DRS - Derivational Reasoning System

Bhaskar Bose

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

Copyright (c) 1995 Derivation Systems, Inc.

Overview

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

Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.

Background

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

Copyright (c) 1995 Derivation Systems, Inc.

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]

Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.

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 verifiable and executable recursive expressions.

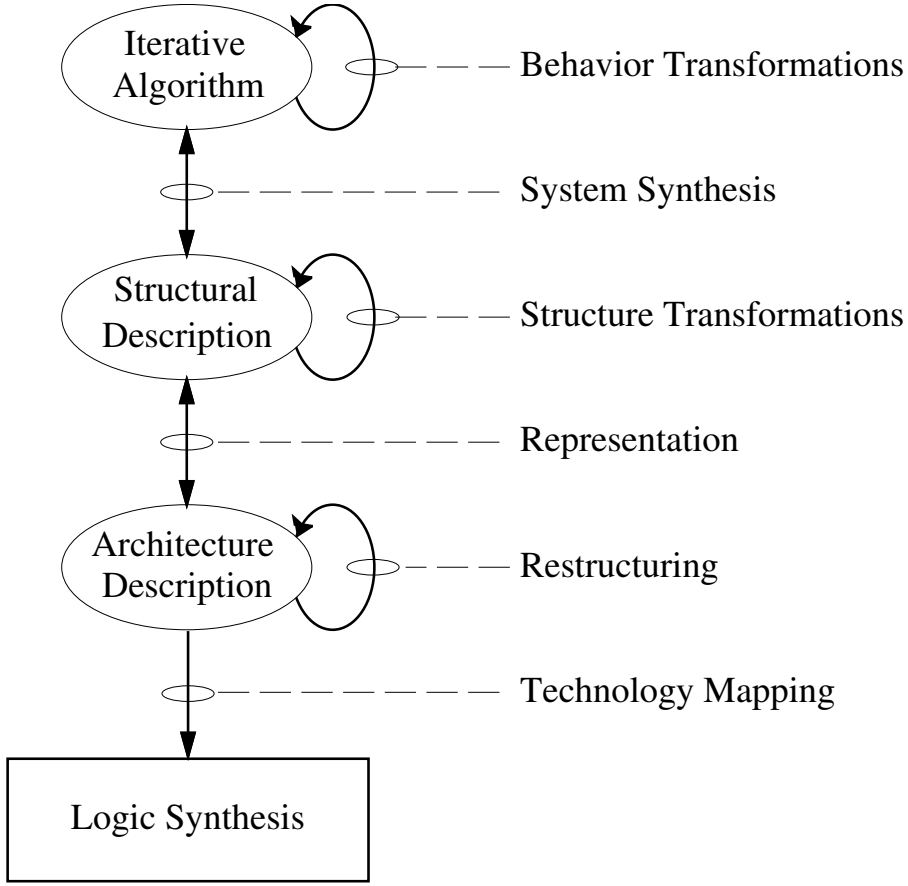
Copyright (c) 1995 Derivation Systems, Inc.

DRS Design Methodology

- Top-down design methodology
- Deriving a “correct by construction” design
- Correctness preserving transformations
- Ad hoc transformations
- Algebraic mechanisms for isolating verification problems to small building blocks.

Copyright (c) 1995 Derivation Systems, Inc.

DRS Design Flow



Copyright (c) 1995 Derivation Systems, Inc.

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))

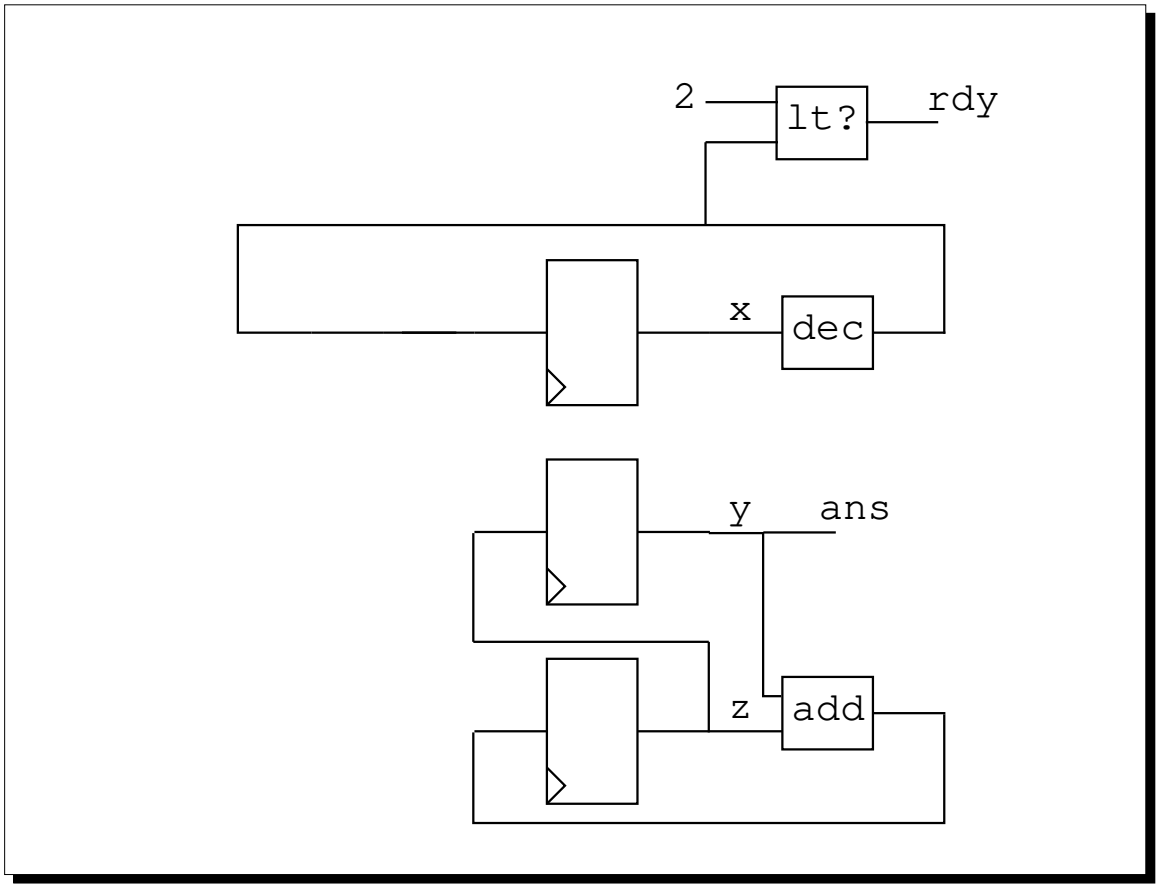
where

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

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

Copyright (c) 1995 Derivation Systems, Inc.

An Initial Architecture



Copyright (c) 1995 Derivation Systems, Inc.

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))
```

Copyright (c) 1995 Derivation Systems, Inc.

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

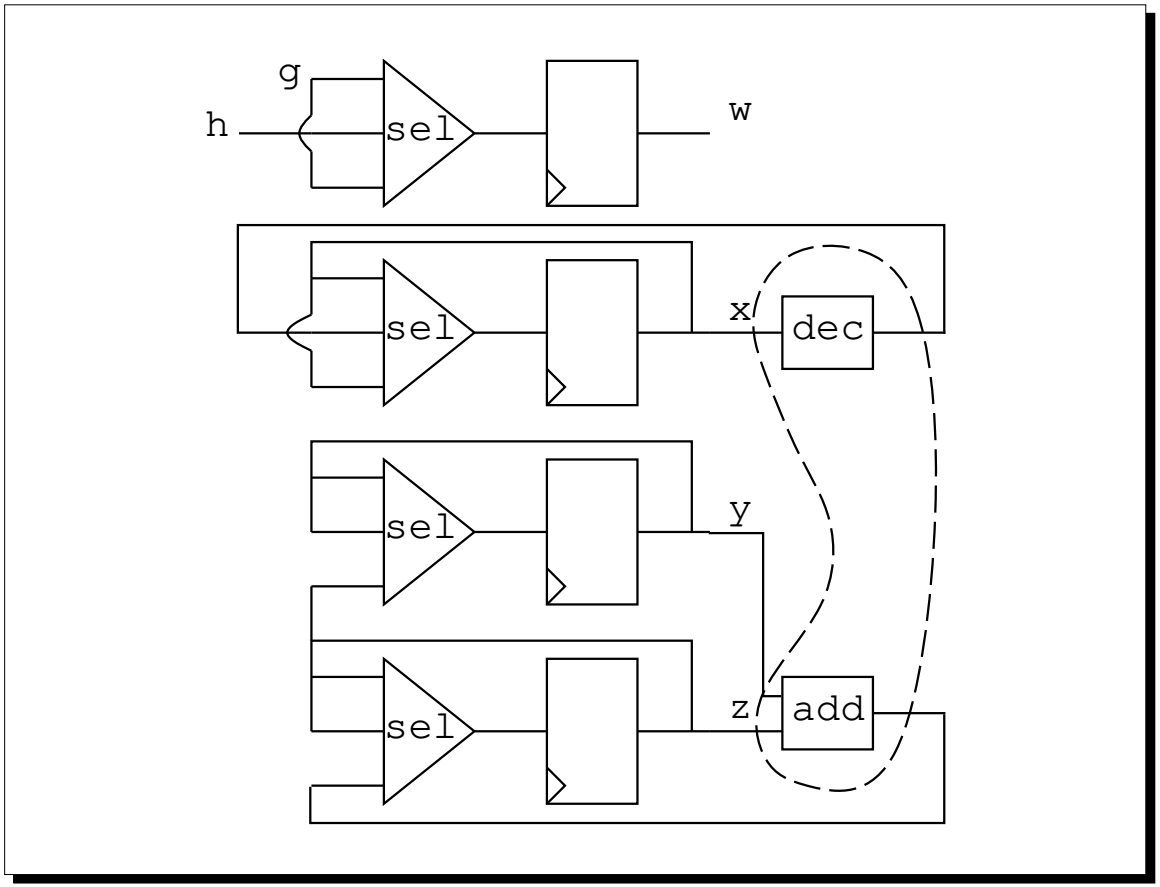
```
g(x, y, z) = lt?(x, 2) -> y, h(dec(x), y, z)
h(x, y, z) = g(x, z, add(y, z))
```

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:

Copyright (c) 1995 Derivation Systems, Inc.

The Initial Structural Specification



Copyright (c) 1995 Derivation Systems, Inc.

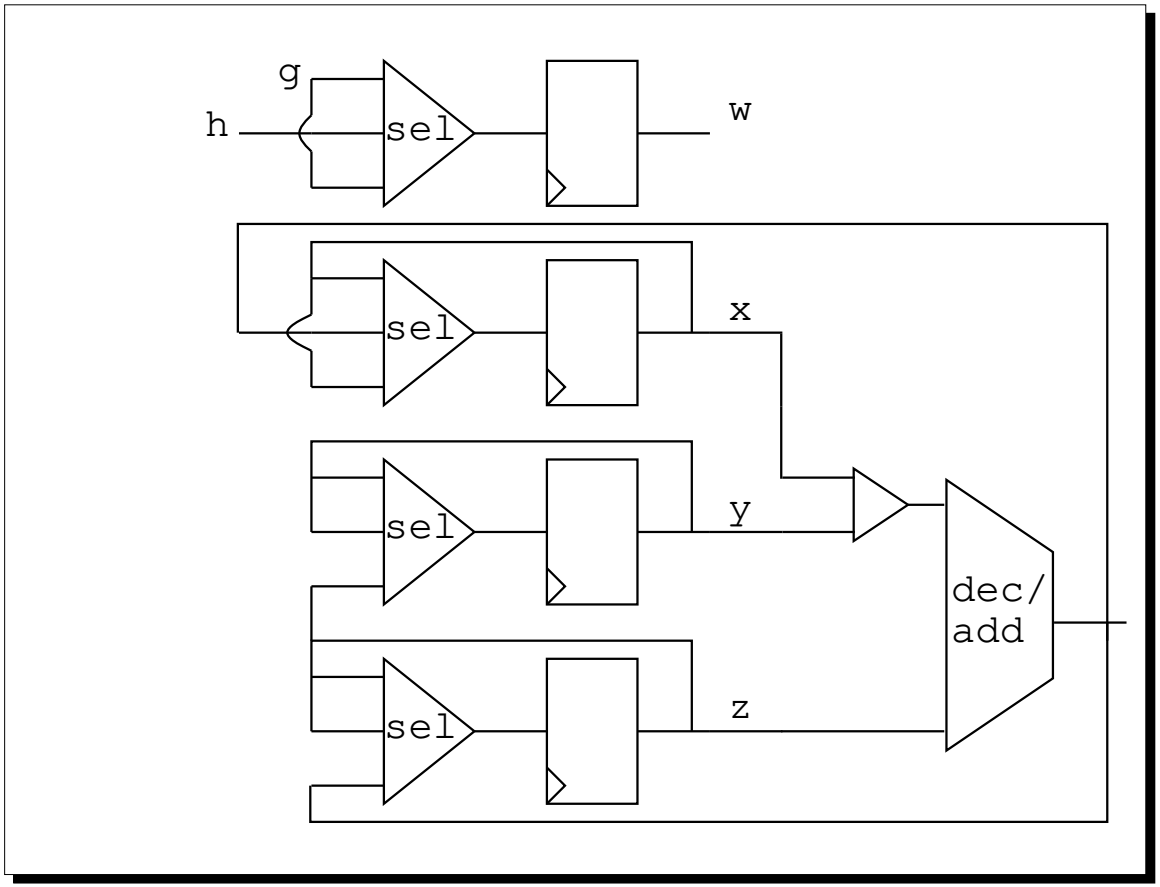
The Initial Structural Specification (contd.)

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.

Copyright (c) 1995 Derivation Systems, Inc.

Factoring: dec and add



Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.

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.

Copyright (c) 1995 Derivation Systems, Inc.