Ain Shams University
Faculty of Engineering
Computer and Systems Engineering Department

Property Verification Patterns for Automated Production Systems

A Thesis

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctorate of Philosophy in Electrical Engineering (Computer and Systems Engineering)

Submitted by

Nahla Ahmed Mohamed El-Araby

Supervised by

Prof. Dr. Ayman Mohamed Hassan Wahba

Dr. Mohamed Taher

Cairo 2013

Statement

This dissertation is submitted to Ain Shams University for the degree of Philosophy Doctor in Electrical Engineering (Computer and Systems Engineering).

The work is included in this thesis was carried out by the author at the Computer and Systems Engineering department, Faculty of Engineering, Ain Shams University.

No part of this thesis has been submitted for a degree or qualification at other university or institution.

Nahla Ahmed Mohamed El-Araby
Computer and Systems Engineering Department
Faculty of Engineering
Ain Shams University
Cairo, Egypt
2013

Signature

Date / /

Acknowledgment

First and foremost, my utmost gratitude to Allah for helping me through all the hard times, and leading me on the way to this success.

My Supervisor **Prof. Dr. Ayman Wahba**, whose sincerity and encouragement I will never forget. Without his guidance and persistent help this thesis would not have been possible.

I was honored to have Prof. Dr. Reda Ammar and Prof. Dr. Ashraf Salem as my examining committee. I would like to thank them deeply for their valuable help and great support.

Thank you my dear father, you are my great role model Prof. Dr. Ahmed El-Araby. Thank you my dearest mother you are my great companion through all the hard times and the real owner of all success in my life. Thank you my sisters, I am really blessed with the best family ever.

I would like to thank all my professors, colleagues, and friends who gave me great support through the tough and hard academic research journey.

Also I have to thank Dr. Mostafa Alnagar who was a great inspiration in my life; he is not only one of the revolution youth who gave me and all the Egyptians the light of dignity and liberty, but also he encouraged me being a part of the great dream for our country, and moreover he was helping and supporting me during my research.

Nahla Ahmed Mohamed El-Araby
Computer and Systems Engineering Department
Faculty of Engineering
Ain Shams University
Cairo, Egypt
2013

ABSTRACT

Property Verification Patterns for Automated Production Systems

The most common procedure to ensure the reliability of a design is simulation. Unfortunately simulation cannot fully inspect all the execution states of the system. The significant increase in the complexity and size of digital systems together with the nature of real time systems boosted up the need for a different approach for the validation of the behavior of a system in the early design stages. Formal verification is an approach to validate a system by formally reasoning the system behavior. In formal verification the system implementation is checked against the requirements or the properties to be satisfied. B method is one of the common paradigms used in formal verification. B method offers strong verification domain as it based on a mathematical and logical approach. The proof obligations (properties that must be satisfied) are automatically generated from the model, also the available tools provides both automatic and interactive proofs.

VHDL is a mature implementation domain where many synthesis and simulation tools are available.

The work in this thesis presents a technique to convert B machines into the corresponding VHDL implementation in order to implement a correct by construction system, benefit from the advantages of both strong domains, and maintain the properties of the verified model.

Challenges in going from B domain to VHDL domain:

- Not all constructs of B are available in VHDL.
- Completely different Semantics.
- Different approaches of handling design modules.
- Variables and signals are represented in completely different ways.

We reached for a method to cross the gap and converted the B machines into VHDL implementations. The unavailable B constructs are described by VHDL code constructing the required part of the model. The developed tool starts by an initial step to handle the connected modules in a way that can be converted into VHDL code. The B machine is investigated to find the way each variable is used and indicate how it can be handled in the generated VHDL code. Five popular models were used as workbenches where we applied the developed technique. Simulation at some critical points was used to ensure that the generated VHDL satisfy the verified properties in the original B machine.

Publications

1- FORMAL VERIFICATION OF REAL TIME DISTRIBUTED SYSTEMS USING B METHOD, International Journal of Engineering Science and Technology (IJEST), ISSN : 0975-5462 Vol. 3 No. 4 April 2011

2- Implementation of Formally Verified Real Time Distributed Systems: Simplified flight control system, The 2011 International Conference on Computer Engineering & Systems *(ICCES'2011),* 978-1-4577-0126-9/11©2011 IEEE

TABLE OF CONTENTS

Title **Page No.**

List of Tables

List of Figures

List of Abbreviations

BDD Binary Decision Diagram

CTL Computation Tree Logic

DES Discrete Event Systems

GSL Generalized Substitutions Language

FSM Finite State Machine

HDL Hardware Description Language

LTL Linear Temporal Logic

MDA Model Driven Architecture

OBDD Ordered Binary Decision Diagram

PLC Programmable Logic Controller

PO Proof Obligation

RLL Relay Ladder Logic

RTL Register Transfer Language

UML Unified Modeling Language

V&V Verification and Validation

Chapter 1
Introduction

Throughout the previous years, the complexity and size of digital systems has increased dramatically, as a result design flow phases changed a lot. Simulation used to be the most common procedure to assure the correctness of a system under design, but it cannot exhaustively examine all the execution scenarios of the system. A different approach to validate a system by formally reasoning the system behavior is Formal verification, where the system implementation is checked against the requirements or the properties to be satisfied. The most common paradigms are based on *theorem proving*, *model checking* and *language containment*.

People and products safety are directly affected by the reliability of automated systems. Safety aspects should be considered from early design stages up to operational stages and this needs a very accurate design approach [1]. This becomes more sophisticated in real time systems as real-time systems differ from untimed systems in that the correct behavior relies on computation results plus the time on which they were produced. The resulting state-space explosion makes it infeasible to run a satisfactory number of simulation traces to achieve enough coverage of the state spaces and enough confidence in the design correctness within a project schedule. Even if it were feasible to have extensive coverage of the system, missing only single untested sequence of events may cause the system failure.

1.1 Formal Verification

Formal verification means to thoroughly investigate the correctness of system designs expressed as mathematical models. Formal verification is a useful and powerful technique for guaranteeing the reliability of systems in design stages [2]. In recent years, several approaches for applying formal verification techniques on automation systems dependability have been proposed. These range from formal verification by theorem proving [3] to formal verification by model-checking [4], [5], [6], [7]. Many achievements in the formal verification of real-time systems are presented in [8], [9], [10] and [11] .

The verification problems of timed systems are usually exponentially more complex than their untimed counterparts. Most major projects are spending over 50% of their development costs in verification and integration, so using formal verification can substantially reduce the explosive growth of verification and integration costs and improve the quality of system designs in industry. On the one hand, using formal verification for complex real-time systems will likely enhance the intelligence and performance of simulation and testing. For example, coverage metrics can be more precisely mapped to the functions to be verified. Also, formal verification can be used to carefully check the components and the interfaces and progressively could be

accepted as standard methods in the automation of industrial quality control. It is claimed that this approach has already had a remarkable effect on the SLAM project of Microsoft, which plans to include model-checking capability in its Windows driver development kit (DDK) [12].

Formal specification is defined by the IEEE standard as a specification written in a formal method. Formal methods are particular type of mathematically-based procedures for the specification, development and verification of systems. Performing appropriate mathematical analysis that contributes to the reliability and robustness of a design is the motivation for using formal methods for design. Systems can be formally described at different levels of abstraction. The formal description can be used to guide further development activities; moreover, it can be used to verify that the requirements for the system being developed have been entirely and precisely specified. A variety of formal methods and notations available are available, like Z notation, VDM and B-Method.

Verification plays a vital role in the design cycle of any safety critical system. The development of any system is not complete without careful testing and verification that the implementation satisfies the system requirements. In the past, verification was an informal process performed by the designer. But as the complexity of systems increased, it became necessary to consider the verification as a separate step in the overall development cycle. Verification techniques can be either based on simulation or based on formal methods. Simulation is based on a model that describes the possible behavior of the system design at hand. This model is executable in some sense, such that a simulator can determine the system's behavior on the basis of some scenarios. Formal Verification is defined as "establishing properties of hardware or software designs using logic, rather than (just) testing or informal arguments. This involves formal specification of the requirement, formal modeling of the implementation, and precise rules of inference to prove that the implementation satisfies the specification" [13]. Three categories can be used to classify the Formal Verification methods - *equivalence checking*, *model checking* and *theorem proving*.

1.2 B-Method

The B method is a model-oriented formal method for engineering software systems developed by Abrial [16]. It is a comprehensive formal method that covers the entire development cycle. The method is based on the mathematical principles of set theory and predicate calculus while its semantics is given using a variant of Dijkstra's weakest precondition calculus [17]. A hierarchy of components that are described using the Abstract Machine Notation (AMN) constitutes a B specification. Each component in a specification represents a state machine where a set of variables defines its state and a set of operations query and modify that state. Generalized substitutions describe state transitions. Constraints on the operation and variable types are described as invariants of a machine. In B models *Abstract Machines* are the top-level components describing state machines in an abstract way. *Refinements* are another type of components that exist in a B specification; they represent enriched versions of either an *Abstract*

Machine or another *Refinement*. The last type of components is *Implementations* where ultimate refinement of an *Abstract Machine* is described; both data and operations need to be implementable in a high-level programming language.

Syntax and type checking can be performed on a system modeled in B. Also a B model consistency can be verified to check the preservation of invariants and the correctness of all refinement steps.

1.3 Problem statement

The common approach for system design is to start the design cycle by implementing the basic requirements then starting to test and correct errors in the developed design. This "construct-by-correction" approach leads to a long and more expensive design cycle.

Since the B method offers a strong framework for developing and verifying models at different abstraction levels, the verified B models can be used to develop "correct-by-construction" designs but the problem is when converting the model into an implementation, some verified properties may be lost. VHDL is a mature implementation domain where many synthesis and simulation tools are available.

Automatic conversion of the verified B models into implementation avoids losing any of the verified properties. In addition to providing an implementation directly mapped from the verified model which achieve "correct-by-construction" design approach.

Also the developed implementation will take the advantages of both the verification B domain and the strong well matured VHDL implementation domain.

The work in this thesis presents a technique to convert B machines into the corresponding VHDL implementation in order to build a "correct- by-construction" system, maintain the properties of the verified B model, and benefit from the advantages of both strong domains.

1.4 Related work

Integration of B with UML is achieved by Butler [14], B affords a semantic framework to UML components and allows analysis of UML models. It is interesting to study the impact of the B refinement into UML models.

The work in [15] develops a method for translating a system described by B method into formal specification language of PVS. In this method, a machine in B method is represented to be a theory in PVS, while an invariant to be a type in PVS. Some properties described by B method

are converted into conjectures of PVS language, and then these conjectures can be proved effectively by PVS prover.

A tool denoted BHDL was presented in [16] introducing a project converting VHDL code to B models to ensure the circuits under design are correct by construction.

Generating executable code from B models is an interesting field and a lightweight framework for executable code generation from B formal specification is presented in [17]. The translation framework including the translation strategy, process and implementation is introduced.

Model Driven Architecture (MDA) design approach in [18] proposes to separate design into two stages: implementation independent stage then an implementation dependent one. This improves the reusability, the reliability, the standability, the maintainability, etc. MDA can be augmented using the B method as a formal refinement approach. The approach allows to gradually refining the development from the abstract specification to the executing implementation through many controlled steps. Each refinement step is mathematically represented and is proven to be correct, accordingly the implementation is proven to satisfy the specification; furthermore the presented approach enables proving the coherence between components in low levels even if they are branched in different technologies during the development.

In [1] the authors have looked at the issue of supporting the expression of property specifications. Meaningful properties can be hard to write and hard to get right, this is even more the case when considering the behavior of complex automated systems whose requirements are difficult to describe. The approach was to provide the designer with patterns that can be instantiated to produce the properties of interest. A finite state system can be represented by a labeled state transition graph, where labels of a state are values of atomic propositions in that state. Model checking is the modeling approach used which consists of traversing the graph of the transition system and verifying that it satisfies the formula representing the property. By studying and identifying the properties used for the verification of DES “Discrete Event System” automation, it becomes possible to systematize the writing of such properties in an automatic way. The paper discusses the property specification patterns proposed by [Dwyer et al., 1998]. A pattern’s goal is to capture proven solutions to known problems, and demonstrate how they can be used in practice to solve the same or similar problems in new situations. They have briefly introduced a tool that supports the use of specification patterns for the generation of properties for verification. The tool helps to generate properties from the combination of models and patterns. A list of property patterns and help for instantiating those patterns with attributes and actions in the model are provided. Instantiation of patterns to produce a property specification can be done in two modes: *manual* where the values of the parameters are indicated manually by the user, and *automatic* where the system reads the attributes from a model and the user can use