Formal Analysis of Steady State Error in Feedback Control Systems

By

Muhammad Ahmad

2010-NUST-MS-EE(S)-18

Supervisor

Dr. Osman Hasan

Department of Electrical Engineering

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Electrical Engineering (MS EE)

In

School of Electrical Engineering and Computer Science,

National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(July 2014)

Approval

It is certified that the contents and form of the thesis entitled "**Formal Analysis of Steady State Error in Feedback Control Systems** " submitted by **Muhammad Ahmad** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Osman Hasan**

Signature: _____

Date: _____

Committee Member 1: **Dr. Ammar Hasan**

Signature: _____

Date: _____

Committee Member 2: **Dr. Sohail Iqbal**

Signature: _____

Date: _____

Committee Member 3: **Dr. Adnan Khalid Kiani**

Signature: _____

Date: _____

Dedication

To My Family

Certificate of Originality

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Muhammad Ahmad**

Signature: _____

Acknowledgment

I have been very fortunate to have Dr. Osman Hasan as my supervisor. I am deeply grateful for his support throughout my research work.
I would also like to acknowledge the help of my lab fellows especially Muhammad Usman and Muhammad Binyameen.
I also wish to express my gratitude to Adnan Rashid from SAVE lab for his support.
This journey would not have been possible without the support of my parents. Thank you for believing in me and wanting the best for me.
To my sisters, thank you for your encouragement and for the invaluable moral support which helped me in stressful time.

Abstract

The meticulousness of safety critical control system design has always been of vital significance as even a trifling glitch in the design analysis may result in grievous penalties, even in the loss of human life. To ensure the correctness of the system, traditionally it based on paper and pencil proofs. With the beginning of the modern age, simulations and computer algebra systems were employed to ensure the correctness of design. However, the said techniques compromise the accuracy of the analysis and thus are not a viable option for the analysis of safety critical systems. Over the years Formal Verification has emerged as one of the most viable option for the verification of hardware and software systems. It combines the strong paper and pencil mathematics along with modern computer aided tool to ensure correctness. Since the new formal model must satisfy the existing mathematical reasoning, therefore the chance to catch critical design errors increases which are often ignored otherwise. This thesis makes use of formal methods to establish a higher-order logic based framework to reason about the correctness of safety critical control systems.

In particular, we first provide a generic platform to express a wide variety of control systems in terms of their formal model. We identified the least basic blocks which can be arranged in various topologies to express a wide range of systems. We then provide theorems which can be used to reason

various properties of these systems. Moreover, a generic expression for the steady-state error of unity-feedback control systems using the multivariate analysis theories of HOL-Light has been verified. The unique feature of this expression is that it can be used to reason about the steady-state error of any system type and input. Moreover, it facilitates reusability when reasoning about the state-state error of the same system while considering different inputs. The quest for minimizing the user interaction in the higher-order-logic theorem-proving based analysis for steady-state errors led us to develop this useful relationship, which to the best of our knowledge has not been reported in the control systems literature before. In order to illustrate the utilization and practical effectiveness of the formalization for verifying real-world control systems, two case studied have been conducted. The steady-state error analysis of the Pulse Width Modulation (PWM) push-pull DC-DC converters, which is used in power electronics and many safety-critical aerospace applications and the steady-state error analysis of a Solar Tacking control systems developed for aerospace applications.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Control systems [26] form an integral part of all automated systems used in a wide range of safety-critical applications, including industrial automation, surgical robots, automobiles and aerospace systems. Accurate modeling and analysis of such safety critical systems is of paramount importance since a minor design flaw can result in severe penalty, even in the loss of human life. The dire need for accurate analysis have given rise to various techniques. These techniques have evolved over the time. Traditionally paper-and-pencil based was adapted which was quite accurate. However as the systems grew in size as well as in complexity, thus resulted in the scalability issue that it became difficult to ensure the correctness of large mathematical proofs which were greatly influenced by risk of human error. The advent of computers led to a new era of modeling and analysis. A large number of techniques were introduced which can be broadly categorized as simulation based techniques and Computer Algebra Systems (CAS). The availability of fast computational resources led the researchers to work in simulations based analysis

using the numerical methods. The essence of simulation based techniques is to build a discretized model of the system and then calculate the output of the system for a set of given inputs patterns. For small systems, all the input test vectors were simulated and the output was reasoned to be correct or false. However, with the rise in complexity of the systems and multiple inputs and outputs, this model rendered insufficient to ensure the required output behavior pattern of the system. Also the discretization was erroneous since it had floating point and rounding-off errors. Computer Algebra Systems (CAS) on the other hand ensured better results than numerical methods based simulations. In this approach, a mathematical model corresponding to the real world system is made and verified. But these systems also lack absolute correctness since computed results are not always mathematically correct. The accurate analysis is thus the main concern since erroneous analysis may lead to fatal consequences. To further assert on this point we quote the Air France flight 447 crash which resulted in the loss of 228 lives. The crash of Arian 5 in 1996 was due to data conversion error, which resulted in loss of over 500 million US$. Thus, due to the mentioned limitations, we cannot rely on these techniques for design and analysis of safety critical control systems.

In order to overcome the above-mentioned limitations, the usage of formal methods [12] in the safety-critical domain of control system analysis is recommended and is also being investigated [29, 4]. Formal methods bridges the gap between the traditional strong mathematical paper-and-pencil approach along with modern Computer Aided Tool (CAD). The main theme is to construct a mathematical model corresponding to the real world system and then reason the correctness of mathematics which inturn increases the probability of finding design errors since an error will be violating some well

defined mathematical principle. There are two formal method techniques i.e., model checking [20] and higher-order-logic theorem proving [11]. In model checking, a finite state machine of the system is constructed and reasoned upon and is thus an automated verification technique. Whereas, theorem proving is much more expressive in nature and is an interactive verification technique. Both these techniques have been used to accurately analyze hardware and software systems. Since it is of extreme importance to ensure the correctness of design, therefore formal method techniques must be used for safety-critical control system applications. However, due to the continuous nature of our safety-critical systems, and the use of transcendental functions, the automated finite state machine based approach such as model checking can not be used in this domain.

The aim of this thesis is to present a basic platform for the formal reasoning of control systems using higher-order logic theorem proving. This approach is modular and the platform can be extended to represent further more types of control systems and also reason about their properties.

1.2 Control Systems

In this section we present a general introduction to control systems. All real world systems need to be controlled in order to perform the required tasks. Thus control system theory plays a vital role in the correct operation of the systems. These control systems work along with the given systems (plants) and are designed in such a way that they ensure the desired behavior of their corresponding systems while adhering to the stability constraints and allowable error margins. We shall now give a brief description of the basic building blocks. Later we will discuss the steady state error analysis.

1.2.1 Building Block Approach

We focus on the laplace or s-domain analysis of control system and follow the modular approach that the complete system can be divided into smaller subsystems which are connected in a regular fashion to represent the required functionality. The advantage of this approach is that we can identify the least number of these smaller blocks required to represent a wide range of control systems. From formal methods point of view it is also very advantageous since we can model systems using the same formalized blocks rather than writing new definitions and theorems for every new system. We term these as basic building blocks which can then be connected in a way to represent a model of real world system. A pictorial view of the identified basic building block is shown in Fig, 3.1. *Cascaded blocks, summation block, pickoff point block* and *feedback block* have been identified as the basic blocks and their formalization is also presented in Chapter 3.

Control systems can be generally configured in two topologies i.e., open loop topology or a closed loop topology [26]. In open-loop systems, the controller generates the control signals based on a reference or input signal

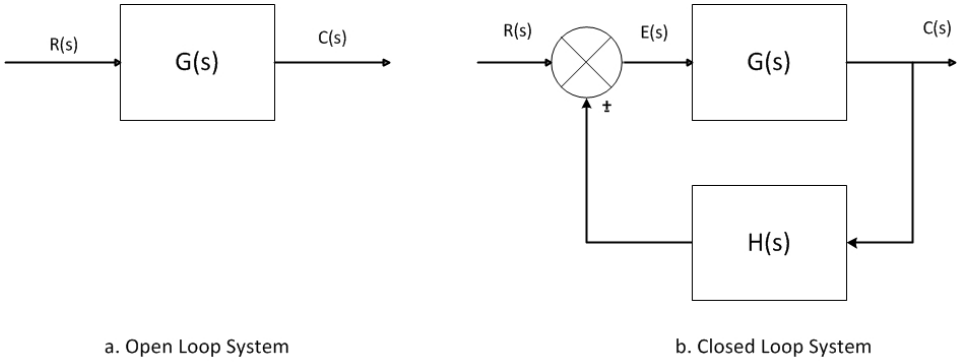

Figure 1.1: Control System Configurations

$R(s)$, which acts upon the plant or the system $G(s)$ as shown in Fig. 1.a . A disadvantage of this kind of configuration is that there is no feedback coming to the controller and the controller has no information about the output of the plant. Another disadvantage to this topology is that the controller will generate fixed pattern and will not be able to cope with unexpected disturbances that may occur due to any reason. a disturbance can occur due to some mechanical part of the plant such as rotor assembly or it may occur due to the effect of an unwanted external force applied. To overcome this limitation, control systems are often configured in a feedback or closed loop pattern where the output of the plant $C(s)$ is measured and compared with a reference or an input signal $R(s)$, as shown in Fig. 1.b. This error signal $E(s)$ is then used for decision making in the controller to compensate for disturbances. A unity-feedback is a frequently used closed-loop system where the output of the system is compared with the reference input signal as is, i.e., without any gain or loss in the feedback path.

1.2.2 Steady State Error

The steady state error gives a parametric measure that how well a system will respond to disturbances and is thus also a benchmark for the controllability of a control system designed [27]. It tells the response of a closed-loop system when steady-state conditions have been reached.

We will focus on the steady state error for unity-feedback control systems. The analysis is performed in Laplace domain due to the advantage that the main system can be represented in terms of connected sub-systems. The net transfer function for unity-feedback error model is mathematically expressed as[26]:

$$E(s) = \frac{R(s)}{1 + G(s)} \tag{1.1}$$

Here $G(s)$ is the net transfer function of the plant. $R(s)$ represents the input to the system, which is traditionally taken to be as the *unit step* ($\frac{1}{s}$), *ramp* ($\frac{1}{s^2}$) and *parabola* ($\frac{1}{s^3}$) functions. The above mentioned expression for error model have been obtained by solving the feedback block along with arithmetic simplifications.

The steady-state error is measured at time t tends to infinity. By applying the Final Value Theorem, the Laplace domain representation of steady state error analysis for unity feedback system is:

$$e_\infty = \lim_{s \to 0} sE(s) \tag{1.2}$$

Traditional methods, like paper-and-pencil proof methods and computer simulations and numerical methods, cannot guarantee the accuracy of the above-mentioned steady-state error analysis. The paper-and-pencil based analysis methods are error prone due to the human involvement. Moreover, it is quite often the case that many key assumptions of the results obtained of sub-system using paper-and-pencil proof methods are not documented, which may lead to to erroneous systems. Computer simulations and numerical methods, such as MathWorks Simulink [25], cannot guarantee accurate results while dealing with feedback-control systems mainly due to their inherent non-exhaustive nature coupled with the imprecision of computer arithmetics. The mathematical models of control systems can also be analyzed in computer algebra systems (CAS), such as Mathematica [22]. CAS are very efficient for computing mathematical solutions symbolically, but are also not completely reliable due to the presence of unverified huge symbolic manipulation algorithms. In order to overcome the above-mentioned limitations,

the usage of formal methods in the safety-critical domain of control system analysis is increasingly being investigated [29, 4].

1.3 Literature Review

This section presents some important research contributions in formalization of control systems. First we mention ClawZ [5] which allows us to translate models of control systems developed in MathWorks Simulink into Z language specifications, which are then verified by proving the equivalence of the controller implementation using Ada in ProofProver. Another similar approach is presented in [1] in which the author translates the discrete-time Simulink model to Circus notations, which combines Z language and refinement calculus and then compares a parallel Ada implementation. An interesting methodology adopted in [8] calls for using the Timed Interval Calculus (TIC) library to capture the behavior of Simulink blocks, which could be verified in a theorem prover. A similar approach was adapted by Mahony, and modeling and analysis of feedback control systems was introduced using the DOVE environment [24]. Model checking has also been successfully used to analyze dynamic systems by abstracting the behavior of the system to a state-space model [31]. Herencia-Zapana [18] proposed to formally analyze control software properties by first expressing the stability proofs as C code annotations and then translating them to PVS proof obligations and automatically verifying them. However, all these pioneering frameworks are based on automatic formal verification tools and thus require some sort of abstraction mechanism to model the exact behavior of real-world control systems and their environments, which are always continuous in nature.

In order to formally model and analyze continuous models of control

systems, Boulton et al. provided some reasoning support for verifying frequency response of continuous-time control systems using Hoare logic using the HOL98 theorem prover [7]. The main idea is to reason about the gain and phase relationships of a control system using the gain and phase relationships of its subsystems in the block diagram. This framework does not provide generic functions to model arbitrary block diagrams for control systems and also lacks reasoning support for complex number analysis principles, such as limits and summation, which are essential to reason about many control system design related parameters, such as steady-state errors and stability. In order to overcome these shortcomings, Boulton et al [6] proposed to use automated symbolic methods to replace the classical graphical charts, such as Nichole and Bode plots along with their formal models. Based on this principle, the authors developed a prototype tool using Maple and the PVS system. Maple is used to compute the verification conditions for the given control system and PVS is used to discharge these conditions using theorem proving principles. Due to the usage of Maple, the accuracy of the analysis is again somewhat compromised as has been mentioned above.

The foremost foundation of analyzing the steady-state error of control systems is the formalization of complex number analysis theories. The multivariate calculus theories of HOL-Light theorem prover [16] fulfill this requirement. These theories have been recently used to formalize the basic building blocks of control systems [17] and the Laplace theory [30], which are the most relevant contributions to our work. We build upon and enhance the results reported in [17] to analyze steady-state errors of unity-feedback control systems and facilitate the formal reasoning process by verifying a generic expression for steady-state error in this paper. The recent formalization of Laplace theory[30] opens up many interesting research directions

in the context of our work since now we can link our formalization to the time-domain as well.

1.4 Proposed Framework

The motivation of the thesis is to develop a theorem proving based formalized control system framework, which can model and reason the properties of real world safety critical control systems. We require the system to have the following features:

1. To formally express a given real world system into its equivalent higher-order logic based formal model using the basic building blocks.

2. To be able to formally verify the properties of the control systems in higher-order logic theorem prover. These properties can be used to write theorems corresponding to the characteristics of interest.

3. To formally reason about the correctness of the steady state error analysis of the formalized system.

Figure 1.2 outlines the main theme of the theorem proving based formalization of control system analysis. On top left corner we see the two boxes which represents the control system along with the set of properties which needs to be verified.

First of all, the control system must be expressed in terms of its formal model. For this purpose we adapt the modular approach i.e., the system can be split into smaller modules, connected in the right pattern to replicate the same behavior. This approach is feasible in the Laplace domain which make use of complex number theories. The HOL Light Multivariate complex theories provide a strong basis for complex number analysis. Now

we built upon these theories to formalize the basic building blocks such as pickoff point, feedback block etc as described in the previous sections. Once the basic building blocks are formalized, then the given control system can be translated into its formal model. The next step is to utilize these formal models to write theorems corresponding to the properties such as the equivalence or the steady state error property to be verified.

In order to conduct a theorem proving based verification of control systems, we need to verify the higher-order logic theorems in a theorem prover. It would come in handy to have some classical results already verified so that we may directly use them instead of proving them every time for a new system. To fulfil this requirement, we have formalized some essential properties such as the simplification of feedback loop. We have also verified generic theorems for the steady state error property. This exercise will aid in reducing the human machine interaction required for the verification and will also

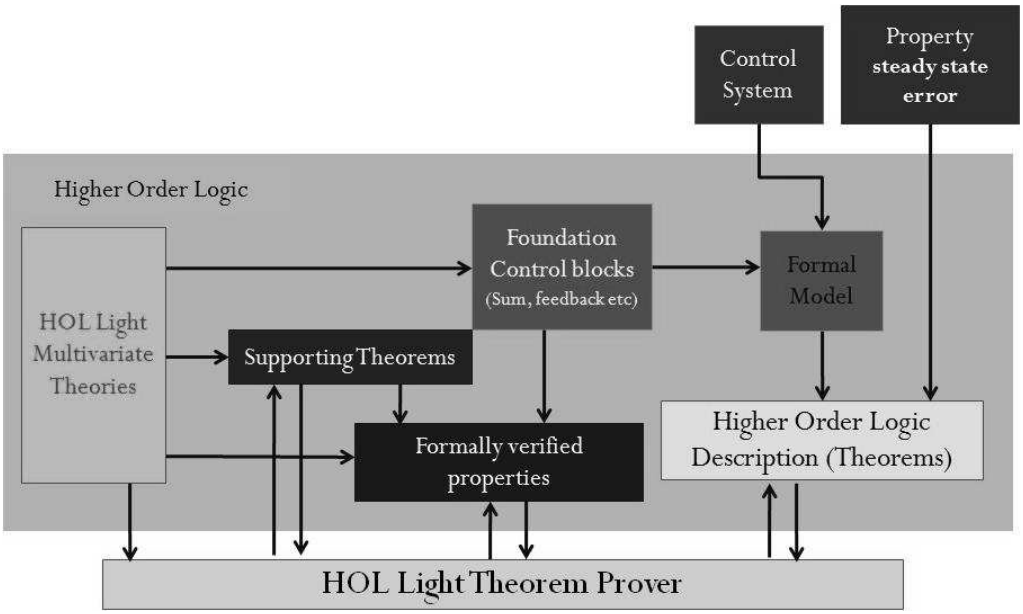

Figure 1.2: Proposed Framework

speed up the process. The outcome of the exercise will be the formal proof that certify that the given system properties are valid.

1.5 Thesis Contribution

The main scope of this research work is to establish a framework for the formal analysis of control systems using theorem proving. The proposed framework allows us to reason about the control systems accurately and thus is a viable option for the verification of safety critical applications. The contributions are summed as follows:

- A generic infrastructure have been proposed which allows us to formally model a wide range of safety critical control systems. The infrastructure also allows s to formally specify and verify higher-order logic theorems corresponding to various properties of control systems.

- It also presents the higher order logic formal verification of steady state error analysis for unity feedback control systems. Some helping tactics have also been formalized to help minimize the user interaction required for the formal analysis.

- The formal reasoning of few real-world control system applications have also been presented in the end to judge the correctness and usefulness of the proposed framework. We have presented the block diagram simplification of Pulse Width Modulation (PWM) push-pull dc-dc converter. Also, we have presented the formal analysis of the steady-state error analysis of the converter. Similarly we also present the formal reasoning of solar tracking control system which is used in aerospace applications. The approach used for the analysis purpose of the case studies is quite

general and can be applied to other real-world safety critical systems as well.

1.6 Organization of Thesis

The rest of the thesis is organized as follows. In Chapter 2, a brief introduction to the HOL-Light theorem prover and an overview of Multivariate Complex Theories to equip the reader with some notation and concepts that are going to be used in the rest of this thesis. Chapter 3 describes the formalization of block diagram representation of control systems along with their formal verification of the related properties in HOL-Light. The formalization of Steady-state error analysis is presented in Chapter 4. The necessary properties and functions required to formalize any given control system have been described. The formalization steps of steady-state error analysis make use of the formalization of control system block diagram representation presented in Chapter 3. In order to demonstrate the practical usefulness of the proposed methodology, two case studies namely formalization of steady state error analysis of Pulse Width Modulation push-pull dc-dc converter and formalization of steady-state error analysis of solar tracking control system have been presented in Chapter 5. Lastly, Chapter 5 concludes the thesis and points out some future research areas.

Chapter 2

Preliminaries

This chapter provides with a basic overview of the HOL theorem porver and the Multivariate complex theories used in our proposed framework. The aim is to introduce the basic working knowledge of the notations and terms in HOL which are used in the theories.

2.1 Theorem Proving

Theorem proving is an emerging research area in *automated reasoning*. Automated reasoning is concerned with the construction of mathematical theorems of a given problem which can then be verified using computer programs. The mathematical theories can be build upon various types of logic, such as, first-order logic , propositional logic, or higher-order logic, depending upon the requirements of the system. However, given the increasing complexity of today's systems, higher-order logic deems to be a viable choice. The essence of theorem proving is to mathematically model the given system in an appropriate formal logic and then formalize the properties in such a way as they can be verified using computer based formal reasoning. The use of higher-

order logic theorem proving for formal modeling of the system behaviors is advantageous since it has a wide variety of quantifiers and an expressive nature which can be used to model any real world physical systems which can be expressed mathematically. Every theorem prover core constitutes of some well axioms and inference rules. The soundness of correctness of theorem proving based verification is ensured in such a way that each new theorem must be created from the basic axioms and inference rules. The new theorems created can then be used for the verification of new theorems while adhering to the basic axioms and inference rules.

Theorem provers or proof assistants are tools which facilitate in the formalization of a system at hand by expressing them in mathematical notations which can be used in computer programs. A broad categorization of theorem prover is automatic and interactive theorem provers. An automatic proof assistant performs various proof tasks automatically while an interactive theorem prover requires significant user-computer interaction. Some well known automatic theorem provers are MetiTarski [3], SATURATE, Gandalf, LeanTAP, SETHEO, METEOR and Otter [15]. Interactive theorem prover includes HOL, HOL-Light, Isabelle, Coq, ProofPower and MIZAR [15]. This thesis make use of HOL-Light theorem prover to execute the formalization of control system block diagram representation and steady state error analysis. The logic behind the preference is the rich complex number theories which are essential for the research.

2.2 HOL Light Theorem Prover

HOL Light belongs to the HOL family of theorem provers. The first version of HOL is HOL88 followed by subsequent versions i-e, HOL90 and HOL98

and HOL4. It is an interactive theorem prover which follows the LCF [23] approach which is similar to the deduction system in which every minute detail must be declared in detail. It aids in writing the given system and its properties in form of mathematical properties in metalanguage ML [28] to automate the inferences and hence. The HOL family implements a version of λ-calculus which follows the Church[9] approach. HOL Light consists of ten primitive rules of inference. HOL Light has been used for the formal verification for both hardware and software.

2.2.1 Terms

Terms are manipulated as symbolic expressions. The type of terms includes constants, variables and lambda-terms. Variables begins with letter and may contain digits in them e.g. L, C, rc. The variables needs to be bounded by the quantifiers which intern depends on which theory they belong to. λ-terms also known as lambda abstractions are used for representing functions. $\lambda x.f(x)$ is representing a function which has an argument x and will return $f(x)$.

2.2.2 Types

Every term defined in HOL Light must have a type according to lambda calculus implemented. This type can be one of the basic types or it can be the outcome of applying a type constructor to the already defined types. Each constant and variable needs a type to be defined with them. There can be two variables with same name but different type. In this case both will be considered two different variables and operations from the concerned theory will be applicable to them respectively. It is also possible to mention the type of the term explicitly, e.g $(x : real)$ or $(y : complex)$.

2.2.3 Inference Rules

Inference rules are represented as ML functions and they are basic procedures required for deriving new theorems. HOL Light has a set of ten primitive inference rules. Any new theorem must be derived from these inference rules. These rules are *Reflexivity*, *Transitivity*, *Congruence*, *Abstraction*, *Beta-conversion*, *Assumption introduction*, *Substitution*, *Discharging an assumption*, *Type instantiation*.

2.2.4 Theorems

A theorem represents a formal statement that can be an axiom or it follows from theorems by an inference rule. A theorem constitutes of a finite set of boolean terms Ω called assumptions and a boolean term S called the conclusion. Any new theorem must be build upon the already proved theorems thus satisfying the inference rules presented above.

2.2.5 Theories

A HOL Light theory comprises of a set of type, type operators, definitions, axioms and proved theorems. A theory can also include other available theories as well. These theories can be loaded by the user to utilize the available theorems and definitions. The objective is to eradicate the duplication of the work already done and then build upon the already formalized HOL Light theories. In this thesis HOL theories of Booleans, complex number, real numbers, limits and transcendental functions are utilized. In fact, one of the reason to select HOL-Light theorem prover was to make use of these already formalized theories.

2.2.6 Proofs in HOL Light

Generally two approaches are followed for proof methods : forward and reverse approach. In the forward approach, one has to start from the inference rules and then build upon them to reach the required goal. This is not an easy approach since it requires an extensive knowledge of the theories and the system at hand. In the reverse or backward approach one start from the goal to be proved and then using the formalized theories and primitive inference rule, split the goal into smaller subgoals to be proved. These subgoals when proved stacks to the main goal and when all these are proved, then the main goal is also proved. There are some automatic solvers also available which can work with certain theories which aids the user to solve the desire goals. In interactive theorem proving, the user has to interact with the proof editor and using the right tactic, direct the proof to the end. Although some of the basic simplifications are catered by the HOL Light theorem prover but other requires extensive user interaction.

2.2.7 HOL Light Notations

The Table 2.1 provides the mathematical interpretation of some HOL Light functions and symbols used in the thesis. These notations will be widely seen in the formalization to come in the later chapters. The purpose to mention them here is to get the reader handsomely equipped with the terminologies to come in the thesis.

Table 2.1: HOL Light Symbols and Functions

HOL Light Symbol	Standard Symbol	Meaning
/\	and	Logical *and*
\/	or	Logical *or*
~	not	Logical *negation*
==>	$\longrightarrow$	Implication
<==>	$=$	Equality
!x.t	$\forall x.t$	for all $x : t$
?x.t	$\exists x.t$	for some $x : t$
@x.t	$\epsilon x.t$	an x such that : t
λx.t	$\lambda x.t$	Function that maps x to $t(x)$
suc n	$(n + 1)$	Successor of natural number
exp x	e^x	Exponential function
sqrt x	$\sqrt{x}$	Square root function
abs x	$\|x\|$	Absolute function

2.3 Multivariate Calculus Theories in HOL-Light

In this section, we give a brief introduction to the existing multivariate complex theories in the HOL Light theorem prover. The intent is to get the reader some working knowledge of the definitions and notations used, to make the thesis self contained and thus facilitate its understanding for a wider audience, including both formal methods and control communities.

A n-dimensional vector is represented as a $\mathbb{R}^n$ column matrix of real

numbers in HOL-Light. All of the vector operations are then handled as matrix manipulations. This way, complex numbers can be represented by the data-type $\mathbb{R}^2$, i.e, a column matrix having two elements [13]. In this formalization of complex numbers, the first real number represents the real part and the second real number represents the imaginary part of the given complex number[14]. The main advantage of this choice is that all the topological and analytic formalization developed for vectors is inherited by the complex numbers.

Definition 1: *Re and Im*

```
⊢ ∀ z.  Re z = z$1
⊢ ∀ z.  Im z = z$2
```

The functions Re and Im accepts complex number and then return its real and imaginary parts, respectively. The notation z$n represents the n^{th} component of a vector z.

Definition 2: *Cx*

```
⊢ ∀ a.  Cx a = complex(a,&0)
```

The Cx function accepts a real number and return the corresponding complex number. Since the real number has a zero imaginary part, hence we have a &0 in the imaginer part. Here **&** operator is used which is used to map a natural number to its corresponding real number.

Definition 3: *Complex Number*

```
⊢ ∀ x y.  complex (x,y) = vector [x; y]
```

The above mentioned definition returns the complex and imaginary part of complex number in its equivalent vector representation.

Definition 4: *Complex Square-root*

```
⊢ ∀ z.  csqrt z =
        (if Im z = &0
        then if &0 <= Re z
             then complex (sqrt (Re z),&0)
             else complex (&0,sqrt (--Re z))
        else complex (sqrt ((norm z + Re z) / &2),
        Im z / abs (Im z) * sqrt ((norm z - Re z) / &2)))
```

cqrt functions defines the complex square-root. If the imaginary part of the complex number is zero then the complex square-root is the same as real number square root.

In the above mentioned definition, norm is used. This norm is actually the normalization of a complex number which is also a widely used phenomena and has been formalized in HOL-Light [14] as follows:

Definition 5: *Normalization of a Complex Number*

```
⊢ ∀ z.  norm z = sqrt (Re z pow 2 + Im z pow 2)
```

where `sqrt` represents the HOL-Light square root function for real numbers and **pow** is the power.

The concept of limit of a function is used in our formalization to model the steady-state error and is formalized in HOL-Light as follows:

Definition 6: *Limit of a function*

```
⊢ ∀ f net.  lim net f = (@l.  (f → l) net)
```

The function `lim` is defined using the Hilbert choice operator `@` in the functional form. It accepts a *net* with elements of arbitrary data-type A and a function f, of data-type $A \to \mathbb{R}^m$, and returns l:$\mathbb{R}^m$, i.e., the value to which the function f converges to at the given net.

Definition 7: *Limit of a constant*

⊢ ∀ `net a. ((λx. a) → a) net )`

This definition states that the limit of a constant is always equal to the the constant.

Similarly, we also use the following theorem which formalizes the sum of geometric progression in our development:

Theorem 1: *Sum of a geometric Progression*

⊢ ∀ `z. norm z < &1 ⇒`

`((λk.z pow k) sums z pow n / (Cx(&1) - z)) (from n)`

Where the function `f sums k (from n)` ensures that the infinite summation of a multivariate sequence f is equal to k with n as the starting point.

Chapter 3

Formalization of Basic Building Blocks

This chapter provides a set of formal definitions of the basic building blocks of control systems, given in Fig. 3.1. Identification of the least number of basic building blocks to represent a wide variety of control systems was of utmost importance. After consulting the literature and control system experts, the following blocks have been identified as the basic building block which can be used to represent a wide variety of real world control systems. The definition corresponding to each of the basic block diagram have been formalized and presented in [17] . For facilitation purpose, simplification theorems have also been verified and are mentioned.

3.1 Formal Definitions of Block Diagrams

We start with introducing the basic building blocks and then present their formal definitions in higher-order logic are provided as follows.

The net transfer function of n subsystems connected in *cascade* is the

product of their individual laplace transfer functions (Fig. 3.1.a).

Definition 3.1: *Cascaded Subsystems*

```
⊢ series [] = Cx (&1) ∧
                (∀ h t.  series (CONS h t) h * series t)
```

The function `series` accepts a list of complex numbers, corresponding to the transfer functions of all the given subsystems, and recursively returns their product by calling the series function till the list goes empty. Here two type injections `&` and `Cx` are used to transform a positive integer to its corresponding real and complex number, respectively. **CONS** is a list constructor with **h** as first element and **t** as last element of list.

Fig. 3.1.b depicts a *summation junction* of transfer functions where the net transfer functions of a set of incoming branches is formed by adding their individual transfer functions. The formalization of this behavior accepts a list of complex numbers and returns their sum.

Definition 3.2: *Formalization of Summation Junction*

```
⊢ sum_junction [] = Cx (&0) ∧
  (∀ h t.  sum_junction(CONS h t) h + sum_junction t)
```

The *pickoff point* represents a subsystem connected to a network of parallel branches of subsystems (Fig. 3.1.c):

Definition 3.3: *Formalization of Pickoff point*

```
⊢ ∀ A h t.  pickoff A [] = [] ∧
            pickoff A (CONS h t) = CONS (h * A) (pickoff A t)
```

The function `pickoff`, accepts a complex number `A`, corresponding to the transfer function of the first subsystem, and a list of complex numbers, corresponding to the transfer functions of all the subsystems in the branches,

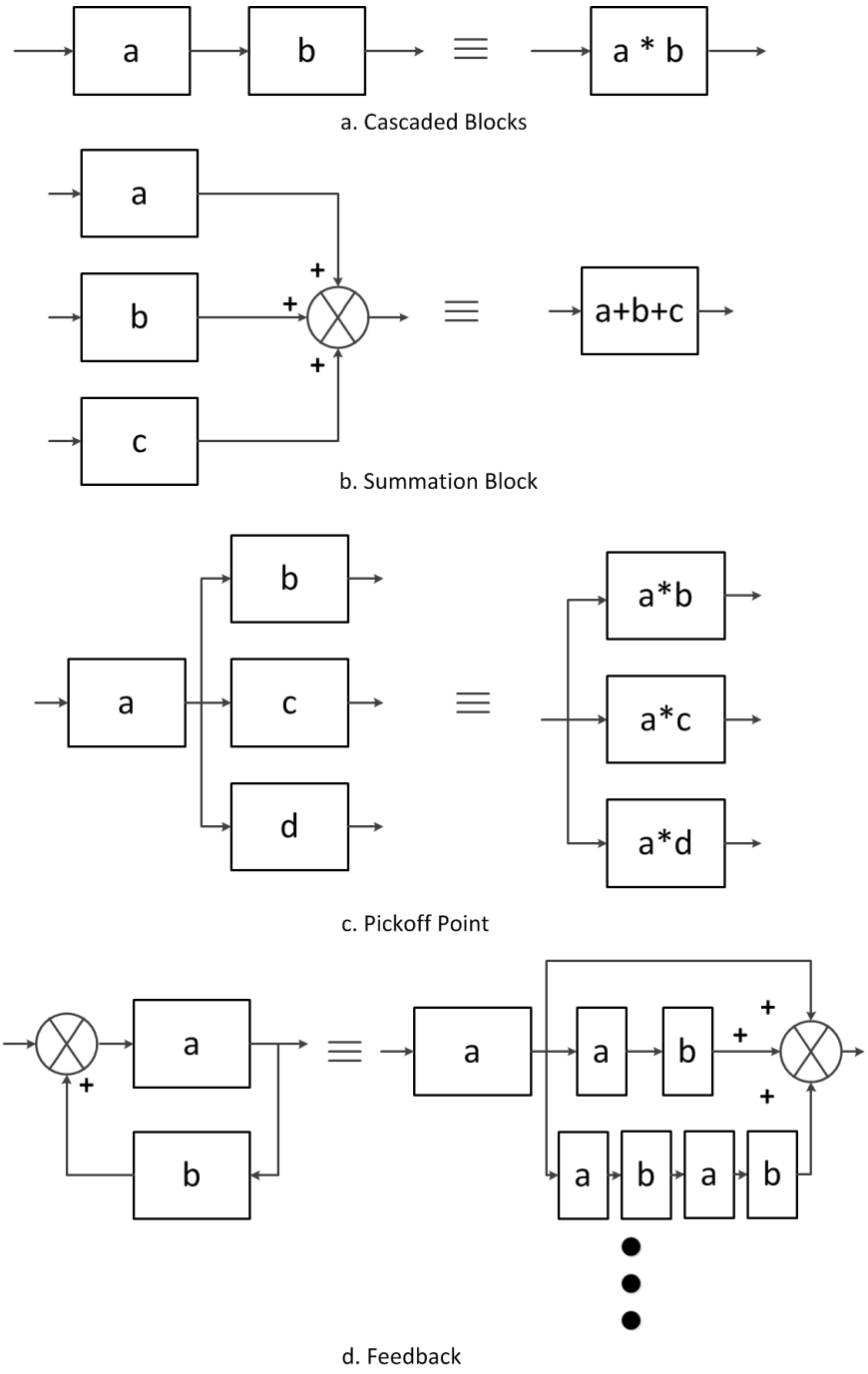

Figure 3.1: Basic Building Blocks of a Control System

and the recursively returns a list of complex numbers corresponding to the equivalent block diagram.

The *feedback* block (Fig. 3.1.d), is the foremost element required to model closed-loop control systems. Due to the feedback signal, it primarily represents an infinite summation of branches that comprises of serially connected subsystems.

Definition 3.4: *Branch of a Feedback Loop*

```
⊢ ∀ a b n.  feedback_branch a b 0 = Cx (&1) ∧
            feedback_branch a b (SUC n) = series [a; b] *
            (feedback_branch a b n)
```

The function `feedback_branch` accepts the forward path transfer function `a`, the feedback path transfer function `b` and the number of the branches `n`. It returns the net transfer function for n branches of a feedback loop as a single complex number. Now, the infinite summation of all branches of the feedback loop can be modeled as the following HOL-Light function:

Definition 3.5: *Feedback Loop*

```
⊢ ∀ a b.  feedback_loop a b = (infsum (from 0)
                               (λk.  feedback_branch a b k))
```

The HOL-Light function `infsum (from n) f` above provides the infinite summation a multivariate sequence f with n as the starting point. Now, we can model the behavior of the feedback loop in HOL-Light as follows:

Definition 3.6: *Feedback Loop*

```
⊢ ∀ a b.  feedback a b = series [a; (feedback_loop a b)]
```

The function `feedback` accepts the forward path transfer function `a` and the feedback path transfer function `b` and returns the net transfer function

by forming the series network of the summation of all the possible infinite branches and the final forward path transfer function, since the output is taken after the forward path `a`.

3.2 Simplification Theorems

Now we present some theorems used for the simplification of the model of a given control system formalized using the above mentioned definitions of basic blocks. For feedback definitions, we have verified a simpler theorem which is more familiar and commonly used form in the control system domain. The feedback related definitions mentioned in the last section formalized the mathematics behind the feedback loop simplification. However, control system experts are not familiar with so much extensive mathematics of the end result. Therefore, for the facilitation purpose of control system experts, we present the simplification theorem:

Theorem 3.1: *Feedback loop simplification*

```
⊢ ∀ a b.  norm (a * b) < &1 ⇒
            feedback a b = a / (Cx(&1) - a * b)
```

The proof of Theorem 3.1 is primarily based on the infinite summation of a geometric series in the feedback loop [14], given in Theorem 2.1 along with certain arithmetic simplifications while adhering to the mentioned assumption. The assumption norm (a * b) < &1 is there to ensure the convergence of the geometric series.

Often is the case when there are multiple feedback paths converging. For analysis purpose some feedback loops need to be simplified while keeping others in the feedback loop. Depiction of such a scenario is illustrated in Fig.

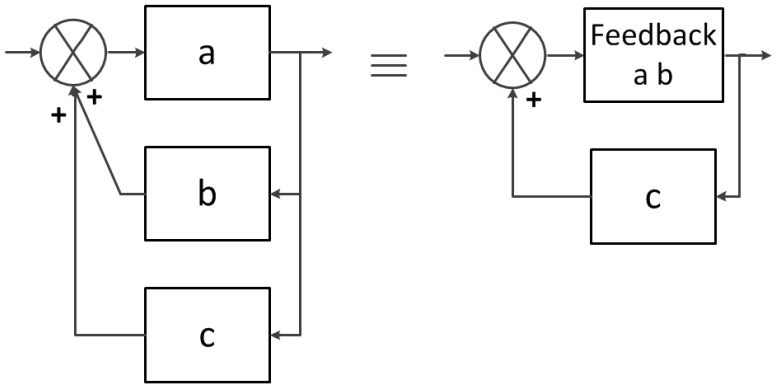

Figure 3.2: Multiple Feedback Simplification Rule

3.2. The equivalence relationship between the block diagrams, shown in , has been formally verified as follows:

Theorem 3.2: *Feedback loop simplification*

```
∀ a b c.  (norm (a*b) + norm (a*c)) < &1⇒
    feedback a (sum_junction (pickoff Cx(&1) [b;c])) =
                feedback (feedback a b) c
```

The proof of Theorem 3.2 utilizes Theorem 3.1 along with some complex arithmetic reasoning. This theorem can also be used to convert any non-unity-feedback control system into a unity-feedback control system required for steady-state error analysis.

In this chapter the formal definitions of the basic building blocks have been presented. Some useful theorems for simplification purpose have also been reported. Now using the mentioned formalization in this chapter, the formalization of steady-state error will be conducted.

Chapter 4

Formalization of Steady-State Error

In this chapter we shall describe the higher order logic formalization of steady-state error analysis by utilizing the formalization of basic blocks of control system. The information is arranged into two section. First section describes the mathematical foundation and the declaration of notations required for the formalization. Then we shall proceed with the higher order logic definitions and theorems in the subsequent section.

4.1 Mathematical Analysis

The steady-state analysis of unity-feedback control systems can be performed in the Laplace domain because this choice allows us to model the main system in terms of the transfer functions of its sub-systems, as a block diagram. The formal definitions of the blocks can then be used from previous chapter. The steady-state error representation in terms of block diagram is shown in Fig. 4.1. Where E(s) is the steady-state error, G(s) is the transfer function of the

plant or system and R(s) is the input type.

The overall transfer function of the plant $G(s)$ is then expressed as follows by manipulating the transfer functions of its subsystems using a set of predefined rules[26]:

$$G(s) = \frac{1}{s^b}\frac{Y(s)}{Z(s)} \tag{4.1}$$

where the integer variable $b : 0, 1, 2 \cdots$ categorizes the system type or the number of integrators in the forward path[26], and $Y(s)$ and $Z(s)$ represent the zeros and poles of $G(s)$ apart from $\frac{1}{s^b}$, respectively. Now, the net transfer function for unity-feedback error model is mathematically expressed as[26]:

$$E(s) = \frac{R(s)}{1 + G(s)} \tag{4.2}$$

where $R(s)$ models the input to our system, which in the case of steady-state error analysis is traditionally taken to be as the *unit step* ($\frac{1}{s}$), *ramp* ($\frac{1}{s^2}$) and *parabola* ($\frac{1}{s^3}$) functions. The steady-state error is measured at a very large time, i.e., when the time t tends to infinity. Thus, it can be defined in the Laplace domain by applying the Final Value Theorem to the error model:

$$e_\infty = \lim_{s \to 0} sE(s) \tag{4.3}$$

This is the expression which we shall use and thus will formalize it. The following section comprises the formalization of steady state error analysis.

4.2 Formal Analysis

We now present the formal verification of a generic expression that can be used to reason about the steady-state error of any unity-feedback system (Fig. 4.1), irrespective of its type and input. We proceed in this direction by first

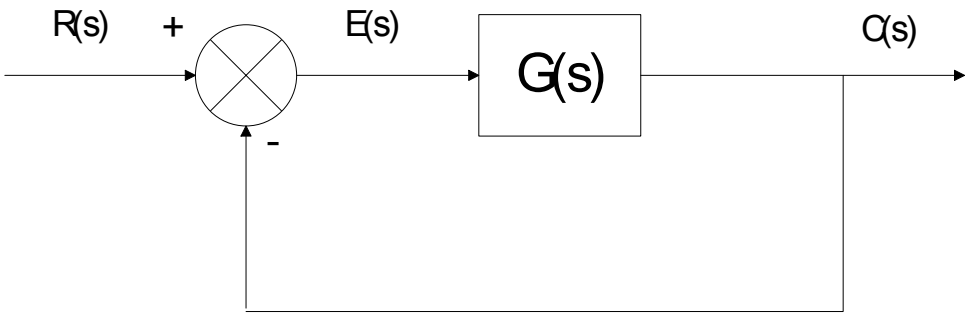

Figure 4.1: Steady-State Error of Unity-Feedback Systems

formalizing a generalized representation of the transfer function according to Equation (4.1).

Definition 4.1: *General Transfer function*

```
⊢ ∀ Y Z a.  general_tf Y Z b = (λs.  Y s / (s pow b * Z s))
```

The function `general_tf` accepts two complex functions Y and Z of data type $\mathbb{R}^2 \to \mathbb{R}^2$ along with a complex number b which represents the system type and returns the transfer function using the lambda abstraction format.

Now, the error model of unity-feedback systems in terms of the generalized representation of $G(s)$, according to Equation (4.2), is as follows:

Definition 4.2: *Steady-state-error-model*

```
⊢ ∀ G a.  uf_error_model G a =
        (λs.  series [Cx (&1) /s pow a; feedback_loop (G s)
                        (--Cx(&1))])
```

The function `uf_error_model` accepts a variable $G : \mathbb{R}^2 \to \mathbb{R}^2$, which represents the general transfer function, and a complex number $a : \mathbb{R}^2$, which generalizes the input type, i.e., if the input is a unit step then $a = 1$ and similarly $a = 2$ and $a = 3$ for the ramp and parabola inputs, respectively.

The function uses the functions `series` and `feedback` to capture the structure of the error model of the unity-feedback system, depicted in Fig. 4.2, and returns its net transfer function with data type $\mathbb{R}^2 \to \mathbb{R}^2$.

Now, the steady-state error can be formally defined as the limit of the net transfer function of the error model, as given in Equation (4.3),

Definition 4.3: *steady-state-error*

```
⊢ ∀ E. steady_state_error E = lim (at (Cx(&0))) (λs.  s (E s))
```

where the function `lim(at(vec i))(λx.f x)`, represents the limit of a function f at point i, i.e., $\lim_{x \to i} f(x)$ in HOL-Light. The function `steady_state_error` accepts a variable $E : \mathbb{R}^2 \to \mathbb{R}^2$, which represents the net transfer function of the error, and returns its corresponding steady-state error as a complex value.

4.3 Steady state error analysis simplification theorem

Now, using the above mentioned definitions, one can formalize the steady state error analysis of a given control system. But for facilitation, we have verified a generic expression for the steady state error as follows:

Theorem 4.1: *Unity-feedback steady-state error*

```
⊢ ∀ Y Z a b l m.
   (∀ s.  Z continuous at s) ∧
   ¬(l = Cx(&0)) ∧
   ¬(m + l = Cx(&0)) ∧
   0 ≤ b ∧
   1 ≤ a ∧
```

```
a ≤ b+1 ∧
(Y → l) (at Cx(&0)) ∧
(Z → m) (at Cx(&0)) ∧
¬(Z (Cx(&0)) = Cx(&0)) ∧
(?k.  &0 < k ∧ (∀s.  norm s < k ⇒
          norm (Y s /(s pow b * Z s)) < &1))
⇒ steady_state_error (uf_error_model (general_tf Y Z b) a) =
   (if b = 0 then m / (m+l) else if a = b+1 then m / l
     else Cx(&0))
```

The first three assumptions are used to avoid singularities. The next two assumptions declare the allowable ranges of the system type and input characterization variables, respectively. The next assumption (a $\leq$ b+1) defines the upper bound of the input type based on the type of the system. The next two assumptions ensure that the variables, l and m, represent the limiting values of the functions Y and Z at point 0, respectively. The next assumption declares that the initial value of Z(s) is not zero. Finally, the last assumption is required for the feedback simplification. To the best of our knowledge, this relationship between the type of the system and its allowable input, given in Theorem 4.1, is not mentioned in most of the control systems literature. To ascertain our finding, we consulted some control systems experts and they confirmed our results. Missing such corner cases is a common problem in paper-and-pencil based mathematical analysis and simulation and is one of the major causes for faulty system designs. The proof of Theorem 4.1 is based on various properties of limit of a complex function and complex arithmetic reasoning.

The formalization presented so far in this section consumed about 300 man-hours, which are mainly spent in the user guided verification due to the

undecidable nature of the higher-order logic. Our proof script is available at [2]. The main benefit of this development, however, is that it greatly facilitates the formal reasoning about unity-feedback control system properties by reducing the human interaction in such proofs, as will be illustrated in the next section.

It is important to note that the universal quantification over the variables Y, Z, a and b in Theorem 4 allows us to use this result for reasoning about steady-state error of any unity-feedback control system irrespective of its type, input and behavior. To the best of our knowledge, such a generic relationship for the steady-state error for unity-feedback systems has not been reported in the control systems literature.

Now, the step-wise process for reasoning about the steady-state error of unity-feedback systems using Theorem 4 is outlined here. The first step is to use the formal definitions, given in Chapter 3 , to develop a formal model of the given system using its structural description. Then verify the equivalence of this model and the expression `general_tf Y Z b`, by choosing appropriate assignments of the functions Y and Z and the variable b. Next, express the theorem for the steady-state error of the given unity-feedback system: `steady_state_error (uf_error_model (<transfer function of the given system>) a) = <steady state error>`. Now, using Theorem 4 along with the fact that all of its assumptions hold for the given values of Y, Z, a and b, we can conclude the proof of steady-state error of the given unity-feedback system. In order to illustrate the effectiveness and practical utilization of Theorem 4.1 and the above mentioned process, we analyze a real-world control system in the next chapter.

Chapter 5

Case studies

In order to test the effectiveness and utilization of our proposed framework, we apply it to real-world control systems. First case study we present is the formalization of Pulse Width Modulation (PWM) push-pull dc-dc converter which is an integral part of various electronic modules used in applications such as aerospace. Next we present the formal analysis of a solar tracking control system used in satellites to keep the soar panels in optimum position to get the maximum effeciency of solar cells. To the best of our knowledge, such formal analysis has not been reported in the academic society.

5.1 Pulse Width Modulation (PWM) push-pull dc-dc converter

The Pulse Width Modulation(PWM) push-pull dc-dc converters are widely used to step down dc voltages and thus have many applications in areas, like aerospace applications, where dc voltage is produced and consumed. The steady-state response of this electronic device is of utmost importance and thus has been extensively studied [19, 10]. A commonly used model [10] for

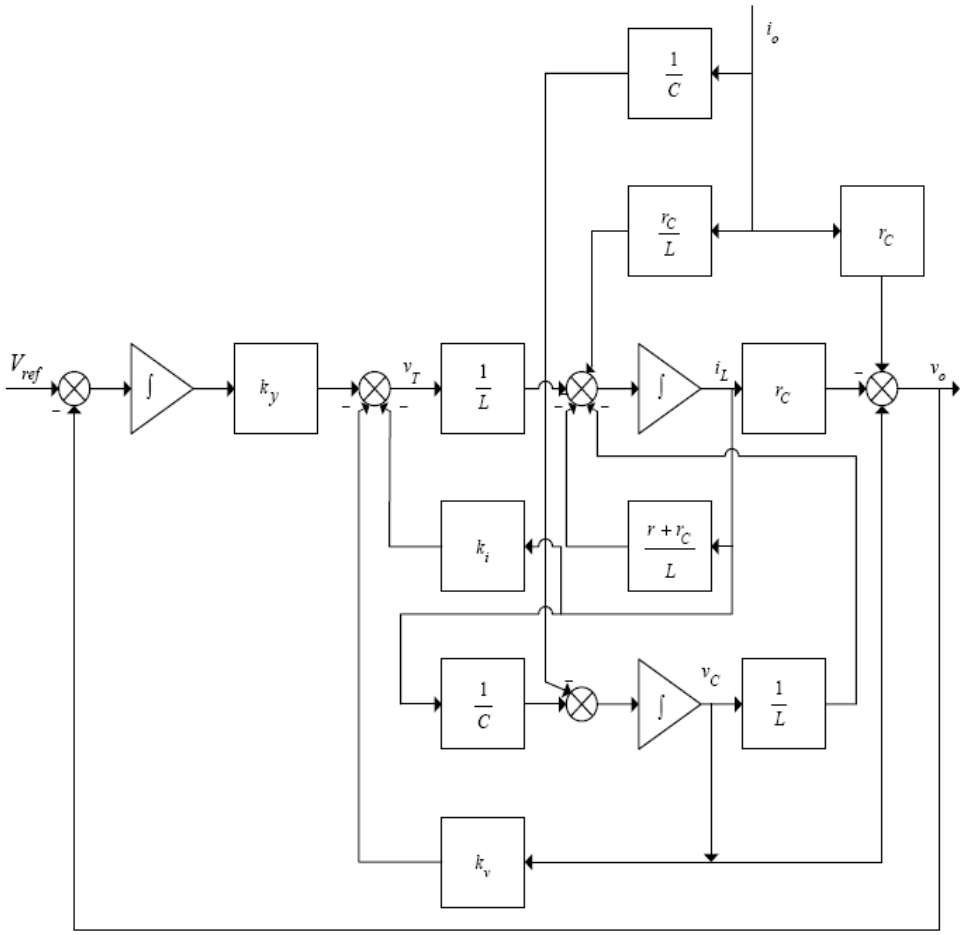

Figure 5.1: Steady-State Error Model of PWM push-pull dc-dc converters

steady-state error analysis of the PWM push-pull dc-dc converter is given in Fig. 5.1.

We now proceed with the dc-dc converter by the formalization of converter using the basic building blocks described in Chapter 3. The formal definition of the PWM push-pull dc-dc converter shown in Fig. 5.1 is given below:

Definition 5.1.1: *PWM push pull dc-dc converter*

$\forall$ `L C r rc Ky Kv Ki s.`

```
dc_dc_converter L C r rc Ky Kv Ki s =
  series [ Cx(&1)/s ; Ky ;
  feedback ( series [Cx(&1)/L; feedback (Cx(&1)/s)
  (--sum_junction (pickoff (Cx(&1))
  [(r+rc)/L; series [Cx(&1)/s; Cx(&1)/C; Cx(&1)/L]]))])
  (--sum_junction (pickoff (Cx(&1))
  [Ki; series [Kv; Cx(&1)/s; Cx(&1)/C]]));
  sum_junction [rc; series [Cx(&1)/s; Cx(&1)/C]]]
```

Where the `--` symbol represents the minus operator in HOL-Light. The variables L, C, r and rc donate inductor, capacitor, the equivalent resistance in series with an inductor and the equivalent resistance in series with a capacitor, respectively. While kv, ki and ky are the voltage, current and feedback loop gains, respectively. None of these variables can be zero for the correct operation of the dc-dc converter.

In order to verify the correctness of our formalized definition of the system, a subgoal of the equivalence theorem is given below, where the structure of the PWN push pull dc-dc converter is simplified to obtain its corresponding characteristic equation given in [10]:

Theorem 5.1.1: *dc-dc converter Transfer function simplification*

```
⊢ ∀ L C s r rc Ky Kv Ki.
  ¬( C * L * r * rc * Ki * Kv * Ky = Cx (&0)) ∧
  ¬(s = Cx (&0)) ∧
  ¬(s pow 2 *C *L + s*C*(r + rc) +Cx(&1) = Cx(&0)) ∧
  norm (inv s * --((r+rc) * inv L + inv (s*C*L))) < &1 ∧
  norm ((s * C) / ( s pow 2 * C * L + s * C * (r + rc) +
           Cx(&1)) * --(Ki + Kv * inv (s *C))) < &1
```

```
⇒ dc_dc_converter L C r rc Ky K Ki s =
    (Ky * (s*C*rc +Cx(&1))) / (s pow 3 * C* L + s pow 2 *
      C * (r+rc+Ki) + s (Cx(&1)+Kv) )
```

Note that none of the physical values in the model can be zero and this is ensured by the first assumption. The next two assumptions are used to avoid singularities and the last two assumptions are required for solving the feedback paths.

Since the correctness of the formalization has been established, now we proceed with the steady state error analysis of the dc-dc converter as described in Chapter 4. As our system is of Type 1 system, therefore its steady-state-error for the unit step input should be zero[19]. We have also verified this result and the theorem is given below:

Theorem 5.1.2: *Steady-State Error for step input*

```
⊢ ∀ L C r rc Ky Kv Ki.
  ¬( C * L * r * rc * Ki * Kv * Ky = Cx (&0)) ∧
  ¬(Cx (&1) + Kv = --Ky) ∧
  ¬(Cx (&1) + Kv = Cx (&0)) ∧
  (?k.  &0 < k ∧
  (∀s.  norm s < k
  ⇒ norm ((Ky * (s * C * rc + Cx (&1))) /
  (s pow 1 * (C * L * s pow 2 + s * C * (r + rc + Ki) +
  Cx (&1) + Kv))) < &1))
  ⇒ steady_state_error (uf_error_model (general_tf
      (λs.Ky * (s*C*rc + Cx(&1))) (λs.C*L* s pow 2 +
        s*C*(r+rc+Ki) + Cx(&1) + Kv ) 1) 1) = Cx(&0)
```

The first assumption ensures that none of the component in the dc-dc

converter has a zero value and the next two assumptions are used to avoid singularities. The last assumption is for the feedback simplification.

The verification of the above theorem involves the equivalence theorem, as described in the previous chapter, along with the simplification theorem described in Chapter 3.

Type 1 systems have a finite steady state error for ramp input. This error has been described in [10]. We now verify the steady state error of dc-dc converter for ramp input:

Theorem 5.1.3: *Steady-State Error for ramp input*

```
⊢ ∀ L C r rc Ky Kv Ki.
   ¬( C * L * r * rc * Ki * Kv * Ky = Cx (&0)) ∧
   ¬(Cx (&1) + Kv = --Ky) ∧
   ¬(Cx (&1) + Kv = Cx (&0)) ∧
   (?k.  &0 < k ∧
   (∀s.  norm s < k
   ⇒ norm ((Ky * (s * C * rc + Cx (&1))) /
   (s pow 1 * (C * L * s pow 2 + s * C * (r + rc + Ki) +
   Cx (&1) + Kv))) < &1))
   ⇒ steady_state_error (uf_error_model (general_tf
      (λs.  Ky * (s*C*rc + Cx(&1))) (λs.  C*L* s pow 2 +
       s*C*(r+rc+Ki) + Cx(&1) + Kv ) 1) 2) = (Cx(&1) + Kv) / Ky
```

The reasoning process was very similar to the one used for Theorem 6 since the same values for the functions Y and Z are used in these theorems. Further details about its verification can be found in our proof script[2].

5.2 Solar Tracking Control System

Solar tracking systems are used for the optimum generation of electricity from solar energy. These systems are usually deployed in aerospace applications where solar power is the only source for power generation. The block diagram representation of a traditional control system used in a solar tracking system is shown in Fig 5.2 [21], where `k` is gain adjustment factor. This system can be formalized, using our definitions, as follows:

Definition 5.2.1: *Solar Tracking Control System*

```
∀ s K. stcs s K = feedback (series [(s + inv (Cx (&100))) / s;
        feedback (series[(s + inv(Cx(&100))) / s; K; Cx(&1)/s])
         (--Cx (&1)); Cx (&1) / s])(--Cx (&1))
```

Here k is the gain of the system. *inv* is the inverse. Now, in order to verify the correctness of the formal model of solar tracking control system, we have verified the equivalence theorem. The theorem is given below:

Theorem 5.2.1: *Solar Tracking transfer function simplification*

```
⊢ ∀ s K.
  ¬(s pow 2 + K * (s + inv (Cx (&100))) = Cx (&0)) ∧
  ¬(s = Cx (&0)) ∧
  norm ((K * (s+inv(Cx(&100))) / s pow 2) * --Cx (&1)) < &1 ∧
  norm (((K * (s + inv (Cx (&100))) pow 2 * inv (s pow 2) *
  inv (s pow 2)) * inv (Cx (&1) + (s + inv (Cx (&100))) *
  inv (s pow 2) * K)) * --Cx (&1)) < &1
  ⇒ stcs s K =
    (K * (s + inv (Cx (&100))) pow 2) /
     (s pow 2 * (s pow 2 + K * (s + inv (Cx (&100)))) +
      K * (s + inv (Cx (&100))) pow 2)
```

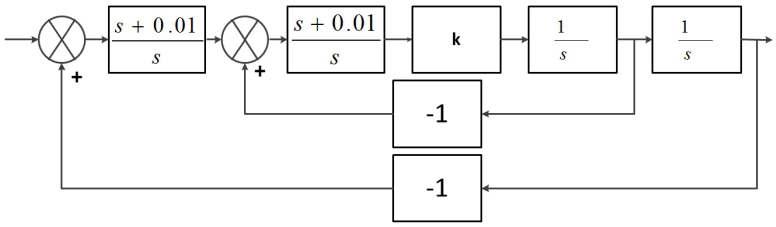

Figure 5.2: Solar Tracking Control System

Note that the first two assumptions are there to avoid singularity. The next two assumptions are required for the feedback goal simplification. Once the correctness is established, now we shall move to the steady state error analysis of the system. Since, it is a type 2 system so the system will have zero steady state error for unit step and ramp inputs.

Theorem 5.2.2: *Steady-State Error for unit-step input*

```
⊢ ∀ K.
  ¬(K = Cx (&0)) ∧
  (?k.  &0 < k ∧
  (∀s.  norm s < k
  ⇒ norm ((K * (s + inv (Cx (&100))) pow 2) /
    (s pow 2 * (s pow 2 + K * (s + inv (Cx (&100)))))) < &1))
  ⇒ steady_state_error (uf_error_model (general_tf
     (λs.  K * (s + inv (Cx (&100))) pow 2)
    (λs.  s pow 2 + K * (s + inv (Cx (&100)))) 2) 1) = Cx(&0)
```

The first assumption states that the gain K is not zero. While the second assumption is required for the feedback simplification. Now we verify the steady state error for ramp input which will be zero sice it is a *Type 2* system.

Theorem 5.2.3: *Steady-State Error for ramp input*

```
⊢ ∀ K.
   ¬(k = Cx (&0)) ∧
   (?k.  &0 < k ∧
   (∀s.  norm s < k
   ⇒ norm ((K * (s + inv (Cx (&100))) pow 2) /
     (s pow 2 * (s pow 2 + K * (s + inv (Cx (&100)))))) < &1))
   ⇒ steady_state_error (uf_error_model (general_tf
      (λs.  K * (s + inv (Cx (&100))) pow 2)
     (λs.  s pow 2 + K * (s + inv (Cx (&100)))) 2) 2) = Cx(&0)
```

The reasoning for the ramp input theorem is similar to that of unit-step input. The system will exhibit a finite steady state error for parabola input. This is given as follows:

Theorem 5.2.4: *Steady-State Error for parabola input*

```
⊢ ∀ K.
   ¬(k = Cx (&0)) ∧
   (?k.  &0 < k ∧
   (∀s.  norm s < k
   ⇒ norm ((K * (s + inv (Cx (&100))) pow 2) /
     (s pow 2 * (s pow 2 + K * (s + inv (Cx (&100)))))) < &1))
   ⇒ steady_state_error (uf_error_model (general_tf
      (λs.  K * (s + inv (Cx (&100))) pow 2)
        (λs.  s pow 2 + K * (s + inv (Cx (&100)))) 2) 3)
             = Cx(&1) / Cx(&100)
```

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis presents a novel application of theorem proving in the domain of control system analysis. We have developed a higher order logic based framework to accurately model and analyze control systems. The correctness of the framework is ensured by the soundness of the theorem prover which relies on primitive axioms and theorems to be satisfied for any new theorem to be accepted as correct. The approach is of vital importance since the correctness of design of control systems is of paramount importance due to their use in safety critical applications such as aerospace and robotic medical surgeries.

The contribution of the thesis can be categorized into two main parts. Firstly, we have identified the least number of basic control blocks which is required for representation of a wide range of control systems and their formalizations is also presented. Secondly, it provides the formalization of steady state error analysis of unity-feedback system which is of quite significance in control system design within the sound core of HOL-Light theorem

prover. The formal nature of the proposed methodology ensures error free analysis which is of vital significance in safety critical applications of control systems. To the best of our knowledge, this approach is novel and has not been used to reason for the correctness of control systems.

6.2 Future Work

The proposed methodology have opened new portals to various interesting research arenas. The new work can be build upon the reported results to further strengthen the formal analysis of control systems. Some interesting extensions are mentioned below:

- Enhancing the presented library of formally verified properties and adding new helping theorems. An automated prove tactic should be formalized which can perform the simplifications automatically. This will help in reducing human effort required.

- Formalization of Laplace Theory is also required if one wants to formal reasoning of control systems in true spirit. Infact, pliminary work have been reported in [30]. Both the theories should be linked to model control systems.

- Formalization of stability criterion should be done as it is among the most widely used criterion to judge the quality of a control system design.

Bibliography

[1] P. Clayton A. Cavalcanti and C. OHalloran. From Control Law Diagrams to Ada via Circus. *Formal Aspects of Computing*, 23(4):465–512, 2011.

[2] M. Ahmad. Formal Verification of Steady State Errors in Unity-Feedback Control Systems. http://save.seecs.nust.edu.pk/students/ahmad/fvsse.html, 2014.

[3] B. Akbarpour and L. C. Paulson. Metitarski: An Automatic Prover for the Elementary Functions. In *AISC/MKM/Calculemus*, pages 217–231, 2008.

[4] R. Alur. Formal Verification of Hybrid Systems. In *Embedded Software*, pages 273–278, 2011.

[5] R. Arthan, P. Caseley, C. O'Halloran, and A. Smith. ClawZ: Control Laws in Z. In *Formal Engineering Methods*, pages 169–176, 2000.

[6] R.J. Boulton, H. Gottliebsen, R. Hardy, T. Kelsey, and U. Martin. Design Verification for Control Engineering. In *Integrated Formal Methods*, volume 2999 of *LNCS*, pages 21–35, 2004.

[7] R.J. Boulton, R. Hardy, and U. Martin. A Hoare Logic for Single-Input Single-Output Continuous-Time Control Systems. In *Workshop*

on Hybrid Systems, Computation and Control, volume 2623 of *LNCS*, pages 113–125, 2003.

[8] J. S. Dong C. Chen and J. Sun. A Formal framework for Modeling and Validating Simulink diagrams. *Formal Aspects of Computing*, 21(5):451–483, 2009.

[9] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[10] D. Czarkowski, L. R. Pujara, and M. K. Kazimierczuk. Robust Stability of State-Feedback Control of PWM DC-DC push-pull Converter. *IEEE Transaction on Industrial Electronics*, 42(1):108–111, 1995.

[11] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.

[12] A. Gupta. Formal Hardware Verification Methods: A Survey. *Formal Methods in System Design*.

[13] J. Harrison. A hol theory of Euclidean Space. In *Theorem Proving in Higher Order Logics*, volume 3603 of *LNCS*, pages 114–129, 2005.

[14] J. Harrison. Formalizing Basic Complex Analysis. *Studies in Logic, Grammar and Rhetoric*, 10:151–165, 2007.

[15] J. Harrison. A List of Theorem Provers. http://www.cl.cam.ac.uk/users/jrh/ar.html, 2011.

[16] J. Harrison. The HOL Light Theory of Euclidean Space. *Journal of Automated Reasoning*, 50(2):173–190, 2013.

[17] O. Hasan and M. Ahmad. Formal analysis of steady state errors in feedback control systems using hol-light. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, pages 1423–1426, 2013.

[18] H. Herencia-Zapana, R. Jobredeaux, S. Owre, P. Garoche, E. Feron, G. Perez, and P. Ascariz. PVS Linear Algebra Libraries for Verification of Control Software Algorithms in C/ACSL. In *NASA Formal Methods*, volume 7226 of *LNCS*, pages 147–161. Springer, 2012.

[19] Y. Hote. A New Approach to Time Domain Analysis of Perturbed PWM push-pull DC-DC Converter. *Journal of Control Theory and Applications*, 10(4):465–469, 2012.

[20] E. M. Clarke Jr., O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[21] R . R. Kumar, P. A. Cooper, and T. W. Lim. Sensitivity of Space Station Alpha Joint Robust Controller to Structural Modal Parameter Variations. *Journal of Guidance, Control, and Dynamics*, 15(6):1427–1433, 1992.

[22] M. D. Lutovac and D. V. Toic. Symbolic Analysis and Design of Control Systems using Mathematica. *International Journal of Control*, 79(11):1368–1381, 2006.

[23] R. Milner M. J. C. Gordon and C. P. Wadsworth. Edinburgh LCF: A Mechanised Logic of Computation. In *Lecture Notes in Computer Science*, volume 78. Springer Verlag, 1979.

[24] B. Mahony. The DOVE approach to the Design of Complex Dynamic Processes. In *Workshop on Formalising Continuous Mathematics*, pages 167–187. NASA conference publication, 2002.

[25] MathWorks Simulink. www.mathworks.com/products/simulink, 2012.

[26] N.S. Nise. *Control System Engineering*. Wiley and Sons, 2003.

[27] K. Ogata. *Modern Control Engineering*. Prentice-Hall, 1997.

[28] L. C. Paulson. *ML for the Working Programmer*. Cambridge University Press, 1996.

[29] L. Pike. Pervasive Formal Verification in Control System. In *Formal Methods in Computer-Aided Design*. Panel Discussion, 2011.

[30] S. H. Taqdees and O. Hasan. Formalization of Laplace Transform Using the Multivariable Calculus Theory of HOL-Light. In *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 8312 of *LNCS*, pages 744–758. Springer, 2013.

[31] A. Tiwari and G. Khanna. Series of Abstractions for Hybrid Automata. In *Workshop on Hybrid Systems: Computation and Control*, volume 2289 of *LNCS*, pages 465–478, 2002.