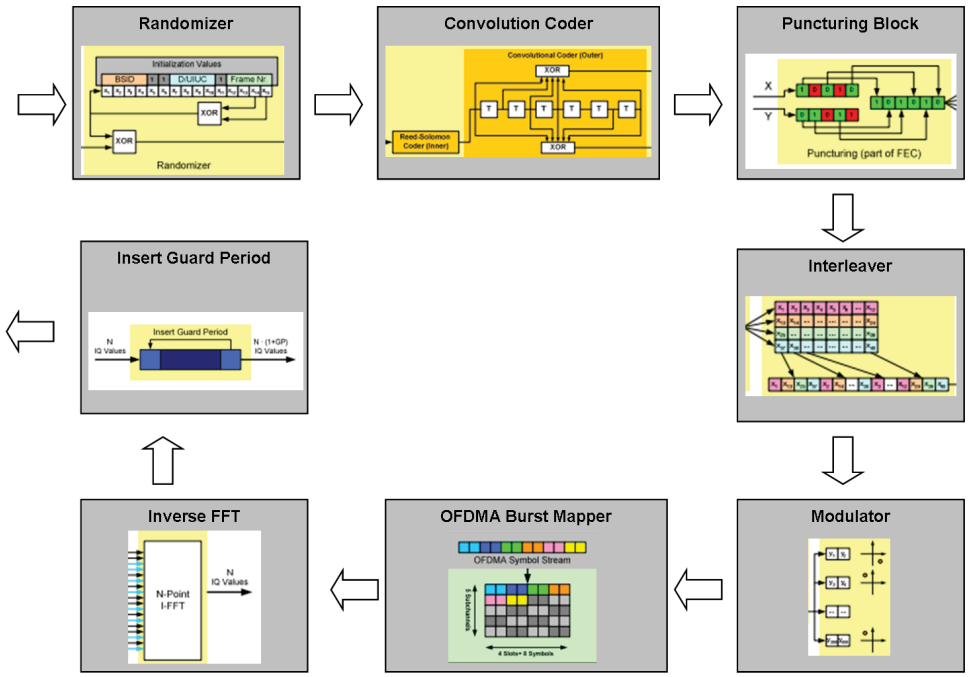

Figure 2.1: WiMax Wireless-MAN Transmitter Architecture.

The receiver architecture is mainly the inverse of the transmitter with at least the following additional processing:

- Data decoding (the inverse of convolution encoding) is more complex with respect to processing complexity.
- Require extra processing for frequency synchronization and equalization.

2.3.1 WiMax Transceiver Major Processing Blocks

STMicroelectronics have divided the Transmitter/Receiver into each one of the major processing blocks. This section will provide an overview of the scope of each of those processing blocks.

Transmitter

- **Data source:** Data source is a variable rate random data generator. Depending on preset parameters, the blocks will generate a certain amount of data at each sampling time. This block will not be part of the mapped blocks. The application rate and throughput is taken into account to set the sampling time.

- **Modulator bank blocks:** This block includes all algorithms related to channel coding. It is composed of seven subsystems, each one representing the modulation mode mandatory in the WiMax Wireless-MAN OFDM [38].Only one subsystem is active at any point of time. A control signal is used to enable a specific subsystem (modulation mode) dynamically during the operation of the modem. An RS-CC encoder followed by interleaving and digital modulation blocks compose each subsystem. RS block encodes an input stream vector using an (N, K) Reed-Solomon encoder. For full description of each block see [38].

- **OFDM frame assembly:** The frame assembly is an example of simple operation on data streams. Those operations are selective vertical and horizontal concatenation. Elementary operations on streams could be regarded as specialized instructions to the processing unit.

- **IFFT:** Once the bits are mapped and the frame is assembled, a 256 points IFFT block moves information from frequency domain to time domain.

Receiver

- **FTT:** 256 points FTT block extracts the amplitude value of each subcarrier.

- **Channel estimation and equalization:** Channel estimation is responsible of estimating the channel using the pilot and/or preamble. The estimated

channel is used to perform data equalization.

- **Frame disassembly:** This block manipulates the data stream to disassemble the frame by removing the preamble and pilots.

- **Demodulator bank:** It implements the inverse operation of the modulator bank. For each modulation mode a chain of processing blocks is activated. Each processing chain is formed by a demodulator followed by a deinterleaving, zero padding Viterbi decoder and Reed Solomon decoder.

ST's implementation of the physical layer model of the WiMax is an extension of a WiFi OFDM model. The burst mapping step is the main difference between the two models. The mapping is simpler in OFDM than OFDMA. OFDMA supports many more options: different zone types, further combinations of encoding and modulation (up to 52 combinations of encoding and modulation are supported in the IEEE 802.16 standard [38], yet only 7 are mandatory, as in the case of OFDM). Table 2.1 shows the mandatory modes of operation in the WiMax system.

Table 2.1: WiMax Supported FEC Code Types

Value	Modulation	Encoding Scheme	Puncturing Mode
0	QPSK	Convolution Coding (CC)	1/2
1	QPSK	(CC)	3/4
2	16-QAM	(CC)	1/2
3	16-QAM	(CC)	3/4
4	64-QAM	(CC)	1/2
5	64-QAM	(CC)	2/3
6	64-QAM	(CC)	3/4

2.3.2 Model Based Design Methodology

STMicroelectronics follow model-based design methodology. In this design methodology a set of design refinements are performed from stage to stage in the design

process. ST's model-bases design methodology for embedded systems is explained in details in [42].

The WiMax Transmitter design that was provided by ST has been provided as an example for the functional requirement mapping into hardware platform. This mapping proposal is not bound to any ST product. We identified three refined models of abstraction in the provided design. We will discuss those models and their implementation details in Chapter 4.

- **model 1:** In this model all functional blocks of the outer modem are executed sequentially in a single thread. No parallelism or communication structures are involved in this modeling level (see Figure 2.2).

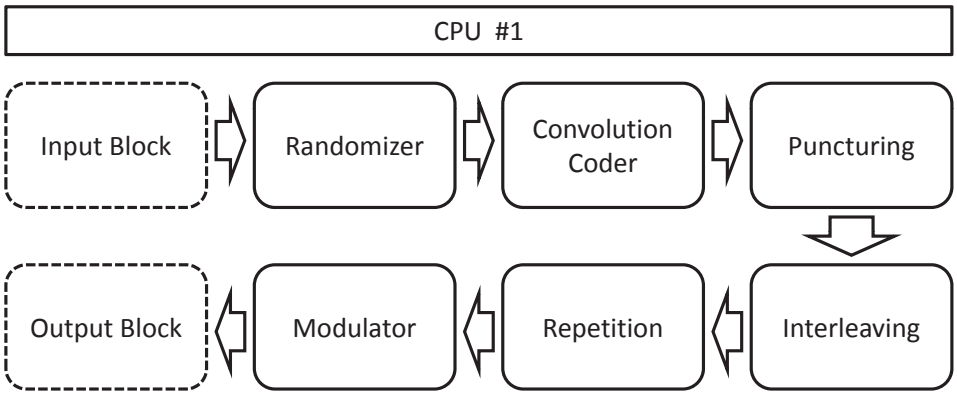

Figure 2.2: WiMax Transmitter Model - Sequential.

- **model 2:** In this experiment, the transmit-only part of the outer modem is mapped to 8 STxP70 processors - one processor per block - using multi-STxP FIFO Application Programming Interface (API) implementation [17] as shown in Figure 2.3.

- **model 3:**The transmit-only part of the outer modem is mapped to 4 STxP70 processors - one for the input block, one for the output block, and two for a partitioning of the six core functional blocks - using multi-STxP FIFO API

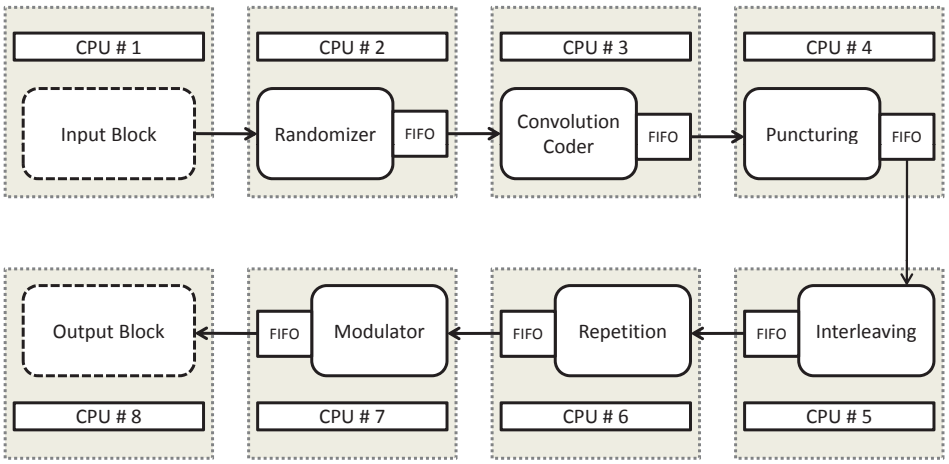

Figure 2.3: WiMax Transmitter Model - FIFO Based.

implementation, and a dynamic scheduler to distribute the tasks over the computing resources (see Figure 2.4).

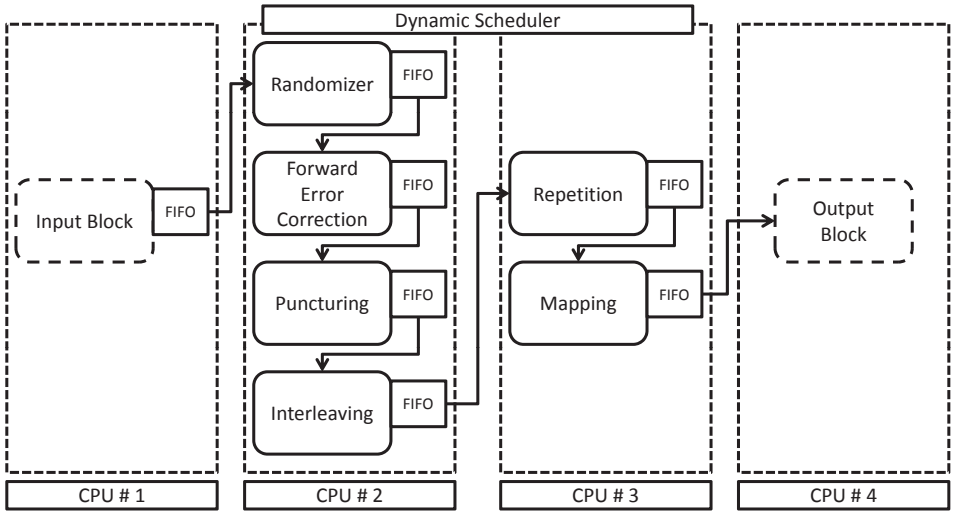

Figure 2.4: WiMax Transmitter Model - FIFO and Scheduler.

ST's design framework includes APIs for block definition, block configuration, and block communication as well as a run-time API for block execution. To understand some high-level application characteristics, we studied a simplified model of a WiMax Wireless-MAN OFDMA transceiver with a simple channel estimation

algorithm implemented in Matlab/Simulink. The model uses the default Simulink blocks for digital transmission (channel coding, interleaving, modulation and FFT processing).

The focus of our verification work will be on the modulation/demodulation and the channel estimation blocks. After identifying the three refined models of ST's WiMax modem. Our verification objectives can now be defined more clearly as:

- Prove the functional equivalence between the three models which shows the correctness of the refinement process.
- Prove the conformance of those models to the system specifications.

Chapter 3

Symbolic Simulation Based Verification Methodology

In this chapter, we present a detailed description of the proposed verification methodology. The methodology is based on symbolic simulation of mathematical models which are syntactically equivalent to the models under test. Symbolic simulation is faster than numerical simulation which saves the time needed for the verification process. Specially for systems with large computational data path components.

This chapter will start by brief mathematical preliminary background that is needed to fully understand the verification tasks. Then it will describe in detail the two main parts of the verification process, Equivalence Checking and Property Checking, by discussing all needed theories and algorithms.

3.1 Preliminaries

In this section some mathematical preliminaries about important concepts will be discussed. In particular, the focus will be on Symbolic Simulation and Sequence of Recurrence Equations (SRE).

3.1.1 Symbolic Simulation

Symbolic simulation is a form of simulation where many possible executions of a system are considered simultaneously. This is typically achieved by abstracting the domain over which the simulation takes place. A symbolic variable can be used in the simulation state representation in order to refer to multiple executions of the system. For each possible evaluation of these variables, there is a concrete system state that is being indirectly simulated. The symbolic simulation described in this section relies on rewriting rules based on the algorithms developed in [1] for digital systems. In the context of functional programming and symbolic expressions, we define the following functions.

Definition: Substitution.

Let u and t be two distinct terms, and x a variable. We call $x \to t$ a substitution rule. We use $Replace(u, x \to t)$, read replace in u any occurrence of x by t, to apply the rule $x \to t$ on the expression u.

The function *Replace* can be generalized to include a list of rules. *ReplaceList* takes as arguments an expression *expr* and a list of substitution rules $\Re = \{\Re_1, \Re_2, ..., \Re_n\}$. It applies each rule sequentially on the expression. The symbolic simulation function $ReplaceRepeated(Expr; \Re)$ shown in the definition below is based on rewriting by repetitive substitution, which applies recursively a set of rewriting rules $\Re$ on an expression *Expr* until a fixpoint is reached.

Definition: Repetitive Substitution.

Repetitive Substitution is defined using the following procedure:

$ReplaceRepeated(expr; \Re)$ applies a set of rules $\Re$ on an expression *expr* until a fixpoint is reached as shown in the next definition.

Definition: Substitution Fixpoint. A substitution fixpoint $FP(expr; \Re)$ is obtained, if:

$Replace(expr, \Re) \equiv Replace(Replace(expr, \Re), \Re)$

Depending on the type of expressions, we distinguish the following kinds of rewriting

Algorithm 3.1 Repetitive Substitution

1: $ReplaceRepeated(Expr; \Re)$;
2: $Begin$
3: **repeat**
4: $\quad expr_1 = ReplaceList(expr, \Re)$
5: $\quad expr = expr_1$
6: **until** $FP(expr_1, \Re)$
7: End

rules:

- *Polynomial Symbolic Expressions* R_{Math}: are rules intended for the simplification of polynomial expressions ($\mathbb{R}^n[x]$).

- *Logical Symbolic Expressions* R_{Logic}: are rules intended for the simplification of Boolean expressions and to eliminate obvious ones like $(and(a, a) \rightarrow a)$ and $(not(not(a)) \rightarrow a)$.

- *If-formula Expressions* R_{IF}: are rules intended for the simplification of computations over If-formulae. The definition and properties of the IF function, like reduction and distribution, are defined as follows:

  IF Reduction: $IF(x; y; y) \rightarrow y$
  IF Distribution: $f(A1, ..., IF(x, y, z), ..., An) \rightarrow$
  $IF(x, f(A1, ..., y, ..., An), f(A1, ..., z, ..., An))$

3.1.2 Sequence of Recurrence Equations (SRE)

A recurrence equation or a difference equation is the discrete version of an analogue differential equation. In conventional system analysis, recurrence equations are used in the definition of relations between consecutive elements of a sequence. In [1], the notion of recurrence equation is extended to describe digital circuits using the normal form: *generalized If-formula*.

Definition: Generalized If-formula

In the context of symbolic expressions, the generalized If-formula is a class of expressions that extend recurrence equations to describe digital systems. Let K be a numerical domain ($\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R} and \mathbb{B}$), a generalized If-formula is one of the following:

- A variable $X_i(n)$ or a constant $C \in K$
- Any arithmetical operation $\alpha \in \{+, -, \times, \div\}$ between variables $X_i(n) \in K$
- A logical formula: any expression constructed using a set of variables $X_i(n) \in B$ and logical operators: *not*, *and*, *or*, *xor*, *nor* . . . etc.
- A comparison formula: any expression constructed using a set of $X_i(n) \in K$ and comparison operator $\alpha \in \{=, <>, <, =, >, =\}$.
- An expression *IF(X, Y, Z)*, where X is a logical formula or a comparison formula and *Y*, *Z* are any generalized If-formula. Here, *IF(x, y, z)*: $B \times K \times K \rightarrow K$ satisfies the axioms:
  1. IF(True,X,Y) = X
  2. IF(False,X,Y) =Y

Definition: A System of Recurrence Equations (SRE)

Consider a set of variables $X_i(n) \in K, i \in V = 1...k, n \in Z$, an SRE is a system of the form:

$$X_i(n) = f_i(X_j(n - \gamma)), (j, \gamma) \in \epsilon_i, \forall n \in Z$$

where $f_i(X_j(n - \gamma))$ is a generalized If-formula. The set ϵ_i is a finite non empty subset of $1...k \times N$. The integer γ is called the delay.

3.2 Equivalence Checking

The problem of the equivalence checking between a high-level and a lower description level is a major challenge for system level design methodologies. We focus here on the computational equivalence of models; two models are not equivalent from the perspective of architecture but they still compute the same function. In particular we establish a proof for the computational equivalence between different abstraction models of the WiMax design.

The objective is to compare two descriptions of the system at deferent levels of abstraction. The higher level is referred to as *Specification Model (Spec)* while the lower description level is called *Implementation Model (Imp)*. As shown in Figure 3.1. The first step in the equivalence checking process is to translate each of these models to a mathematical model in terms of Systems of Recurrence Equations *SRE (Spec)* for *Spec* and *SRE (Imp)* for *Imp*. Then, we execute each model for a certain number of times using a rewriting based symbolic simulator.

The symbolic simulator is implemented inside the computer algebra system, Mathematica 6.0 [4]. It is used as a symbolic computation engine and as database of simplification rules. During symbolic simulation, a reduction is done on the SRE model where recurrence equations are considered as rewriting rules. After symbolic simulation,the obtained results are the symbolic traces of both models. The verification is achieved by comparing the *Spec* trace with *Imp* trace using: Pattern Matching and Equation solving in Symbolic Algebra. We chose Mathematica 6.0 because it has a very powerful computation engine. Also, it has many built in functions that perform pattern matching and symbolic equation solving.

3.2.1 Symbolic Simulation Algorithm

The symbolic simulation algorithm used in the symbolic trace computation step is based on rewriting by substitution. The idea is to compute the symbolic execution

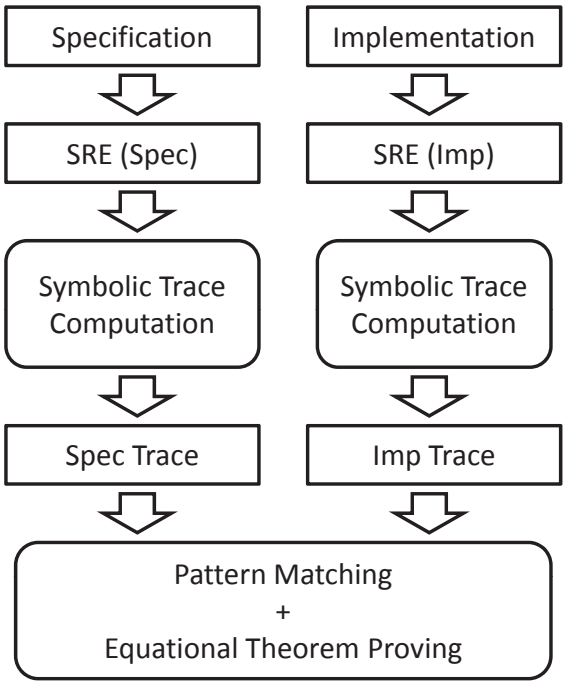

Figure 3.1: Equivalence Checking Methodology.

trace of the SRE model after n simulation cycles. During each cycle, the symbolic expressions of each design object are computed using a set of simplification rules. This algorithm is based on repeated substitutions as defined in Algorithm 3.1. The algorithm repeatedly applies a set of substitution rules R, until a fixed point is reached. Three kinds of symbolic expressions are considered: Algebraic, Logical and If-formula expressions. Each kind is associated with a set of rewriting rules: R_{Math}, R_{Logic} and R_{IF}.

- **Algebraic expressions R_{Math}:** are Mathematica built-in rules intended for the simplification of polynomial expressions ($R^n[x]$).

- **Logical symbolic expressions R_{Logic}:** are rules intended for the simplification of Boolean expressions and to eliminate obvious ones like (and(a,a)$\rightarrow$ a) and (not(not(a)) $\rightarrow$ a).

- **If-formula expressions R_{IF} :** are rules intended for the simplification of

computations over If-formulas. The definition and properties of the IF function, like reduction and distribution, are used.

We add to these rules the trace of the equation at time *n-1* that we consider as rewriting rules of the time (*n-1*) see [1] for more details.

3.2.2 Verification of the Symbolic Trace

The result of the symbolic simulation is a set of expressions that represent the symbolic trace of the system after n cycles. The comparison of expressions is achieved using: *Pattern Matching* [39] and *Equational Theorem Proving* [40]. Pattern matching is used to check that expressions have the desired structure, to find relevant structure, and to substitute the matching part with other expressions.In Mathematica, it is presented as of a regular expression language (Mathematica pattern language) and a set of matching functions. The designer writes properties of the form: $P = verify\ (U_i, S_i(t_n))$ where U_i is a regular expression that describes the expected symbolic expression of a simulated object. $S_i(t_n)$ is the symbolic simulation result of the element S_i after t_n simulation cycles.

Equational Theorem proving is an automatic technique that tries a wide range of transformations on an expression and returns the simplest form it finds. One of the more successful approaches to equational reasoning is the use of equations as one-way rewrite rules, so that a formula can be simplified by repeatedly replacing an instance of the left-hand side of a rule by its right-hand side until a simplest possible form is obtained.

3.2.3 Verification of Computational Equivalence

Algorithm 3.2 presents the used Computational Equivalence checking algorithm.

Algorithm 3.2 Computational Equivalence Checking

1: t = t_0;
2: $\phi(t_0) = \{Spec_j(t_0)\}$ $0 < j \leq m$;
3: **while** $t \leq K_{Spec}$ **do**
4: $\phi(t) = SymSim\_Step(\phi)$
5: If NoDeltaCycle then t = t+1
6: **end while**
7: SPEC = $\phi(t_0 + K_{Spec})$
8: t = t_0;
9: $\varphi(t_0) = \{Imp_i(t_0)\}$ $0 < i \leq m$;
10: **while** $t \leq K_{Imp}$ **do**
11: $\varphi(t) = SymSim\_Step(\varphi)$
12: If NoDeltaCycle then t = t+1
13: **end while**
14: IMPL = ReplaceRepeated ($\varphi(t_0 + K_{Imp}), R_{Abst}$)
15: MatchQ $(\varphi(T), \phi(T))$; // T = t_0 + k

Computing the Trace of the SPEC

(Lines 1-7): Line 1 first initializes the simulation time t to t_0 (equal to zero in most cases). The purpose of line 2 is to store the initial SRE of the SPEC model in the variable $\phi(t_0)$. Lines 3-6 repeatedly execute a symbolic simulation for K_{Spec} steps using the symbolic simulation algorithm; the time is advanced only if no more delta cycles are needed. K_{Spec} is determined by the verifier and it depends on the temporal complexity of the SRE description of the system. For the WiMax application we set K_{Spec} to 1 because the SRE describing the system is of first order. The variable SPEC stores the computed expressions in line 7. This is equivalent to a new SRE where the time variable is changed to $T = t_0 + K_{Spec}$. This traced SRE will be used to compare the traces in line 15.

Computing the Trace of the IMPL

(Lines 8-14): In the same way, the trace of the IMPL model is computed using a symbolic simulation for K_{Imp} steps (same as K_{Spec}). The new SRE where the time variable is changed to $T = t_0 + K_{Imp}$ is stored to be used to compare the trace of

the IMPL model in line 15. In fact, as the IMPL model is more detailed, the direct comparison is not correct. Thus, we need to add some abstraction rules to refine the computed expressions before comparing the results with SPEC. These rewriting rules R_{Abst} are intended to eliminate calls for functions that convert to integers and rename signals in the IMPL model by their correspondent in SPEC. In line 14, these abstracted expressions are stored in the variable IMPL.

Comparing Both Traces

(Line 15): Using pattern matching and algebraic verification, we verify that symbolic expressions in SPEC can be substituted by variables computed in IMPL. The traced symbolic expressions are put in a normal form, and then verified using the function *MatchQ*. This is a built in function in the computer algebra system, Mathematica 6.0 [4] and it implements the *Pattern Matching* and the *Equational Theorem Proving*. If the verification returns true, then computational equivalence is checked. Otherwise, the pattern matcher gives the non equivalent patterns.

3.3 Property Checking

Our methodology aims to prove that a system description satisfies a set of properties using pattern matching and equation solving in Symbolic Algebra. This is archived via several steps as shown in Figure 3.2. The system is described using recurrence equations. The properties are algebraic relations between signals of the system. The system description and properties are input to a symbolic simulator that performs a set of transformations by rewriting rules in order to obtain an SRE. These are combined recurrence relations that describe each property blended directly by the behavior of the system. The next step is to use Pattern Matching and Equation solving in Symbolic Algebra which is defined over the normal structure of the SRE. If the proof is obtained, then the property is verified. Otherwise, we

provide counterexamples for the non-proved properties.

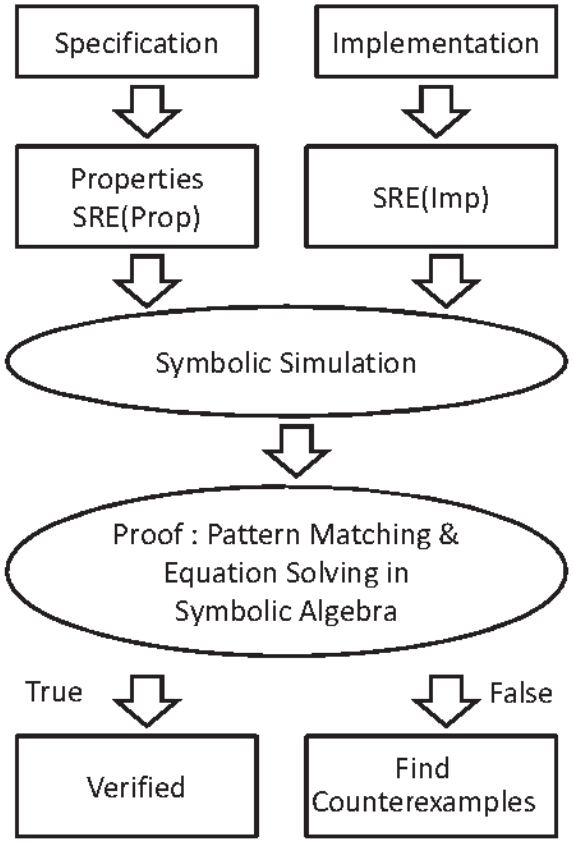

Figure 3.2: Property Checking Methodology.

The proposed property checking approach is given in Algorithm 3.3 which is described below.

Storing System Properties

(Line 1): *Prop (IMPL)* is the set of properties of the system that we want to verify. Those properties are written manually as a system of recurrence equations (SRE).

Computing the Trace of IMPL

(Lines 2-8): Similar to what we have done in the equivalence checking part, the trace of the IMPL model is computed using a symbolic simulation for K_{Imp} steps. In line 7

Algorithm 3.3 Property Checking

1: PROP = { Prop(IMPL) };
2: t = t_0;
3: $\varphi(t_0) = \{Imp_i(t_0)\}\ 0 < i \leq m$;
4: **while** $t \leq K_{Imp}$ **do**
5: $\varphi(t) = SymSim\_Step(\varphi)$
6: If NoDeltaCycle then t = t+1
7: **end while**
8: IMPL = ReplaceRepeated ($\varphi(t_0 + K_{Imp}), R_{Abst}$)
9: MatchQ (IMPL, PROP) // T = t_0 + k

the new SRE where the time variable is changed to $T = t_0 + K_{Imp}$ is stored in IMPL to be used for property checking later. In fact, as the IMPL model is more detailed, the direct property checking is not correct. Thus, we need to add some abstraction rules to refine the computed expressions before comparing the results with PROP. These rewriting rules R_{Abst} are intended to eliminate calls for functions that convert to integers and to rename signals in the IMPL model by their correspondent ones in PROP. In line 8, these abstracted expressions are stored in the variable IMPL.

Comparing PROP and IMPL

(Line 9): Using pattern matching and algebraic verification, we verify that symbolic expressions in PROP can be substituted by variables computed in IMPL. The traced symbolic expressions are put in a normal form, and then verified using the function *MatchQ*. If the verification returns True, then properties are checked. Otherwise, the pattern matcher gives a counter example.

3.4 Summary

In this chapter we have discussed the detailed mathematical description of the two approaches used in the verification process in this project, Symbolic Simulation based Equivalence Checking and Symbolic Simulation based Property Checking.

We used those methods to form a verification framework for a specific case study, the WiMax modem of STMicroelectronics.

We have described the main algorithms and mathematical characteristics of the proposed methodology, which can be summarized in the following points:

- SRE are used to model systems at different levels of abstraction and they are also used to model System Properties.
- Symbolic Simulation is used to compute Symbolic traces of the SRE models of the system under test.
- Efficient symbolic equation solving and pattern matching algorithms are used to prove the functional equivalence of the symbolic traces of each model, hence the functional equivalence of the corresponding system descriptions. They are used also to prove the conformance of those symbolic traces to the SRE modeled properties.

Chapter 4

ST WiMax Modem Verification

In this chapter we will describe in details the application of the chosen methodology on a WiMax modem design provided by STMicroelectronics.

The design complexity and wide range of functions and operational modes of the WiMax modem made it a good case study to apply our proposed verification methodologies. The objective of the project is to formally verify the WiMax modem designed by STMicroelectronics. This includes the design's conformance to the specifications and the verification of the equivalence between the system models implemented at different levels of abstraction.

ST provided us with three different C models of their proposed design, as described in Chapter 2. Each one of those models is at a different level of abstraction. They are:

- Model 1: Functional Level Model.
- Model 2: FIFO Based Process Transfer Model.
- Model 3: FIFO and Scheduler Based Process Transfer Model.

To achieve the goals of this project we applied the proposed verification methodology, described in Chapter 3.

We verified the functional equivalence of the first and second models and the equivalence of the second and third model. We first wrote the SRE description of each model using Mathematica. Then, we validated the correctness of their basic functionality using sample numerical simulation. Next, we generated symbolic traces for SRE models using symbolic simulation in Mathematica. Finally, we used Pattern Matching on those symbolic traces to verify the functional equivalence of all SRE models of the system (Algorithm 3.2). This equivalence implies the equivalence of the corresponding C models.

We also verified the conformance of those models to sample important properties of the WiMax transmitter. We wrote those properties as an SRE. Then we used Pattern Matching and Equational Solving Functions from Mathematica to verify those properties (Algorithm 3.3). The conformance of SRE models to those properties implies also the conformance of the corresponding C models to the same properties.

The modeling and verification experiments are divided into four main parts: model realization, symbolic simulation, equivalence checking and property checking.

4.1 Model Realization

The first step in our verification methodology is to translate the models under test from C models to SRE models. We used Mathematica 6.0 to write those SRE models. We developed an SRE model for the WiMax modem corresponding to each model of the three provided by STMicroelectronics as described in Chapter 2.

SRE Functional Level Model

Figure 4.1 shows the main building blocks of this model. First, we extracted the equations representing each of the system functional blocks according to ST's model. Then, we used those equations to model all Forward Error Correction (FEC) blocks

using SRE. Finally, we connected those SRE blocks serially in the correct order without using any extra communication components.

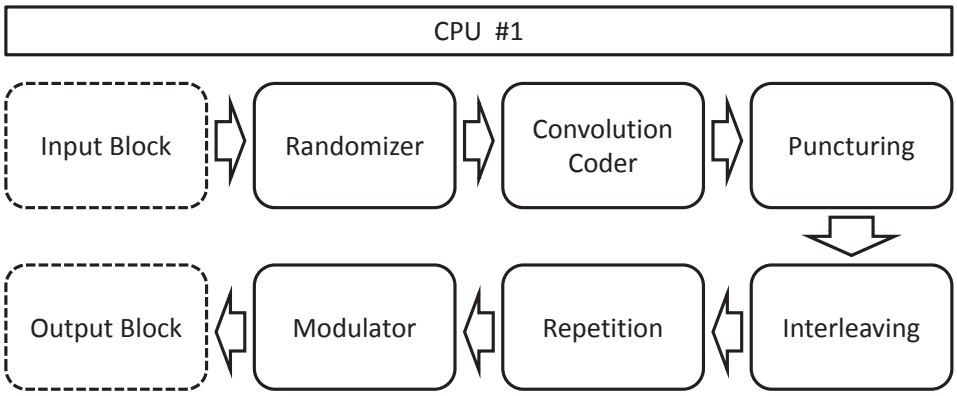

Figure 4.1: Functional Level Model.

FIFO Based Process Transfer Model

In this model, a FIFO structure is used for the communication between the functional blocks of the system (see Figure 4.2). The main purpose of introducing FIFOs in the system is to handle different timing requirements of the system blocks. The current implementation of the system considers that all blocks have zero delay. However, adding communication components to the system's functional blocks is a design decision which is important for more detailed system implementations. The following subsection describes the main features of the implemented FIFO structure.

FIFO Structure

We Modeled the FIFO structure based on the generic model provided by STMicroelectronics. This model implements the FIFO structure using the UNIX semaphores [41]. In our design we replaced the UNIX semaphores with guard local variables to synchronize access to different parts of the code. This determines the behavior and the interface of the FIFO.

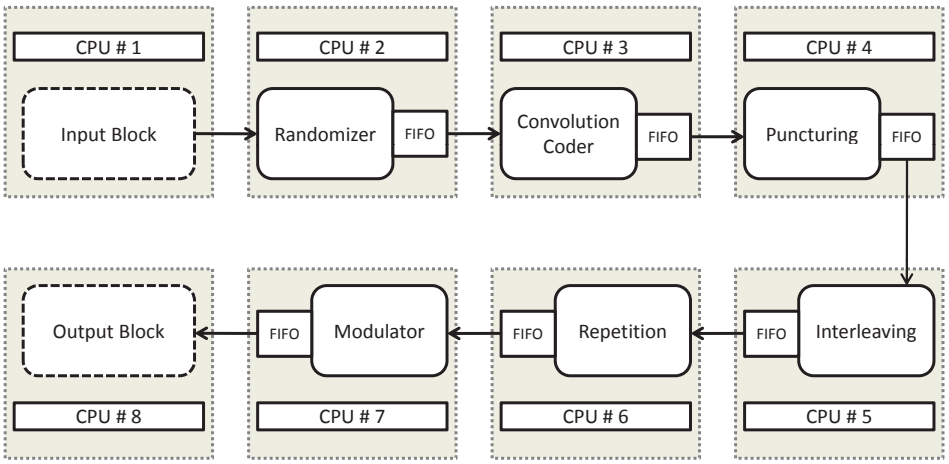

Figure 4.2: FIFO Based Process Transfer Model.

The FIFO used in the provided model supports only a single-writer-single-reader (SWSR) mode, this means that at any point of time, only one process is allowed to read from or write to the FIFO module. This FIFO supports four operations as its basic API. Two operations are used to insert and retrieve data from the FIFO. The other two operations are used by the processes to inquire about the status of the FIFO (full or empty). Four parameters are used to describe the characteristics of each instance of this FIFO structure (see Table 4.1 for the name and description of each parameter and basic API of the FIFO structure).

Table 4.1: FIFO Design Details

FIFO Operation	
Mode	SWSR (single writer single reader mode)
FIFO Parameters	
NBFIFOs	Number of FIFOs in the system
FIFOCellSize	Width of the FIFO cell (number of bits)
FIFONBCells	Depth of the FIFO (number of cells in the FIFO)
FIFOsTail	Pointer to the last populated cell of the FIFO
FIFO basic API	
Push (data)	Increments the FIFOsTail value and insert the data in the new position.
Pop ()	Decrements the FIFOsTail value.
IsFull()	Return True if the FIFOsTail value equals to the FIFONBCells.
IsEmpty()	Return True if the FIFOsTail value equals to the 0.

We described the FIFO structure as a sequence of recurrence equations. the SRE description of the FIFO module has one to one mapping with the model provided by STMicroelectronics. After implementing the FIFO in SRE, we make sure that its basic functionality is correct. This is achieved by a simple numerical execution of the FIFO SRE code. Finally, this module was integrated into the complete system SRE model.

FIFO and Scheduler based Process Transfer Model

The main feature of this model is mapping the functionality of more than one functional block to a single processing element (resource) in the system. Figure 4.3 shows the structure of this model. This mapping is based on time sharing, where

each one of the functional blocks sharing the same processing element will have a time slot to utilize the processor. To synchronize this time sharing of resources, a dynamic scheduling module is attached to each processing element. The main task of the dynamic scheduler is to assign the next time slot to a specific functional block. The following subsection describes the dynamic scheduler unit.

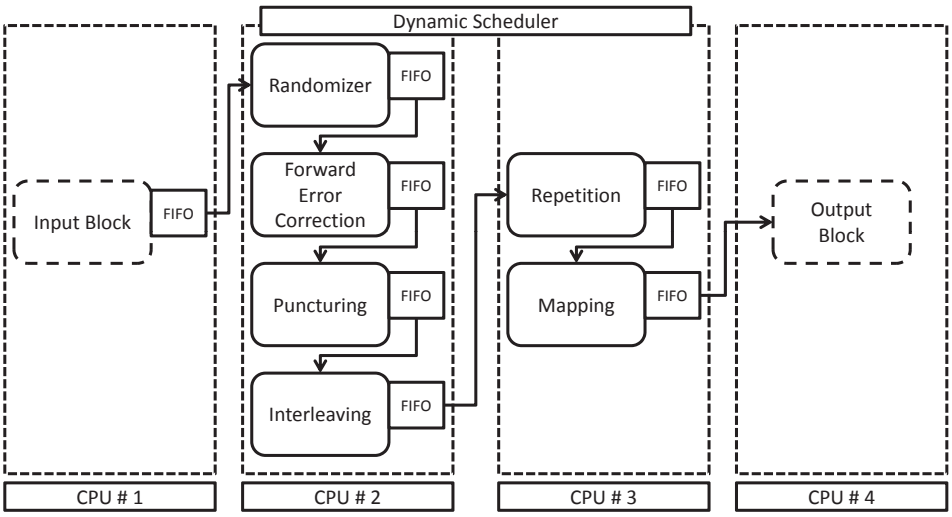

Figure 4.3: FIFO and Scheduler Based Process Transfer Model.

Dynamic Scheduler

In our implementation of the scheduler we followed the generic scheduler model provided by ST in the WiMax package. In this model, two of the CPUs need to be associated with a dynamic scheduler; CPU#2 and CPU#3. The basic functionality of the scheduler is to scan the functional blocks associated with a specific CPU in a round robin manner with a predefined order of the blocks. Then it chooses the first block which is ready for execution and marks it to have the control of the CPU in the next time step.

It is important to mention that each functional block of the system can take one of the following three atomic execution phases during the operation of the system:

1. Read: pops the input data blocks from the input FIFO.

2. Run: runs the block functionality on the input and computes the output.

3. Write: pushes the output data blocks to the output FIFO.

The block is considered to be ready for execution if it satisfies one of the following conditions:

1. If the block is in the read execution phase and its input FIFO is not empty.

2. If the block is in the run execution phase.

3. If the block is in the write execution phase and its output FIFO is not full.

We described the dynamic scheduler as a sequence of recurrence equations. Then the basic functionality of the module was validated using a simple numerical execution of the SRE model. Finally, this module was integrated into the complete system SRE model.

4.2 Symbolic Simulation

This is the second step in our verification methodology. Its main objective is to symbolically simulate the SRE models developed in the first step and generate their symbolic traces. Again we used Mathematica 6.0 as a rewriting engine to run our symbolic simulation.

The WiMax modem system is a dynamic system that supports varies modes of operation, Chapter 2 described briefly those modes of operation and their parameters values. Based on the system's operation flexibility we divided the symbolic simulation part into two phases. The first phase is called (Single Control Scenario). In this phase we considered a default mode of operation of the system. In the second phase (Multiple Control Scenarios) we coded the system so that it can work in different modes.

4.2.1 Single Control Scenario

The main objective of this phase is to perform symbolic simulation of the three SRE models of the WiMax modem. In this phase we have fixed the control signals of the WiMax system to support *mode 0* of operation according to the IEEE 802.16 standard (see Table 2.1).

We validated the correctness of the basic functionality of the SRE models using numerical simulation. We simulated SRE models in Mathematica using 100 randomly generated numerical test vectors. Then we ran ST C++ models using the same generated inputs. We compared the results and found that both ST and SRE corresponding models are sending exactly the same sequence of signals. This step is only performed to provide basic validation of the correctness of the manual translation between C++ and SRE models.

Finally we completed this phase by computing symbolic traces of the three SRE models using Mathematica.

Table 4.2 shows the time and memory utilization of the symbolic simulation experiments. We can notice the following:

- While numerical simulations take a relatively long time to generate numerical traces of 100 test vectors, symbolic simulation can compute symbolic traces in a much shorter time. Knowing that a symbolic trace of a system covers all possible input scenarios, we can say that symbolic simulation is definitely more efficient.

- Memory requirements for both numerical and symbolic simulation are close.

Table 4.3 Summarizes SRE models coding requirements with respect to the number of code lines and the number of SRE equations in the system.

Table 4.2: Numerical and Symbolic Simulation (Single Control Scenario)

Model	Run Time (sec)	Memory
C-Executable Functional Level - Numerical Simulation, 100 iterations	3.21 sec	2.3 MB
SRE Functional Level (FL) - Numerical Simulation, 100 iterations	10 sec	13.45
SRE Functional Level (FL) - Symbolic Simulation	0.321 sec	13.6 MB
SRE Process Transfer Level (PTL-8) (8 processors) - Numerical Simulation, 100 iterations	252.4 sec	16.54 MB
SRE Process Transfer Level (PTL-8) (8 processors) - Symbolic Simulation	5.17 sec	19.59 MB
SRE Process Transfer Level (PTL-4) (4 processors + scheduler) - Numerical Simulation, 100 iterations	266.4 sec	25.32 MB
SRE Process Transfer Level (PTL-4) (4 processors + scheduler) - Symbolic Simulation	5.26 sec	28.31 MB

Table 4.3: Model Coding Requirements (Single Control Scenario)

Model	SRE Eqns	Code Lines
SRE Functional Level (FL) (1 Processor)	8	280
SRE Process Transfer Level (PTL-8) (8 processors + FIFO)	46	772
SRE Process Transfer Level (PTL-4) (4 processors + FIFO + scheduler)	50	882

4.2.2 Multiple Control Scenarios

In this phase we focused on adding variable control signals to the system. Those control variables determine the systems mode of operation. The WiMax system supports 52 different modes of operation. Each one of them is coded with a value between 0 and 51, which is called FEC Code Type. The IEEE 802.16 standard specifies that modes 0-6 are mandatory. We modeled the control signals required to

support those modes. Also, we rewrote the recurrence equation description of some of the blocks to support these dynamic characteristics.

One of the design requirements by STMicroelectronics is to forward the control information inband with the data. That means the control information will be set at the traffic generator and then passed from one block to the other together with the transmitted data. Each block extracts all required control information from its input data at run time and decides about its functional behavior. Again we updated our models to support this feature.

We validated the correctness of the basic functionality of our SRE models using numerical simulation and comparison with the corresponding ST functional models. We simulated each of the models for 100 test vectors with a random selection of operation modes. All of the models gave the same numerical values to the one transmitted by the original ST Model.

Finally, we generated symbolic traces for those models. Now, we have three SRE models of the WiMax modem, each one can run in any of the supported seven operation modes. So, we modified our symbolic simulation script so that it enumerated all supported control scenarios and generated 7 symbolic traces for each model. We call this "mixed simulation mode" because it uses both symbolic and numerical simulation to generate the symbolic traces.

Table 4.4 shows the time and memory utilization of those experiments, which shows the superiority of symbolic simulation over numerical simulation in terms of time requirements.

Table 4.5 Summarizes the coding requirements of SRE models in this phase with respect to the number of code lines and the number of SRE equations in the system.

Table 4.4: Numerical and Mixed Simulation (Multiple Control Scenario)

Model	Run Time (sec)	Memory (MB)
SRE Functional Level (FL) - Numerical Simulation, 100 iterations	10.1	10.11
SRE Functional Level (FL) - Mixed Simulation	2.05	11.96
SRE Process Transfer Level (PTL-8) (8 processors) - Numerical Simulation, 10 iterations	211.3	10.00
SRE Process Transfer Level (PTL-8) (8 processors) - Mixed Simulation	33.70	12.91
SRE Process Transfer Level (PTL-4) (4 processors. + scheduler) - Numerical Simulation, 10 iterations	222.3	11.44
SRE Process Transfer Level (PTL-4) (4 processors. + scheduler) - Mixed Simulation	34.18	13.37

Table 4.5: Model Coding Requirements (Multiple Control Scenario)

Model	SRE Eqns	Code Lines
SRE Functional Level (FL) (1 Processor)	8	290
SRE Process Transfer Level (PTL-8) (8 processors + FIFO)	46	840
SRE Process Transfer Level (PTL-4) (4 processors + FIFO + scheduler)	50	940

4.3 Equivalence Checking

In this part of the project we verified the computational equivalence between SRE models of different levels of abstraction. We applied the Pattern Matching techniques on the symbolic traces calculated by symbolic simulation.

We used Mathematica Pattern Matching built-in function to compare symbolic traces as described in Algorithm 3.2. We conducted four equivalence checking

experiments to prove the following relations:

- Equivalence of Functional Model and FIFO based Process Transfer Model in the Single Control scenario.
- Equivalence of FIFO based Process Transfer Model and FIFO and Scheduler based Process Transfer Model in the Single Control scenario.
- Equivalence of Functional Model and FIFO based Process Transfer Model in the Multiple Control scenario.
- Equivalence of FIFO based Process Transfer Model and FIFO and Scheduler based Process Transfer Model in the Multiple Control scenario.

The results show that the SRE models at different levels of abstraction are functionally equivalent. Therefore we concluded that the corresponding C models are also functionally equivalent.

In order to grantee the basic functionality of our equivalence checking algorithm, we injected one bug in one of the SRE models and reran the symbolic simulation and equivalence checking experiment. The bug was injected in the SRE FIFO based Process Transfer Model. We changed the functional description of the mapping block. Then we ran the equivalence experiments again. Now, the results showed non equivalence between the models and returned the nonequivalent symbols from the model's symbolic trace. By inspecting those symbols we found that they were generated only at three modes of operation (0, 1, or 2) From these results and by looking at Table 2.1 we concluded that the bug was injected in the mapping block implementation only when its puncturing value equals 1/2.

Tables 4.6 and 4.7 summarize the performed equivalence and non equivalence experiments along with their time and memory utilization results. By studying those tables we conclude the following:

- The run time of the experiments is linearly proportional to the number of control scenarios. This is interesting because other techniques have exponential increase in time requirements when we increase the execution paths.
- Memory requirements of various experiments are comparable to each other.
- Since our verification technique depends on pattern matching we obtained interesting results in the case of non-equivalence. Both time and memory requirements stayed at the same rank as in the equivalence experiments.

Table 4.6: Equivalence Checking Experiments

Control Scenario	Experiment	Run Time (sec)	Memory (MB)	Result
Single	Functional Level (FL). vs. Process Transfer Level (PTL-8)	5.17	19.59	Equivalent
Single	Process Transfer Level (PTL-8) vs. Process Transfer Level (PTL-4)	5.26	28.31	Equivalent
Multiple	Functional Level (FL) vs. Process Transfer Level (PTL-8)	33.70	12.91	Equivalent
Multiple	Process Transfer Level (PTL-8) vs. Process Transfer Level (PTL-4)	34.18	13.37	Equivalent

Table 4.7: Equivalence Checking Experiments - Injected Bug

Control Scenario	Experiment	Run Time (sec)	Memory (MB)	Result
Single	Functional Level (FL). vs. Process Transfer Level (PTL-8)	4.96	20.88	Not-Equiv
Single	Process Transfer Level (PTL-8) vs. Process Transfer Level (PTL-4)	6.21	27.21	Not-Equiv
Multiple	Functional Level (FL) vs. Process Transfer Level (PTL-8)	32.10	14.51	Not-Equiv
Multiple	Process Transfer Level (PTL-8) vs. Process Transfer Level (PTL-4)	30.26	15.67	Not-Equiv

It is to be noted that the results in Tables 4.6 and 4.7 show the time and memory utilization for the complete equivalence checking process (i.e. symbolic trace computation and pattern matching).

4.4 Property Checking

The purpose of this part of the project is to use the Property Checking algorithm (Algorithm 3.3) to verify the conformance of the WiMax models to some important properties of the system and generate counterexample for properties that are verified to be false.

We divided the properties with respect to their scope into three categories:

1. Global Properties: specify a functionality of the whole system.
2. Local Properties: specify a functionality of a single block.
3. Control Properties: specify a functionality of a single control configuration (Code Type in the WiMax case)

We wrote a sample property of each of those main categories:

- P1: Eventually all Input Data Bits will be transmitted
- P2: Eventually all Input Data Bits with the positions specified by the randomizer bit list will be flipped.
- P3: Eventually the Appropriate Puncturing Function will be applied to all Convolution Coded Data Bits in the same order.

Next, we translated those properties into SRE (see Appendix A.5 for a sample property written in a form of SRE). After that we applied our proposed Property Checking Algorithm to verify their correctness according to the symbolic traces calculated in the symbolic simulation of the model under test. The results of our

experiments show that all tested properties are verified to be true under the three models in both single and multiple control scenarios. This shows the conformance of the corresponding C models from STMicroelectronics to the verified properties.

We also repeated our experiments after injecting the following bugs into all the SRE models.

1. Cut one of the data lines between two of the internal blocks.
2. Changed the randomizer reference array.
3. Changed the condition checker at the mapping block that specifies the block behavior when the control scenario changes.

The results of the simulation detected all the bugs and returned counterexamples that specify the failed property and print the wrong signal value.

Tables 4.8, 4.9, 4.10 and 4.11 show all property checking experiments results, together with their time and memory requirements. By looking at those results we can conclude the following:

1. The run time of the experiments is linearly proportional to the number of control scenarios.
2. Memory requirements of various experiments are close to each other.
3. Both time and memory requirements stayed at the same rank in both cases, verified true and verified false cases.

Note that the results in Tables 4.8, 4.9, 4.10 and 4.11 show the time and memory utilization for the complete property checking process (i.e., symbolic trace computation and pattern matching).

Table 4.8: Property Checking (Single Control Scenario)

Experiment	Property	Run Time (sec)	Memory (MB)	Result
SRE Functional Level (FL)	1	3.25	22.05	True
	2	3.10	23.10	True
	3	2.92	22.65	True
SRE Process Transfer Level (PTL-8)	1	10.52	22.81	True
	2	11.10	20.60	True
	3	10.30	13.80	True
SRE Process Transfer Level (PTL-4)	1	10.16	23.84	True
	2	10.23	21.02	True
	3	10.98	22.36	True

Table 4.9: Property Checking (Multiple Control Scenario)

Experiment	Property	Run Time (sec)	Memory (MB)	Result
SRE Functional Level (FL)	1	25.70	20.15	True
	2	24.32	21.23	True
	3	24.10	23.10	True
SRE Process Transfer Level (PTL-8)	1	75.20	26.20	True
	2	80.12	23.28	True
	3	78.58	16.55	True
SRE Process Transfer Level (PTL-4)	1	79.95	22.23	True
	2	78.55	21.25	True
	3	72.00	20.36	True

Table 4.10: Property Checking (Single Control Scenario) - Injected Bug

Experiment	Property	Run Time (sec)	Memory (MB)	Result
SRE Functional Level (FL)	1	3.00	22.23	False
	2	3.58	21.25	False
	3	3.10	20.36	False
SRE Process Transfer Level (PTL-8)	1	10.12	20.15	False
	2	10.22	21.23	False
	3	11.11	23.10	False
SRE Process Transfer Level (PTL-4)	1	10.16	26.20	False
	2	11.22	23.28	False
	3	11.94	16.55	False

Table 4.11: Property Checking (Multiple Control Scenario) - Injected Bug

Experiment	Property	Run Time (sec)	Memory (MB)	Result
SRE Functional Level (FL)	1	30.71	22.81	False
	2	25.22	20.60	False
	3	26.10	13.80	False
SRE Process Transfer Level (PTL-8)	1	76.90	22.05	False
	2	78.15	23.10	False
	3	79.80	22.65	False
SRE Process Transfer Level (PTL-4)	1	80.94	22.81	False
	2	79.78	20.60	False
	3	71.02	13.80	False

4.5 Summary

In this chapter we have discussed the details of the modeling and verification of the STMicroelectronics WiMax Transceiver. The effort can be divided into three main phases. First, model realization, which includes the translation of ST C++ models into SRE models. Second is the symbolic simulation experiments that are used to generate symbolic traces based on the models developed in the first phase. The last phase is the verification experiments which in turn can be divided into equivalence checking and property checking. In the verification section we have also discussed error injection experiments, in which we introduce a bug in the model design and rerun the verification algorithm. In addition, we discussed the memory consumption and execution time for each phase of the process to be able to measure the performance of the verification methodology used.

The experimental results in this chapter showed the successful application of the multilevel verification approach discussed in Chapter 3 on STMicroelectronics WiMax PHY implementation. The performance characteristics of the symbolic modeling and verification experiments showed its advantage over its numerical counterpart.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The System-on-Chip design process is becoming increasingly complex and more challenging because of the increasing demand from applications running on those systems. To overcome this complexity in the design process, system designers have started using an efficient multilevel design approach, in which, they start the design process with a high level description of the system that captures its functionality (called executable specs). Then, they refine this design into lower level design descriptions that have more architectural and implementation details, until the physical implementation of the system is achieved.

On the other hand, this increasing design complexity makes it more critical to find an optimal verification methodology. This verification process should be efficient as the verification process is already claiming most of the scheduled time in the SoC implementation. In addition it should have adequate coverage for all design decisions involved in the design process. And, more importantly, this verification methodology should be integrated with the design process from the beginning to allow early detection of design bugs.

The main objective of this thesis is to verify the functional and architectural

level implementation models of a WiMax IEEE 802.16 transceiver that was provided by STMicroelectronics, who adopted a multi-level design approach to design this WiMax modem, in which multiple C++ system descriptions at different levels of abstraction were provided.

To perform the system level functional verification of the WiMax modem, we have proposed a semi-formal verification strategy that uses symbolic simulation based equivalence and property checking techniques. We used system of recurrence equations to model both the systems under test and the key system properties. SRE is very powerful in describing functionality of complex systems at higher levels of abstraction. With SRE, it was also very easy to write interesting system properties with different scopes.

The equivalence of two models was verified by translating both models from C++ to SRE. Then, symbolic simulation was performed on both models. Finally, pattern matching was used to check for equivalence and return counterexamples in case of non equivalence. Besides, the modeling flexibility, symbolic simulation of SRE system description was much faster than numerical simulation. In addition, the powerful mathematical reasoning infrastructure in symbolic algebra made symbolic simulation one of the best candidates to answer our equivalence checking question, by performing pattern matching and equation solving on the system's symbolic traces.

In the context of property checking, we blended our SRE properties with the system model in SRE and ran symbolic simulation in Mathematica 6.0 environment. Then, again pattern matching and equation solving were used to verify the correctness of the properties and return counterexamples in case the properties fail to prove.

We successfully applied our methodology to verify the WiMax system design from STMicroelectronics at three different levels of abstraction, one functional level model and two architectural level models. Our experimental results showed that the three provided models are functionally equivalent which means that the design

decisions made in the design refinement process are correct and were implemented correctly. In addition, the results showed that all models do conform with the specified properties. We were also able to detect some manually injected bugs in those models and counterexamples were provided. The performance measurements showed in general that the used symbolic simulation based verification is much more efficient than numerical simulation.

The main advantages of our proposed methodology are the following:

- The equivalence checking between models at different levels of abstraction is a very important concept that enables the verification process to be integrated with the multi-level design methodology.
- The flexibility and ease of describing systems at system level using SRE. This step can easily be done by the modeling designers.
- The speed of the symbolic simulation which quickens the whole verification process. This is important, especially, that this process is repeated at each design iteration or whenever a new system level description is introduced in the design process.

5.2 Future Work

The performance of our methodology is very good when verifying heavy computational systems at higher levels of abstraction. However, one limitation in the existing SRE symbolic simulation algorithms is that simulation time grows exponentially with number of control signals in the simulated model [1]. This makes this technique more suitable for computational intensive systems rather than control intensive systems. In this thesis, we used mixed simulation technique, described in Chapter 4, to overcome this growth in simulation time and increase efficiency. However, one possible improvement is to investigate other techniques to reduce symbolic

simulation time with multiple control signals.

Another possible future work is to define transition rules to translate system descriptions from standard programming languages such as C or C++ to SRE. This will enable automating the modeling part of the methodology.

The semi-formal verification methodology used in this thesis can also be applied to other more complex system designs from STMicroelectronics or other companies.

Appendix A

Appendix : Sample Mathematica Code

A.1 Sample Mathematica Recurrence Equations

An example of recurrence equations used in the WiMax model written in Mathematica

```
For [i = 1, i < NBDataBlocks + 1, i++, PunctOutputN[[i]] =
    If[And[PunctCount != 0 , ExecPhase[[4]] == 2,
      i == (NBDataBlocks - PunctCount + 1)], PunctOut, PunctOutput[[i]] ];
];

For[i = 0, i < Ncbps, i++,
  InterleavOutN[[(IntegerPart[(((IntegerPart[((Ncbps/DD)* Mod[i, DD] +
       (i/DD))])/SS)*SS + Mod[((IntegerPart[((Ncbps/DD)*Mod[i, DD] + (i/DD))])] + Ncbps -
       (DD*(IntegerPart[((Ncbps/DD)*Mod[i, DD] + (i/DD))])/Ncbps)), (SS)])]) + 1]] =
       If[And[InterleavCount != 0, ExecPhase[[5]] == 1], InterleavIn[[i + 1]],
         InterleavOut[[(IntegerPart[(((IntegerPart[((Ncbps/DD)* Mod[i, DD] +
          (i/DD))])/SS)*SS + Mod[((IntegerPart[((Ncbps/DD)*Mod[i, DD] + (i/DD))])] + Ncbps -
          (DD*(IntegerPart[((Ncbps/DD)* Mod[i, DD] + (i/DD))])/Ncbps)), (SS)])]) + 1]]];
];
```

A.2 FIFO's Tail Index Update Using Recurrence Equations

A code fragment that is used to model the WiMax FIFO tail update using SRE in Mathematica.

```
FIFOsTailN[[1]] =
  If[And [SourceCount != 0, ExecPhase[[1]] == 2 ,
     FIFOsTail[[1]] < FIFONBCells], FIFOsTail[[1]] + 1,
   If[And [SourceCount != 0, ExecPhase[[1]] ==   2 ,
      ExecPhase[[2]] ==  0, FIFOsTail[[1]] == FIFONBCells],FIFOsTail[[1]] - 1,
    If[And [SourceCount != 0, ExecPhase[[1]] !=  2 ,
       ExecPhase[[2]] ==  0, FIFOsTail[[1]] != 0], FIFOsTail[[1]] - 1,
     If[And[SourceCount == 0, RandCount != 0 , ExecPhase[[2]] == 0,
       FIFOsTail[[1]] != 0], FIFOsTail[[1]] - 1, FIFOsTail[[1]] ]]]];
```

A.3 Dynamic Scheduler in Recurrence Equations

A code fragment that is used to model the WiMax dynamic scheduler using SRE in Mathematica

```
CPUBlockN[[2]] =
If[And [RandCount != 0, ExecPhase[[2]] == 2 , FIFOsTail[[2]] < FIFONBCells], 2,
     If[And [CCCount != 0, ExecPhase[[3]] == 2 , FIFOsTail[[3]] < FIFONBCells], 3,
         If[And [PunctCount != 0, ExecPhase[[4]] == 2 , FIFOsTail[[4]] < FIFONBCells], 4,
             If[And [InterleavCount != 0, ExecPhase[[5]] == 2 ,
                 FIFOsTail[[5]] < FIFONBCells], 1, CPUBlock[[2]]]]]];


CPUBlockN[[3]] =
If[And [RepetitionCount != 0, ExecPhase[[6]] == 2 , FIFOsTail[[6]] < FIFONBCells], 2,
   If[And [FraminCount != 0, ExecPhase[[7]] == 2 ], 1, CPUBlock[[3]]]];
```

A.4 Traffic Generator Sets FEC Code Type

A code fragment that is used to model traffic generator in Mathematica

```
If[  FECMode  == 0,
  CodingRate = WMRATE12; (*WMRATE12,WMRATE23,WMRATE34*)
  NbBitMapping = WMQPSK;
  ,
  If[ FECMode ==  1,
    CodingRate = WMRATE34; (*WMRATE12,WMRATE23,WMRATE34*)
    NbBitMapping = WMQPSK;
    ,
    If[FECMode == 2,
      CodingRate = WMRATE12; (*WMRATE12,WMRATE23,WMRATE34*)
      NbBitMapping = WM16QAM;
      ,
      If[FECMode == 3,
        CodingRate = WMRATE34; (*WMRATE12,WMRATE23,WMRATE34*)
        NbBitMapping = WM16QAM;
        ,
        If[FECMode == 4,
          CodingRate = WMRATE12; (*WMRATE12,WMRATE23,WMRATE34*)
          NbBitMapping = WM64QAM;
          ,
          If[FECMode == 5,
            CodingRate = WMRATE23; (*WMRATE12,WMRATE23,WMRATE34*)
            NbBitMapping = WM64QAM;
            ,
            If[FECMode == 6,
              CodingRate = WMRATE34; (*WMRATE12,WMRATE23,WMRATE34*)
              NbBitMapping = WM64QAM;
              ,
              Print ["Invalid FEC Mode !!!"]
              ];
            ];
          ];
        ];
      ];
    ];
  ];
```

A.5 Sample Property in Recurrence Equation

An example of writing system properties using recurrence equations in Mathematica

```
If[ CodeRate ==  WMRATE23,

 For [i = 0, i < CycleCounter, i++,
  If[ PunctOutput[[1, i*3 + 1]] == CCOutput[[1, i*4 + 1]],
   PuncturedSymbols ++,
   Print["Symbol Not Punctured Properly : PunctOutput [[", i*3 + 1,
     "]] is not correct"];,
   Print["Symbol Not Punctured Properly : PunctOutput [[" , i*3 + 1,
     "]] is not correct"];
   ];
  If[ PunctOutput[[1, i*3 + 2]] == CCOutput[[1, i*4 + 2]],
   PuncturedSymbols ++,
   Print["Symbol Not Punctured Properly : PunctOutput [[", i*3 + 2,
     "]] is not correct"];,
   Print["Symbol Not Punctured Properly : PunctOutput [[" , i*3 + 2,
     "]] is not correct"];
   ];
  If[ PunctOutput[[1, i*3 + 3]] == CCOutput[[1, i*4 + 4]],
   PuncturedSymbols ++,
   Print["Symbol Not Punctured Properly : PunctOutput [[", i*3 + 3,
     "]] is not correct"];,
   Print["Symbol Not Punctured Properly : PunctOutput [[" , i*3 + 3,
     "]] is not correct"];
   ];
  ],
```

Bibliography

[1] G. Al Samman. "Simulation Symbolique des Circuits Decrits au Niveau Algorithmique." PhD Thesis, Universite Joseph Fourier Grenoble 1, July 2005.

[2] T. Matsumoto, H. Saito and M. Fujita. "An Equivalence Checking Method for C Descriptions Based on Symbolic Simulation with Textual Differences." in Proceedings of the IASTED International Conference on Advances in Computer Science and Technology, vol. 88, no. 12, Mar. 2004.

[3] G. Ritter, H. Eveking and H. Hinrichsen. "Formal verification of designs with complex control by symbolic simulation." in Proceedings of the 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods, 1999, pp. 234-249.

[4] S. Wolfram. Mathematica: A System for Doing Mathematics by Computer. Redwood City: Addison-Wesley Longman Publishing Co, 1991.

[5] G. S. V. Radha Krishna Rao and G. Radhamani. WiMAX a Wireless Technology Revolution. New York: Auerbach Publications, 2007.

[6] M. Fujita, I. Ghosh and M. Prasad. Verification Techniques for System-Level Design. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 2008.

[7] R. v. Nee and R. Prasad. OFDM for Wireless Multimedia Communications. Norwood, MA, USA: Artech House, Inc, 2000.

[8] C. Eklund, K. L. Stanwood, S. Wang and R. B. Marks. "IEEE Standard 802.16: A technical overview of the WirelessMAN Air Interface for broadband wireless access." IEEE Communications Magazine, vol. 40, pp. 98-107, 2002.

[9] B. A. Cipra. "The Ubiquitous Reed-Solomon Codes." SIAM News, vol. 26, 1993.

[10] Rohde & Shwarz. "WiMAX - General Information about the Standard 802.16 - Application Notes." pp. 34, 2010. "http://www2.rohde-schwarz.com/file_1782/1MA96" [Accessed April/2012].

[11] S. Abdi and D. Gajski. "Verification of system level model transformations." International Journal of Parallel Programming, vol. 34, pp. 29-59, Feb. 2006.

[12] A. K. Deb, A. Jantsch and J. Oberg. "System design for DSP applications in transaction level modeling paradigm." in Proceedings of the 41st Annual Design Automation Conference, San Diego, CA, USA, June 2004, pp. 466-471.

[13] F. Qiao, P. Lin, J. Yu, K. Dong and Z. Wang. "System level modeling of WiMAX SoC system." in Proceedings of the 7th International Conference on ASIC, Guillin, China, 2007, pp. 946-949.

[14] J. He, J. S. Yang, Y. Kim and A. S. Kim. "System-level Time-domain Behavioral modeling for a Mobile WiMax Transceiver." in Proceedings of the 2006 IEEE International Behavioral Modeling and Simulation Workshop, San Jose, California, 2006, pp. 138-143.

[15] G. Al-Sammane, M. H. Zaki and S. Tahar. "A symbolic methodology for the verification of analog and mixed signal designs." in Proceedings of the Conference on Design, Automation and Test in Europe, Nice, France, 2007, pp. 249-254.

[16] M. H. Zaki, G. Al-Sammane and S. Tahar. “Formal verification of analog and mixed signal designs in Mathematica.” in Proceedings of the 7th International Conference on Computational Science, Part II, Beijing, China, 2007, pp. 263-267.

[17] STMicroelectronics. “Model-based mapping to parallel architecture of 802.16a (WiMax).” Ottawa, Canada, 2007.

[18] STMicroelectronics. “WiMAX wireless-MAN OFDMA functional refinement.” Ottawa, Canada, 2007.

[19] STMicroelectronics. “Multiflex WiMax OFDMA Library 2007 - Code package.” Ottawa, Canada, 2007.

[20] E. M. Clarke, O. Grumberg and D. A. Peled. Model Checking. Cambridge, MA, USA: MIT Press, 1999.

[21] D. Cyrluk, S. Rajan, N. Shankar and M. K. Srivas. “Effective theorem proving for hardware verification.” in Proceedings of the Second International Conference on Theorem Provers in Circuit Design - Theory, Practice and Experience, 1994, pp. 203-222.

[22] R. E. Bryant. “Symbolic simulation - techniques and applications.” in Proceedings of the 27th ACM/IEEE Design Automation Conference, Orlando, Florida, United States, 1990, pp. 517-521.

[23] Altera. “DSP Builder.” 2009. “http://www.altera.com/products/software/products/dsp/dsp-builder.html” [Accessed April/2012].

[24] B. Alizadeh and Z. Navabi. “Word-level symbolic simulation in processor verification.” Computers and Digital Techniques, IEE Proceedings, vol. 151, pp. 356-366, Sept. 2004.

[25] A. Chiang, W. Han and B. Kapoor. "Validating physical access layer of WiMAX using SystemVerilog." in Proceedings of the 2009 10th International Symposium on Quality of Electronic Design, 2009, pp. 356-359.

[26] Agilent Technologies. "Move Forward to What's Possible in WiMAX." pp. 8, 2009. "http://cp.literature.agilent.com/litweb/pdf/5989-5914EN.pdf" [Accessed April/2012].

[27] A. N. M. Abdullah, B. Akbarpour and S. Tahar. "Formal analysis and verification of an OFDM modem design using HOL." in Proceedings of the Formal Methods in Computer Aided Design, 2006, pp. 189-190.

[28] A. Meyer. Principles of Functional Verification. Burlington, MA, USA: Newnes - Elsevir Science, 2003.

[29] S. Liao, S. Tjiang and R. Gupta. "An efficient implementation of reactivity for modeling hardware in the scenic design environment." in Proceedings of the 34th Annual Design Automation Conference, Anaheim, California, United States, 1997, pp. 70-75.

[30] S. Swamy, A. Molin and B. Covnot. "OO-VHDL: Object-Oriented Extensions to VHDL." IEEE Computer, vol. 28, pp. 18-26, Oct. 1995.

[31] N. Medvidovic and R. N. Taylor. "A Classification and Comparison Framework for Software Architecture Description Languages." IEEE Transactions on Software Engineering, vol. 26, pp. 70-93, Jan. 2000.

[32] E. M. Clarke, O. Grumberg and D. E. Long. "Verification tools for finite-state concurrent systems." in the proceedings of REX School/Symposium (A Decade of Concurrency, Reflections and Perspectives), 1994, pp. 124-175.

[33] J. Fernandez, A. Kerbrat and L. Mounier. “Symbolic equivalence checking.” in Proceedings of the 5th International Conference on Computer Aided Verification, 1993, pp. 85-96.

[34] F. Laroussinie, N. Markey and P. Schnoebelen. “On model checking durational kripke structures.” in Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures, 2002, pp. 264-279.

[35] D. D. Gajski, J. Zhu, R. Dmer, A. Gerstlauer and S. Zhao. SpecC: Specification Language and Methodology. Massachusetts: Kluwer Academic Publishers, 2000.

[36] System Verilog. “SystemVerilog Overview.” 2008. ”http://www.systemverilog.org/” [Accessed April/2012]

[37] IEEE standard. “IEEE Standard for Property Specification Language (PSL)(IEEE Std 1850-2005).” pp. 1-188, 2005.

[38] IEEE standard. “IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1 (IEEE Std 802.16e-2005 and IEEE Std 802.16-2004/Cor 1-2005).” pp. 1-822, 2006.

[39] F. Franek, C. G. Jennings and W. F. Smyth. “A simple fast hybrid pattern-matching algorithm.” Jornal of Discrete Algorithms, vol. 5, pp. 682-695, Dec. 2007.

[40] L. Bachmair and H. Ganzinger. “Rewrite-based Equational Theorem Proving with Selection and Simplification.” Journal of Logic and Computation, vol. 4, pp. 217-247, July 1994.

[41] N. Dunstan and F. Ivan. “Process scheduling and UNIX semaphores.” Softwarepractice and Experience, vol. 25, pp. 1141-1153, Oct. 1995.

[42] G. Nicolescu and P. J. Mosterman. Model-Based Design for Embedded Systems (Computational Analysis, Synthesis, and Design of Dynamic Systems). New York: CRC Press Taylor and Francis Group, 2010.