Polynomial Abstraction for Verification of Sequentially Implemented Combinational Circuits*

Tarvo Raudvere, Ashish Kumar Singh, Ingo Sander and Axel Jantsch
Royal Institute of Technology
Stockholm, Sweden
{tarvo,ashish,ingo,axel}@imit.kth.se

Abstract

Todays integrated circuits with increasing complexity cause the well known state space explosion problem in verification tools. In order to handle this problem a much simpler abstract model of the design has to be created for verification. We introduce the polynomial abstraction technique, which efficiently simplifies the verification task of sequential design blocks whose functionality can be expressed as a polynomial. Through our technique, the domains of possible values of data input signals can be reduced. This is done in such a way that the abstract model is still valid for model checking of the design functionality in terms of the system's control and data properties. We incorporate polynomial abstraction into the ForSyDe methodology, for the verification of clock domain design refinements.

1. Introduction

Due to the increasing capacity of integrated circuits the complexity of the embedded system design process is continuously growing. In order to ensure a guaranteed functionality of these systems the design process has to be based on a formal basis and allow for the application of formal methods, in particular formal verification, since simulation alone is not sufficient.

The main formal verification techniques are theorem proving and model checking. Though theorem proving allows infinite data types and an infinite state space, its practical use is restricted, since it demands good knowledge and ingenuity of the designer to construct a mathematical proof. The use of model checking is more practical, since the design is represented as a finite state machine and the properties to verify are specified in temporal logic. Unfortunately model checking can only be applied to systems with a finite state space and suffers from the state space explosion problem. These problems are addressed by abstraction techniques.

This paper introduces the *polynomial abstraction* technique that is a modification of *spatial abstraction* [1]. Spatial abstraction separates the control part from the data path of a design and based on the assumption about the correctness of the data path, it allows to reduce the data path so that the behaviors of the control part are preserved. This reduces the state space and allows to verify the correctness of the control part and data path predicates.

In comparison to spatial abstraction, our method based on polynomial abstraction allows to verify the system functionality as a mix of data and control properties, i.e. it can be verified that a system output is a function of the system inputs.

The polynomial abstraction will be incorporated into ForSyDe [2], which is a design methodology for embedded systems using formal design transformations. The abstraction is applied for verification of design transformations.

2. The ForSyDe Methodology

The ForSyDe design process starts with the development of a synchronous, formal, abstract and functional *specification model* that can be executed using the functional language Haskell. This model is then refined by a stepwise application of well defined design transformations into an *implementation model*. As the implementation model is a refined version of the specification model, the same validation and verification methods can be applied to both models. The implementation model is partitioned into hardware and software blocks, which are mapped on architectural components. The verification of design transformations in ForSyDe is introduced in [3].

The specification model is based on a synchronous computational model and uses ideal data types such as real numbers and infinite buffers. It abstracts from implementation

* This research was supported by the Swedish Foundation for Strategic Research within the INTELECT program.

details, such as low-level communication mechanisms and enables the designer to focus on the functional behavior of the system rather than structure and architecture.

The implementation model is the result of the refinement process. In contrast to the specification model, which is a network of concurrent synchronous processes, it may also include synchronous sub-domains with a different signal rate. In order to connect clock domains with different clock rates, domain interfaces are settled between domains.

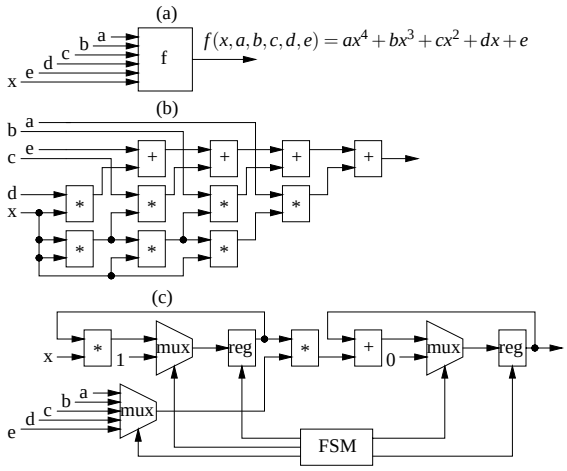

Figure 1. (a) Specification, (b) Intermediate Model and (c) Implementation

An example of clock domain refinement is shown in Figure 1, which transforms a combinational function in the specification model into an implementation as a circuit with data path and control part. The FSM in the control part is configured so that in the first cycle the registers are initialized with constant values 0 and 1 from the multiplexers. In the following cycles the sum of the values x^n multiplied with coefficients *a,b,c,d* and *e* is calculated. The implementation runs on a higher clock rate compared with the intermediate model and thus does not change the functionality of the model.

3. Polynomial Abstraction

Polynomial abstraction targets the verification of systems where a sequential design implementation is functionally equivalent to a combinational circuit. For instance it is obligatory to verify that the sequential circuit in Figure 1.(c) calculates the degree four polynomial.

One possibility to verify this kind of designs is to execute the implementation with symbolic variables on the system inputs and formally find the output function presented as a polynomial in symbolic variables. Through equivalence checking of the corresponding specification and implementation polynomials we can decide over the correctness of the refinement.

Unfortunately the implementation polynomial does not contain all the information about the control part of the circuit. We can only decide over the correctness of the first calculation, since the polynomial does not express how the controller behaves after emitting the first result.

We address the latter problem by introducing *polynomial abstraction*. The technique is based on *the fundamental theorem of algebra* [4]. Using this theorem we can prove a lemma, which states the following: two degree k univariable polynomials are equivalent if they evaluate to pairwise the same values for $k+1$ different input assignments. We can extend the lemma for multi-variable polynomials and prove the corresponding theorem. This theorem allows us to decide over equivalence of two multi-variable polynomials through assigning only a small set of values to every input signal instead of all the values of the input domains.

Instead of the exact output polynomial of the implementation we only need to know the degrees of the input variables in the polynomial. According to the degrees we restrict the domains of input variables and use model checking to verify the implementation against specification. Considering the fundamental theorem of algebra, the verification result is valid for any data domain.

The polynomial abstraction is applicable for verification at a high abstraction level, where a system is described in the sense of ideal data types abstracting from lower level details, such as saturating or floating point arithmetic, and bitwidths. We target designs, which contain integers, booleans and scalar variables, the arithmetic operations + , - , *, /, and logic operations. The design may contain feedback loops with delay and finite state machines. The proposed technique is prospective for the verification of DSP applications and VLSI implementations of cryptological algorithms.

References

[1] Viresh Paruthi, Nazanin Mansouri, and Ranga Vemuri, "Automatic data path abstraction for verification of large scale designs," in *ICCD '98 Topic : Verification and Test*, 1998.

[2] Ingo Sander, Axel Jantsch, and Zhonghai Lu, "Development and application of design transformations in ForSyDe," in *Design, Automation and Test in Europe Conference (DATE 2003)*, Munich, Germany, March 2003.

[3] Tarvo Raudvere, Ingo Sander, Ashish Kumar Singh, and Axel Jantsch, "Verification of design decisions in ForSyDe," in *CODES+ISSS*, Newport Beach, California, USA, October 2003.

[4] J. E. Eaton, "The fundamental theorem of algebra," *American Mathematical Monthly*, vol. 67, no. 6, pp. 578–579, 1960.