DRS - Derivational Reasoning System

Bhaskar Bose

Derivation Systems, Inc.
5963 La Place Court, Suite 208
Carlsbad, California 92008
Tel: (619) 431-1400
bose@dvsi.com

May 11, 1995

Abstract

The high reliability requirements for airborne systems requires fault-tolerant architectures to address failures in the presence of physical faults, and the elimination of design flaws during the specification and validation phase of the design cycle. Although much progress has been made in developing methods to address physical faults, design flaws remain a serious problem. Formal methods provides a mathematical basis for removing design flaws from digital systems.

DRS (Derivational Reasoning System) is a formal design tool based on advanced research in mathematical modeling and formal synthesis. The system implements a basic design algebra for synthesizing digital circuit descriptions from high level functional specifications. DRS incorporates an executable specification language, a set of correctness preserving transformations, verification interface, and a logic synthesis interface, making it a powerful tool for realizing hardware from abstract specifications. DRS integrates recent advances in transformational reasoning, automated theorem proving and high-level CAD synthesis systems in order to provide enhanced reliability in designs with reduced time and cost.

DRS - Derivational Reasoning System

Bhaskar Bose

Derivation Systems, Inc.
5963 La Place Court, Suite 208
Carlsbad, CA 92008
(619) 431-1400
bose@dvsi.com

Overview

- The Problem Domain
- Derivation and Verification
- DRS - Derivational Reasoning System
  - An Example
- Conclusions

The Problem Domain

- Computers are being used with increasing frequency where the correct implementation is critical.
  - complex systems
  - reduced time to market
  - safety-critical applications
- Establish a rigorous framework for reasoning about complex designs and guaranteeing integrity in the design process.

Derivation Systems, Inc.

- Corporate Mission
  - Advanced research for the development of computer engineering tools for the mathematical modeling and formal synthesis of digital hardware systems.

Derivation and Verification: "Alternate" Modes of Formal Reasoning

- Verification
  - Constructing a post factum "proof of correctness" for a design.
- Derivation
  - Deriving a "correct by construction" design.

Integrating Derivation and Verification

- Alternate modes of formal reasoning must be integrated if formal methods are to support the natural analytical and generative reasoning that takes place in engineering practice.

Background

- "Synthesis of Digital Designs from Recursion Equations" (Johnson '84)
- DDD - Digital Design Derivation System (Bose, et.al. '87-94)

Derivation Examples

- DDD-FM9001 [Bose'94]
- Scheme Machine [Burger et.al '94]
- Fault-tolerant Clock Synchronization Circuit [Miner '94]
- FM8501, FM8502 [Bose '90]
- Stop-and-Copy Garbage Collector [Boyer '89]
- SECD Machine [Wehrmeister '89]

NASA SBIR Contract

- NASA Langley Research Center (LaRC)
- Develop a formal design tool based on advanced research in derivational reasoning, automated theorem proving, high-level synthesis, and logic synthesis.

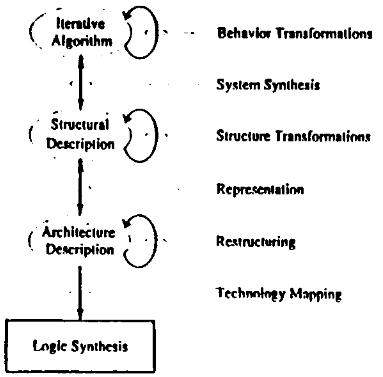
DRS - Derivational Reasoning System

- DRS is a first-order **transformation system** which implements a basic design algebra for deriving digital circuit descriptions from high level functional specifications.
- Specifications are purely functional <u>verifiable</u> and <u>executable</u> recursive expressions.

DRS Design Methodology

- Top-down design methodology
- Deriving a "correct by construction" design
- <u>Correctness preserving</u> transformations
- <u>Ad hoc</u> transformations
- Algebraic mechanisms for isolating verification problems to small building blocks.

DRS Design Flow



Iterative Algorithm

Structural Description

Architecture Description

Logic Synthesis

Behavior Transformations

System Synthesis

Structure Transformations

Representation

Restructuring

Technology Mapping

An Example: Fibonacci

Consider the recursive Fibonacci specification:
(The "->" denotes the conditional operator.)

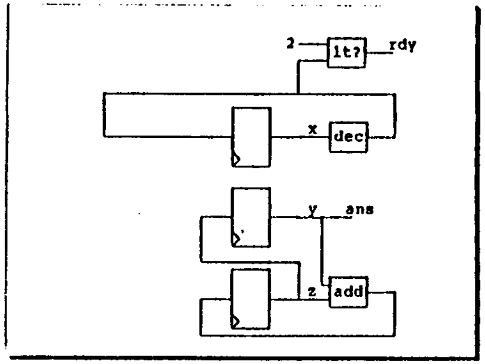
g(x, y, z) = lt?(x, 2) -> y, g(dec(x), z, add(y, z))

where

fib(x) = g(x, 1, 1)

From this recursive definition, we can derive an initial architecture:

An Initial Architecture



State Introduction

The first phase in the derivation is to apply a series of transformations at the behavioral level. In this phase the dec and add operations are serialized so that they may be combined into a single logic unit later in the derivation.

The first step is to introduce a function h, such that the call to g will fold within the definition of h. Introducing a definition h yields

g(x, y, z) = lt?(x, 2) -> y, g(dec(x), z, add(y, z))
h(x, y, z) = g(x, z, add(y, z))

Folding

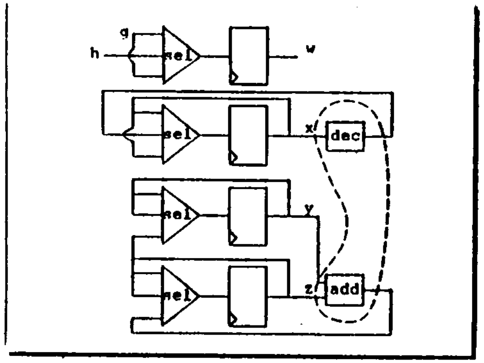
The next step is to fold the call, g(dec(x), z, add(y, z)), in g into the definition of h resulting in

g(x, y, z) = lt?(x, 2) -> y, h(dec(x), y, z)
h(x, y, z) = g(x, z, add(y, z))

This has the effect of splitting the dec and add operations between two separate definitions.

From this transformed specification, we derive the following structural description:

The Initial Structural Specification

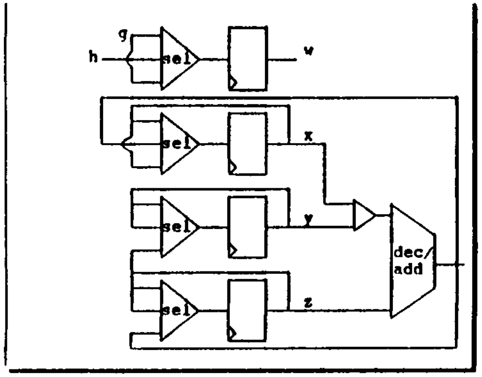


The Initial Structural Specification (contd.)

In this phase of the derivation, transformations are applied to refine the structural specification to an architecture. The goal is to encapsulate the dec and add operations (highlighted by the dotted circle) into a single abstract component.

An algebraic transformation, called factorization, encapsulates the dec and add operations into a single component and synthesizes a behavioral description denoting the abstraction.

Factoring: dec and add



Further Stages of the Derivation

- Implement the abstract component with either
  - continued derivation of the abstract component
  - mapping to a verified library component
  - use of verification to replace with a particular implementation.
- Control and architecture are directly synthesized in hardware.

Conclusion

- Derivation methodology, along with the incorporation of verification and logic synthesis, provides a powerful tool to support the natural analytical and generative reasoning that takes place in engineering practice.
- The DRS System, is a reflection of this ideal.

Future Directions

- Continued development of DRS through Phase II SBIR Contract.
- Mechanical proofs of transformation rules.
- Integration with high-level theorem provers.
- Develop tacticals and analysis tools.
- Application of DRS to a practical design problem.