Final Report for NSF Grant MIP89-21842:

Algebra for Digital Design Derivation

1 June 1990 – 30 November 1992

Steven D. Johnson, Principal Investigator
(sjohnson@cs.indiana.edu)
Hardware Methods Laboratory
Computer Science Department
Indiana University
Bloomington, IN 47405-4101 USA

9 July 1993

Contents

II. Summary of completed project

Design derivation refers to a formalism of design in which algebraic manipulations are used to construct correct implementations. This project expanded on results from NSF Grant MIP87-07067 to refine and automate the derivation formalism as it applies to digital-system design; and this line of research continues under grant MIP92-08745. In theoretical research, several results were published extending the conventional approach of algebraic data types to address hardware structure and timing issues. The first full release of the *DDD* transformation system was completed. Using DDD a number of demonstration designs were realized in various VLSI technologies. The most significant of these demonstrations was the derviation of an implementation of Hunt's FM9001 microprocessor. Previous versions of FM9001 had been verified using a theorem prover. This exercise illustrated how a combination of reasoning tools can be used together to make design verification more practicable. Finally, there was significant progress in extending derivation to system levels of design, although the first articles on this topic did not appear until after the grant period.

III. Technical information

III.1. List of publications

The following articles and reports acknowledge grant MIP89-21842 and represent work done, in whole or in part, during the funded grant period.

[BJ91] Bhaskar Bose and Steven D. Johnson. DDD-FM9001: Derivation of a verified microprocessor. an exercise in integrating verification with formal derivation. In *Proceedings of IFIP Conference on Correct Hardware Design and Verification Methods (CHARME'93)*. Springer, 1993.

[BJP93] Bhaskar Bose, Steven D. Johnson, and Shyam Pullela. Integrating boolean verification with formal derivation. In D. Agnew, L. Claesen, and R. Camposano, editors, *Proceedings of IFIP Conference on Hardware Description Languages and their Applications (CHDL'93)*, pages 127–134. Elsevier, April 1993.

[Bose91] Bhaskar Bose. DDD - A Transformation system for Digital Design Derivation. (User's manual) Technical Report 331, Indiana University, Computer Science Department, May 1991.

[BTJ93] Bhaskar Bose, M. Esen Tuna, and Steven D. Johnson. System factorization in codesign: A case study of the use of formal techniques to achieve hardware-software decomposition. In *Proceedings of the International Conference on Computer Design*. IEEE, October 1993. to appear.

[JBA93] Steven D. Johnson, Gerard Allwein, and K. Jon Barwise. Toward the rigorous use of diagrams in resoning about hardware. Indiana University Logic Group Preprint Series, Department of Philosophy, Bloomington, May 1993.

[JB91] Steven D. Johnson and Bhaskar Bose. A system for mechanized digital design derivation. In P. A. Subrahmanyam, editor, *Proceedings of ACM International Workshop on Formal Methods in VLSI Design*, January 1991.

[RBJ93] Kamlesh Rath, Bhaskar Bose, and Steven D. Johnson. Derivation of a DRAM memory interface by sequential decomposition. In *Proceedings of the International Conference on Computer Design*. IEEE, October 1993. to appear.

[RCW90] K. Rath, I. Celis, and R. M. Wehrmeister. RTBA: A generic bit-sliced bus architecture for datapath synthesis. Technical Report 321, Department of Computer Science, Indiana University, December 1990.

[RJ93] Kamlesh Rath and Steven D. Johnson. Toward a basis for protocol specification and process decomposition. In D. Agnew, L. Claesen, and R. Camposano, editors, *Proceedings of IFIP Conference on Hardware Description Languages and their Applications*

(CHDL'93), pages 157–174. Elsevier, April 1993. Also published as Technical Report No. 375, Dept. of Computer Science, Indiana University.

[RTJ93] Kamlesh Rath, M. Esen Tuna, and Steven D. Johnson. Behavior tables: a basis for system representation and transformational system synthesis. To appear, Proceedings of the International Conference on Computer Aided Design (ICCAD'93).

[Zhu92] Zheng Zhu. *Structured Hardware Design Transformations*. PhD thesis, Computer Science Department, Indiana University, USA, 1992.

[ZJ90] Zheng Zhu and Steven D. Johnson. An algebraic framework for data abstraction in hardware description. In Jones and Sheeran, editors, *Proceedings of The Oxford Workshop on Designing Correct Circuits (DCC'90)*. Springer, 1990.

[ZJ91] Zheng Zhu and Steven D. Johnson. An example of interactive hardware transformation. In P. A. Subrahmanyam, editor, *Proceedings of ACM International Workshop on Formal Methods in VLSI Design*, January 1991.

[ZJ93] Zheng Zhu and Steven D. Johnson. Automatic synthesis of sequential synchronizations. In D. Agnew, L. Claesen, and R. Camposano, editors, *Proceedings of IFIP Conference on Hardware Description Languages and their Applications (CHDL'93)*, pages 285–301. Elsevier, April 1993.

III.2 Software developed.

The DDD system is the principal vehicle for this research. It is a collection of constructions and transformations used interactively to translate high-level behavioral specifications to a form suitable for VLSI synthesis. DDD has been integrated with a number of CAD environments to produce working circuitry. A number of standard-cell and PLA layouts have been generated though the Berkeley Octtools package. However, since the primary purpose of most of our design work is research demonstration, field-programmable devices are the primary target technologies. Paths exist from DDD into Actel Corperation's FPGA devices and to various PLD configurations.

This grant period saw the first full release of the DDD system [Bose91]. It is available free of charge by contacting *bose@cs.indiana.edu* or *sjohnson@cs.indiana.edu*. DDD is implemented in Chez Scheme, a software product that must be obtained seperately. DDD is integrated with various CAD tools and environments, which also must be obtained separately.

Although there was minimal support from MIP89-21842, development continued on the *Daisy/DSI* programming system. Daisy is a concurrent, functional modelling language with stream-based I/O. These language features, together with an Xwindows front-end, make Daisy a good tool for high-level modeling of hardware systems.

III.2. Hardware developed

Systems and circuits developed in this research are demonstrations of methodology. They are of marginal value in themselves. However, complete information on all circuits, derivation scripts, intermediate sources for synthesis tools, layouts, and test software can be obtained by contacting *sjohnson@cs.indiana.edu*.

Numerous small design-derivation examples have been developed for expository purposes. One worth noting is the Winkel-Prosser blackjack dealer, which is explained in great detail in [JB91].

Two major design demonstrations done under this grant are described next. The vehicle for both of the implementation projects is the *Indiana University Logic Engine*, a general purpose digital prototyping environment. This environment includes a general-purpose PC host, used to generate vectors for functional testing and to integrate hardware development with our modeling environment. The Logic Engine (board and parts list) and all supporting documentation are available at cost from the Indiana University

Computer Science Department, Bloomington, Indiana.

The FM9001 derivation (Photograph 1)

The most ambitious demonstration of DDD in its relation to formal-methods research involved the FM9001 microprocessor, designed by Warren Hunt at Computational Logic, Inc., under contract by NSA, NASA, and DARPA. The FM9001 is the third in a series of microprocessor descriptions proved correct by Hunt in the Boyer-Moore theorem prover, *Nqthm*, and the to have been carried all the way to hardware. Hunt proved in *Nqthm* that the gate network used to generate an LSI Corp. gate-array realization was a correct implementation of an instruction-level model of the processor. By nearly all measures, this is the most advanced result in high-level hardware verification, worldwide.

Because the functional s-expressions of *Nqthm* are literally expressions DDD can handle, Hunt's designs are an ideal test-bed for experimentation in our research. We can directly compare derivation oriented and deduction oriented approaches to design verification. Furthermore, this kind of experimentation progresses toward the important goal of an integrated environment in which many reasoning systems are brought to bear on a single design problem.

Prior to this project, DDD had been applied to the first two in Hunt's FM series, but much of the algebra was done manually and the resulting gate-level implementations, though reasonable, were not actually realized in VLSI. For the FM9001, we were able to create a fully mechanized derivation targeted to FPGA technology.

Photograph 1 shows the test environment for the FM9001 derivation. At mid-right in the project area is Hunt's gate-array version of the FM9001. The derived FPGA version with external register file is below that. Both microprocessors are connected to a single 64-kilobyte memory through arbiter circuits. The processors can operate in tandem or in parallel and have been

Photograph 1. **Two verified versions of the FM9001.**

This project was done on the Indiana University Logic Engine prototyping environment. At mid-right of the project area is Hunt's FM9001 microprocessor. The larger FPGA chip beneath it is a version of FM9001 derived using DDD. Just below FM9001/DDD is its external register file (a 16×32-bit static RAM, which would not have fit on the FPGA device). To the right of each microprocessor is a PAL arbiter chip, which governs access to a common 64-kilobyte SRAM memory (upper left).
Both microprocessors are verified in a formal sense, one using inductive proof and the other using algebraic derivation, both starting with an identical behavioral specification. The devices have also been compared experimentally, and have exhibited no differences.

executing software for several months. We have observed no differences in behavior at the instruction level. The two processors differ in numerous ways at the micro-state level, but conformance at this level was not a goal of the derivation. Several more papers reporting this experimentation are in progress.

Scheme Machine derivations (Photograph 2)

Implementation of a programming language is a standard "completeness" exercise for the formal approach of this research. An ongoing demonstration exercise has been the derivation of a computer from the mathematical description of the programming language Scheme (a version of Lisp in which DDD is implemented). This work stems from research on compiler derivation in the early 1980s and contributes to a larger research endeavour at Indiana to develop programming methodology based on functional techniques.

The Scheme-machine derivation starts with a virtual-machine description of a published compiler derivation.[1] It is a full language implementation with the exception of floating-point arithmetic. While there remain significant gaps in the theoretical account of the whole derivation (i.e. from fully abstract semantics to hardware) we are quite close to having a working hardware implementation of a complete computer.

Photograph 2 shows the development environment for this project. The upper half of the circuit is a manually designed, high-performance, dual-ported heap memory, consisting of two 2-megabyte semi-spaces. Beneath the memory is a fully functional storage allocator, which includes a stop-and-copy garbage collector. The collector-allocator consists of three processes, realized in FPGA and PLD circuitry, all of which were fully derived in DDD. CPU specifications have been carried to the gate-network level, partitioned, and optimized for FPGA implementation; but not yet realized.

[1]Daniel P. Friedman, Mitchell Wand, and Christopher T. Haynes, *Essentials of Programming Languages,* MIT Press and McGraw Hill, 1992.

PHOTOGRAPH 2. **A derived heap manager.**

This Logic Engine design project is to derive a specialized computer for symbolic processing, based on the Scheme programming language. At the top is a manually designed high-performance (effectively, two memory operations in one write cycle) dual-ported memory, consisting of two 2-megabyte semispaces. The three FPGA devices across the middle implement a stop-and-copy garbage collector, which was fully derived using DDD. The fourth FPGA at the left is a "trailer process," which runs in parallel with the CPU to initialize new memory. The remaining space on the board is for the CPU—a virtual machine for compiled Scheme—and a primitive I/O subsystem. Gate level implementations of the CPU have been successfully derived using DDD. The CPU specification executes against both the software model of the heap from which the heap manager was derived and against the hardware itself.

These derivations have been performed seperately on the principal subsystems: the storage manager, the allocator, the CPU, and the I/O subsystem. However, we do not yet have the capability to extract these sybsystems from a more comprehensive specification. That is, DDD and its underlying theoretical basis are not developed enough to do "system design." We are approaching this capability, as detailed in recent articles about sequential decomposition.

Part of the Logic Engine development environment is an interface between the project board (shown in Photographs 1 and 2) and an implementation of Scheme on the PC host computer. Part of the research is to explore the migration of hardware specifications, as modeled in Scheme, into hardware, as derived using DDD. In the Scheme machine project for example, a single behavioral model of the CPU has been executed against several models of the heap, an abstract description using Scheme's own heap manager, several stages of the derived implementation, and the actual hardware. These explorations have opened new research investigations in modeling, specification, and hardware-software co-design.

III.4. Other sponsored activities

This grant made nominal contributions to departmental infrastructure as budgeted. Such contributions included clerical support, technical support, and materials and supplies.

Supported personnel

- *Steven D. Johnson*, Principal Investigator, five summer months.
- *Bhaskar Bose* (Research Assistant), one academic year. Nominated to Ph.D. candidacy, August 1990. NASA Fellowship recipient, September 1991 through September 1993. Topic: digital design derivation. Graduation expected in 1993.

- *Zheng Zhu* (Research Assistant), two academic years. PhD 1992, *Structured Hardware Design Transformations*, directed by Steven D. Johnson.

- *Kamlesh Rath* (Research Assistant), one academic year. Also supported by a departmental assistantship during the Fall of 1992. Nominated to PhD candidacy, July 1991. Topic: decomposing specifications into interacting sequential processes. Graduation expected in 1993.

The DDD project has other participants, funded from various sources. For the period of MIP89-21842, participants included *William Hunt* (Member of Technical Staff, Indiana University Computer Science Department), *M. Esen Tuna*, (on departmental assistantship from September 1991 to May 1993.) *Shyam Pullela* (on departmental assistantship September 1992 to January 1993), and *Kathryn Fisler*.

Events and activities

MIP89-21842 contributed $54 to help sponsor a visit to the department by Wayne Wolf, of Princeton University. It also contributed $200 to support the Indiana University Logic Group, a continuing interdisciplinary seminar in logic and its applications. Three presentations of design-derivation research were made to the Logic Group during the period of this grant.

Trips and research presentations

Individuals receiving travel support from MIP89-21842 are indicated by an asterisk.

1. The Oxford Workshop on *Designing Correct Circuits (DCC'90)*, Zheng Zhu*, "An Algebraic Framework for Data Abstraction in Hardware Description." Steven D. Johnson*.

2. IFIP WG10.2 meeting, Charlottesville, November 1990. Steven D. Johnson*

3. ACM/IEEE/IFIP Workshop of Formal Methods in VLSI Design, Miami, January 1991. Steven D. Johnson*, "DDD – A System for Mechanical Digital Design Derivation." Bhaskar Bose*, demonstration of the DDD system. Zheng Zhu, "An Example of Interactive Hardware Design."

4. Algebraic Methodology and Software Conference (AMAST'91), Iowa City, May 1991. Zheng Zhu*.

5. ICCD'91 and IFIP WG10.2 meeting, Boston, October 1991. Steven D. Johnson*.

6. 1991 Microelectronics System Education Conference, San Jose, July 1991. Steven D. Johnson.

7. University of Utah Computer Science Department, October 25, 1991. Steven D. Johnson, "Daisy/DSI: an environment for exploring near-functional parallelism" and "Deductive and Derivational reasoning in Digital Design."

8. TAU'92, Princeton, March 1992. Zheng Zhu*.

9. ICCD'92 Technical Program Committee, Boston, March 1992. Steven D. Johnson*.

10. NASA Formal Methods Group, Langley Research Center, February 1992. Steven D. Johnson, "Derivational Reasoning for Digital-system Verification" and panel member: "Issues in Hardware Verification."

11. University of Cincinnatti Department of Electrical Engineering. Steven D. Johnson, "Heterogeneous Reasoning in Digital System Design."

12. Second NASA Formal Methods Workshop, Langley Research Center, August 1992. Steven D. Johnson, "Derivational Techniques for Hardware Verification." Bhaskar Bose, "Demonstration of the DDD system and FM9001 derivation."

13. *IEEE International Conference on Computer Design,* Cambridge, October 1992. Steven D. Johnson, Session chairman: "Environments for High-Level CAD."