Red Bend Software, Inc. et al v. Google Doc. 9 Att. 4

UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS
EASTERN DIVISION

RED BEND LTD., and RED BEND SOFTWARE INC., Plaintiffs, v. GOOGLE INC., Defendant.	Civil Action No. 09-cv-11813-DPW **DECLARATION OF STEPHEN A. EDWARDS IN SUPPORT OF PLAINTIFFS' MOTION FOR A PRELIMINARY INJUNCTION ENJOINING GOOGLE'S INFRINGEMENT**

I, Stephen A. Edwards, declare as follows:

I. Introduction

1. Counsel for RED BEND LTD. and RED BEND SOFTWARE INC. ("Red Bend") have asked me to investigate whether Google Inc.'s ("Google") differential compression algorithm intended to make Google Chrome updates significantly smaller ("Courgette") infringes the claims of Red Bend's U.S. Patent No. 6,546,552 (the "'552 patent," attached as Exhibit 1) that are currently being asserted in this case. In particular, for purposes of Red Bend's motion for preliminary injunction, Red Bend's counsel has asked me to focus on the question of whether claims 8-10, 21-23, 42-44, and 55-57 ("the relevant claims") of the '552 patent are infringed by Google's Courgette and use of the published Courgette source code by Internet users/software developers. I understand that I may be called upon to testify about additional matters, including additional claims being asserted by Red Bend and other issues related to this action.

Dockets.Justia.com

2. I conclude that under what I consider to be the correct definitions of the terms in the claim construction chart (Exhibit A), Google's Courgette and users of the Courgette source code published by Google infringe these claims either literally or under the doctrine of equivalents. I explain the reasoning for my conclusions below and in the attached exhibits.

A. My Experience and Qualifications

3. I am an Associate Professor in the Computer Science Department of Columbia University in New York City. I research, teach, and practice in the field of software compilation. I know and understand the facts set forth in this declaration and am competent to testify about them.

4. I have substantial experience and expertise in the fields of programming languages, software development, and compliers. Attached as Exhibit B is a copy of my current *curriculum vitae*.

5. I received my PhD in Electrical Engineering from the University of California, Berkeley in 1997. I received my M.S. degree in Electrical Engineering in 1994 also from Berkeley. I received my B.S. in Electrical Engineering from the California Institute of Technology in 1992.

6. I am the sole named inventor of United States Patent "Method and Apparatus for Converting a Concurrent Control Flow Graph into a Sequential Control Flow Graph," Patent Number 7,100,164, issued on August 29, 2006.

B. Bases for My Conclusions

7. In reaching my conclusions, I considered (i) the patent-in-suit and its prosecution history; (ii) the announcement on Google's Chromium Blog (*See* http://blog.chromium.org/2009/07/smaller-is-faster-and-safer-too.html; Exhibit 2); (iii)

the Chromium Developer Documentation (*See* http://dev.chromium.org/developers/design-documents/software-updates-courgette; Exhibit 3); (iv) the published Courgette source code files available at http://src.chromium.org/viewvc/chrome/trunk/src/courgette/; and (v) my experience, including developing and updating software.

8. I also address what I believe are the correct interpretations of the terms used in the asserted claims of the '552 patent. In light of that claim construction and other evidence, I believe all of the limitations of at least the relevant claims of the '552 patent are met by Google's Courgette and the published source code available to all Internet users. In the event that the Court later adopts a different construction of these claims, I may revise my opinions based on any new construction. I also would like to respond to any rebuttal analysis presented by Google regarding claim construction, infringement and/or invalidity.

II. Basic Operation of the Preferred Embodiment of the Red Bend Patent

9. Red Bend's patent describes a method for generating a compact difference between an old executable program and a new executable program. Most importantly, each program is assumed to include entries that contain references to other parts of the program. *See* Abstract of the '552 Patent. The invention described and claimed in Red Bend's patent is concerned with writing and updating software, a field in which I have experience and expertise.

10. A computer program in a remote location can be updated by transmitting a full copy of the new version to that location. However, as explained in the '552 patent, transmission of the entire program is inefficient when the new program file is large. The

amount of data that needs to be transmitted can be reduced by comparing the old and new programs and only sending the difference. However, using file difference utilities in existence before the invention disclosed in the '552 patent typically resulted in more difference data than necessary, even if modifications to the program were small, for reasons I explain below.

11. In general, software developers write programs using high-level languages. Such high-level programming languages spare the developer from the low-level details of the computer on which the program will ultimately execute. Compared to the so-called machine language understood by a typical computer, high-level languages are more English-like, facilitating ease of development. A high-level language program is commonly referred to as "source code."

12. For a program to run on a computer, its source code must be converted, "compiled," into an executable program (or "object code"). Such object code consists primarily of machine code instructions and references to other parts of the program. A reference could be the address of another instruction in a "jump" instruction that directs the processor to begin running instructions at another location.

13. Minor source code changes, such as the addition of a single line, can translate into the addition of many machine code instructions in the object code. The addition of instructions in the object code shifts all the code after that point, meaning that references to this code would have to change (because they would be at a new address).

14. For example, assume that twenty references in a program refer to address 150. If even a single machine code instruction is inserted prior to 150, it would then reside at address 151 and all twenty references to "150" would have to be changed to

"151." Comparing the old and new versions of the program using old comparison algorithms would find all of these differences, rather than just the addition of the single line.

15. The inventor of the '552 patent recognized that "the relatively large size of the difference result stems from the alterations of reference in reference entries as a result of other newly inserted entries (and/or entries that were deleted)." ('552 patent, 3:32-35). That is, he recognized that changed references typically accounted for a large fraction of identified differences, which produced needlessly large diffs.

16. The methods and systems disclosed in the '552 patent overcome this problem, making difference results much smaller. Prior to comparing an old and new program to generate a diff, the methods of the '552 patent create a modified old and a modified new program which in one embodiment involves replacing each reference with a label mark: a means of identifying a particular location other than its physical address. For instance, labels may be sequential (0, 1, etc.). The modified old and new programs are then compared, producing a difference that no longer depends on references, because references in both programs were rendered invariant, making this result file many times smaller than one generated by techniques available prior to the methods disclosed in the '552 patent.

17. In one embodiment described in the patent, the desired invariant references are calculated by generating modified old and new programs where address references in entries are replaced by label marks and a final difference result between the modified old and modified new programs, which now contain corresponding labels for corresponding reference entries, is generated. Since corresponding reference entries are

assigned corresponding labels, changes in the reference (or target) of a reference entry due solely to insertions and deletions will not be included in the difference result. (*See* Exhibit 1, '552 Patent beginning at 10:47).

III. Claim Construction

18. Counsel for Red Bend has asked me to offer my views on how one of ordinary skill in the art in 1998 would have understood certain terms and phrases recited in the relevant claims. My views about the meaning of these terms and phrases are set forth in the claim construction tables (Exhibit A)

19. In the event that Google disputes the meaning of claim terms not covered in the attached tables, I will review such terms and offer my views on them and/or expand on the terms described in the attached tables.

20. A person of ordinary skill in the art of this invention is someone with a bachelors of science degree in the field of computer science (or an equivalent) who has around two years of software development experience with some understanding of how source code is converted to machine instructions to be executed on a computer.

IV. Infringement Analysis

21. My understanding is that one can infringe a patent's claims either literally or by equivalents. Literal infringement requires that the accused system or method contain all the limitations of the claim exactly or inherently. My understanding is that a claim is infringed under the doctrine of equivalents if the accused system or method contains only insubstantial changes from the claims' limitations and/or performs substantially the same function, in substantially the same way, to achieve substantially the same result.

22. Here, Google's use of Courgette infringes the '552 patent literally and, if not literally, by equivalents. The process by which Google's Courgette generates patches meets every limitation of at least the relevant claims. Courgette disassembles the old version of a program (such as Chrome), creating a modified old data table (as described in the claims) in which references are replaced with labels. Courgette also disassembles and adjusts the new version of the program, creating a modified new data table where references of corresponding entries are replaced with labels that correspond to those in the modified old table. Courgette then generates a difference result using the modified old data table and the modified new data table.

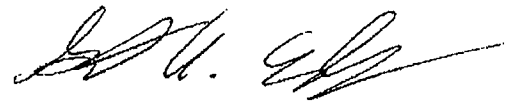
23. The Courgette source code published by Google at http://src.chromium.org/viewvc/chrome/trunk/src/courgette/ provides instructions to Internet users with some programming knowledge and a C++ compiler on how to practice the invention. The instructions combined with the source code would enable persons with less skill than one of ordinary skill in the art to practice the invention. Those Internet users, such as software developers, would simply download the published files and compile them into an executable. The documentation and comments in the files describe how to make use of the tool. The majority of the Courgette source code would have no substantial use other than in infringing the '552 patent.

24. The published source code currently supports generation of difference results for Microsoft Windows executables. However, the code is written such that it is easily adaptable to processing executable files for other platforms, such as those found in mobile devices.

25. Additional details of the analysis that support my conclusion may be found in the infringement charts attached as Exhibit C. Note that while the infringement charts refer specifically to claims 42-44, my analysis also applies to the other claims I considered, which differ only in a few words. Specifically, in this context, I consider "executable program" and "data table" equivalent, and that the difference between a "system for generating a ... difference" and "a method for generating a .. difference" is insubstantial: it is understood that the method is performed on a processor when Google generates a Courgette patch.

26. I did not undertake a validity analysis of the '552 patent for the purposes of this declaration; I understand that the patent enjoys a presumption of validity. If asked, I will respond to any of Google's arguments concerning the validity of the '552 patent if such an analysis is presented.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct and that this declaration was executed on November 17, 2009 at Newport Beach, California

By: 

Stephen A. Edwards

EXHIBIT A

Exhibit A to Declaration of Prof. Stephen A. Edwards

Red Bend v. Google U.S. Patent No. 6,546,552 Claim Construction		
Claims that term appears in	**Term**	**Red Bend's Proposed Constructions**
42, 55	Data Table	A table of entries, where an entry is an addressable unit within the data table. An executable program is one example of a data table. *See, e.g.,* '552 patent 2:33-36; 2:61-63
	Address	A number which is uniquely assigned to a single entry by which that entry is accessed. *See, e.g.,* '552 patent 2:37-38.
8, 21, 42, 55	Compact difference result	A difference result of a smaller size as compared to a conventional difference result obtained by using techniques in existence prior to the invention of the patent-in-suit. *See, e.g.,* '552 patent 3:30-46; 14:5-14
42, 55	Old data table	A data table (or portion of a data table) that is to be updated. *See, e.g.,* '552 patent 2:51-54; 2:61-63
8, 21, 42, 55	Reference	Part of the data appearing in an entry in the data table which is used to refer to some other entry from the same data table. A reference can be either an address or a number used to compute an address. *See, e.g.,* '552 patent 2:42-45
8, 21, 42, 55	Reference entry	An addressable unit containing data that includes a reference. *See, e.g.,* '552 patent 2:35-36; 2:46-47
42, 55	Modified old data table	A data table generated using the old data table. *See, e.g.,* '552 patent claims 42, 55.
42, 55	Modified new data table	A data table generated using the new data table. *See, e.g.,* '552 patent claims

Exhibit A to Declaration of Prof. Stephen A. Edwards

Red Bend v. Google U.S. Patent No. 6,546,552 Claim Construction		
Claims that term appears in	**Term**	**Red Bend's Proposed Constructions**
		42, 55.
8, 21, 42, 55	Invariant	Unvarying, invariable, constant. *Random House Webster's Unabridged Dictionary* 1003 (2nd ed. 1998).
8, 21, 42, 55	Invariant references	References that are the same. *See, e.g.,* '552 patent 10:10-15

EXHIBIT B

Stephen A. Edwards
Department of Computer Science, Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027–7003
(212) 939–7019
sedwards@cs.columbia.edu
http://www.cs.columbia.edu/~sedwards/

Research Interests

Languages and compilers for Embedded Systems. Computer-Aided Digital Design.

Education

University of California, Berkeley Ph.D in Electrical Engineering — 1994–1997
Ph.D, Electrical Engineering. Edward A. Lee, advisor
Thesis: *The Specification and Execution of Heterogeneous Synchronous Reactive Systems.*

University of California, Berkeley M.S. in Electrical Engineering. — 1992–1994
A. Richard Newton, advisor
Thesis: *An Esterel Compiler for a Synchronous/Reactive Development System.*

California Institute of Technology B.S. with Honors, Electrical Engineering — 1988–1992

Experience

Columbia University, New York — 2006–present
Associate professor, Department of Computer Science
Tenure awared June, 2008

Columbia University, New York — 2001–2006
Assistant professor, Department of Computer Science

Synopsys, Mountain View, California — 1998–2001
Senior R&D Engineer

Simplex Solutions, Sunnyvale, California — 1997–1998
Senior Member of Technical Staff.

Interval Research Corporation, Palo Alto, California — 1993
Researcher.

Vitesse Semiconductor, Camarillo, California — 1991
VLSI Design Engineer.

Microsoft, Redmond, Washington — 1990
Hardware Design Engineer.

Honors and Awards

Best paper award — March 2006
Design Automation and Test in Europe
Munich, Germany (given to 2 of 800+ submissions)

Senior Member — 2006
IEEE (Institute of Electrical and Electronics Engineers)
This is the main professional organization for Electrical Engineers

National Science Foundation Faculty Early Career Development ("CAREER") Award — 2002
"Designing Embedded Systems with Domain-Specific Languages"
I won this award the first time I applied.

NSF Graduate Research Fellowship — 1994–1996
Three years tuition & stipend, awarded annually to about 800 of 5000 applicants.

California Fellowship in Microelectronics — 1992–1993
One year tuition plus stipend.

Caltech Merit Award — 1990–1991, 1991–1992
One year full tuition, awarded annually to about 45 of 800 undergraduates.

Research Support

All were new grants.

NSF CSR-EHS, $200k ($50k × 4 years) — 2007–2010
PRET: Precision Timed Architectures

NSF CSR-EHS, $240k — 2006–2008
SHIM: Developing Embedded Systems with Deterministic Concurrency
I was sole PI.

Gift from Altera, $20k — 2006
Hardware Software Co-Synthesis from SHIM,

Joint Semiconductor Research Corporation/Microelectronic Design Center, $300k — 2003–2006
High-Level Synthesis from the Synchronous Language Esterel
I was sole PI.

New York State, NYSTAR program, matching funds, $11k — 2002

Hardware grant from Intel, $13k — 2002

Gift from Intel, $25k — 2002
High-level Synthesis from the Synchronous Language Esterel

NSF Faculty Early Career Development (CAREER) Award, $300k — 2002–2007
Designing Embedded Systems with Domain-Specific Languages
I was sole PI.

Released Software

The Columbia Esterel Compiler (2003–)

- Only open-source compiler for the Esterel language.
- Generates the most efficient C code for Esterel of any known compiler, including the commercial implementation.
- Used at Freescale semiconductor, NASA, University of Kiel, and University of Auckland. Cited in about 15 papers.
- The CEC-Users mailing list has 32 subscribers from companies including Motorola, TI, Esterel Technologies, Philips, Intel, IBM, and Xilinx
- `http://www1.cs.columbia.edu/~sedwards/cec/`

The EstBench Esterel benchmark suite (2004)

- Only public benchmark suite for the Esterel language
- Cited in about ten papers
- `http://www1.cs.columbia.edu/~sedwards/software.html`

The Ext C-code documentation extraction system (1997)

- Used in the widely-distributed VIS and CUDD software packages (at University of Colorado, Boulder). Cited in a few papers.
- `http://www1.cs.columbia.edu/~sedwards/ext/`

Publications

Patent

US Patent 7,100,164. "Method & Apparatus for Converting a Concurrent Control Flow Graph into a Sequential Control Flow Graph." Filed January 6th, 2000, issued September 29th, 2006.

Books

[1] Dumitru Potop-Butucaru, Stephen A. Edwards, and Gérard Berry. *Compiling Esterel.* Springer, 2007.

[2] Stephen A. Edwards. *Languages for Digital Embedded Systems.* Kluwer, Boston, Massachusetts, September 2000.

Chapters in Books

[3] Stephen A. Edwards. Design and verification languages. In Luciano Lavagno, Grant Martin, and Lou Scheffer, editors, *Electronic Design Automation for Integrated Circuits Handbook.* CRC Press, Boca Raton, Florida, 2006.

[4] Stephen A. Edwards. Languages for embedded systems. In Richard Zurawski, editor, *The Embedded Systems Handbook*, pages 7–1–7–19. CRC Press, Boca Raton, Florida, 2005.

[5] Stephen A. Edwards. Languages for embedded systems. In Richard Zurawski, editor, *The Industrial Information Technology Handbook*, pages 85–1–85–18. CRC Press, Boca Raton, Florida, 2004.

Journal Papers

All journal papers were peer-reviewed. The Proceedings of the IEEE papers were invited, as all papers in that journal are.

[6] Marcio Buss, Daniel Brand, Vugranam Sreedhar, and Stephen A. Edwards. A novel analysis space for pointer analysis and its application for bug finding. *Science of Computer Programming*, 2009. doi: 10.1016/j.scico.2009.08.002.

[7] Cristian Soviani, Ilija Hadžić, and Stephen A. Edwards. Synthesis and optimization of pipelined packet processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(2):231–244, February 2009.

[8] Osama Neiroukh, Stephen A. Edwards, and Xiaoyu Song. Transforming cyclic circuits into acyclic equivalents. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(10):1775–1787, October 2008.

[9] Stephen A. Edwards and Jia Zeng. Code generation in the Columbia Esterel Compiler. *EURASIP Journal on Embedded Systems*, 2007:Article ID 52651, 31 pages, 2007.

[10] Cristian Soviani, Olivier Tardieu, and Stephen A. Edwards. Optimizing sequential cycles through Shannon decomposition and retiming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3):456–467, March 2007.

[11] Stephen A. Edwards. The challenges of synthesizing hardware from C-like languages. *IEEE Design & Test of Computers*, 23(5):375–386, September/October 2006.

[12] Stephen A. Edwards and Olivier Tardieu. SHIM: A deterministic model for heterogeneous embedded systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(8):854–867, August 2006.

[13] Stephen A. Edwards. Experiences teaching an FPGA-based embedded systems class. *ACM SIGBED Review*, 2(4):56–62, October 2005. Originally presented at the Workshop on Embedded Systems Education.

[14] Stephen A. Edwards and Edward A. Lee. The semantics and execution of a synchronous block-diagram language. *Science of Computer Programming*, 48(1):21–42, July 2003. 16 citations on Google Scholar.

[15] Stephen A. Edwards. Tutorial: Compiling concurrent languages for sequential processors. *ACM Transactions on Design Automation of Electronic Systems*, 8(2):141–187, April 2003. 19 citations on Google Scholar.

[16] Albert Benveniste, Paul Caspi, Stephen A. Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, January 2003. Invited. 174 citations on Google Scholar.

[17] Stephen A. Edwards. An Esterel compiler for large control-dominated systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(2):169–183, February 2002. 37 citations on Google Scholar.

[18] Stephen Edwards, Luciano Lavagno, Edward A. Lee, and Alberto Sangiovanni-Vincentelli. Design of embedded systems: Formal models, validation, and synthesis. *Proceedings of the IEEE*, 85(3):366–390, March 1997. Invited. 272 citations on Google Scholar.

Conference Papers

All conference papers were peer-reviewed. In my area, conference papers are preferred over journals because conferences are more selective and more widely read.

[19] Stephen A. Edwards, Sungjun Kim, Edward A. Lee, Isaac Liu, Hiren D. Patel, and Martin Schoeberl. A disruptive computer design idea: Architectures with repeatable timing. In *Proceedings of the IEEE International Conference on Computer Design (ICCD)*, Lake Tahoe, CA, October 2009.

[20] Baolin Shao, Nalini Vasudevan, and Stephen A. Edwards. Compositional deadlock detection for rendezvous communication. In *Proceedings of the International Conference on Embedded Software (Emsoft)*, pages 59–66, Grenoble, France, October 2009. 33/106 = 31%.

[21] Nalini Vasudevan and Stephen A. Edwards. Buffer sharing in CSP-like programs. In *Proceedings of the International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, Cambridge, Massachusetts, July 2009. 15/42 = 36%.

[22] Nalini Vasudevan and Stephen A. Edwards. A determinizing compiler. In *Proceedings of Program Language Design and Implementation (PLDI)*, Dublin, Ireland, June 2009.

[23] Nalini Vasudevan, Olivier Tardieu, Julian Dolby, and Stephen A. Edwards. Compile-time analysis and specialization of clocks in concurrent programs. In *Proceedings of Compiler Construction (CC)*, volume 5501 of *Lecture Notes in Computer Science*, pages 48–62, York, United Kingdom, March 2009.

[24] Nalini Vasudevan and Stephen A. Edwards. Celling SHIM: Compiling deterministic concurrency to a heterogeneous multicore. In *Proceedings of the Symposium on Applied Computing (SAC)*, volume III, pages 1626–1631, Honolulu, Hawaii, March 2009. 1084/316 = 29%.

[25] Ben Lickly, Isaac Liu, Sungjun Kim, Hiren D. Patel, Stephen A. Edwards, and Edward A. Lee. Predictable programming on a precision timed architecture. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, pages 137–146, Atlanta, Georgia, October 2008.

[26] Nalini Vasudevan and Stephen A. Edwards. Static deadlock detection for the SHIM concurrent language. In *Proceedings of the International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 49–58, Anaheim, California, June 2008.

[27] Nalini Vasudevan, Satnam Singh, and Stephen A. Edwards. A deterministic multi-way rendezvous library for Haskell. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1–12, Miami, Florida, April 2008. 105/410 = 25%.

[28] Stephen A. Edwards, Nalini Vasudevan, and Olivier Tardieu. Programming shared memory multiprocessors with deterministic message-passing concurrency: Compiling SHIM to Pthreads. In *Proceedings of Design, Automation, and Test in Europe (DATE)*, pages 1498–1503, Munich, Germany, March 2008.

[29] Marcio Buss, Daniel Brand, Vugranam Sreedhar, and Stephen A. Edwards. Flexible pointer analysis using assign-fetch graphs. In *Proceedings of the Symposium on Applied Computing (SAC)*, pages 234–239, Fortaleza, Ceará, Brazil, March 2008. 384/1307 = 29.3%.

[30] Stephen A. Edwards and Edward A. Lee. The case for the precision timed (PRET) machine. In *Proceedings of the 44th Design Automation Conference*, pages 264–265, San Diego, California, June 2007. 8/54 = 15% ("WACI" track).

[31] Haim Cohen and Stephen A. Edwards. {sets}—a lightweight constraint programming language based on ROBDDs. In *Proceedings of the IADIS International Conference on Applied Computing*, Salamanca, Spain, February 2007.

[32] Olivier Tardieu and Stephen A. Edwards. Scheduling-independent threads and exceptions in SHIM. In *Proceedings of the International Conference on Embedded Software (Emsoft)*, pages 142–151, Seoul, Korea, October 2006. 31/94 = 33%.

[33] Olivier Tardieu and Stephen A. Edwards. R-SHIM: Deterministic concurrency with recursion and shared variables. In *Proceedings of the International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, page 202, Napa, California, July 2006. 17 papers + 6 posters / 43 = 53%.

[34] Nicholas Jun Hao Ip and Stephen A. Edwards. A processor extension for cycle-accurate real-time software. In *Proceedings of the IFIP International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 4096 of *Lecture Notes in Computer Science*, pages 449–458, Seoul, Korea, August 2006. approx. 125/500 = 25%.

[35] Stephen A. Edwards and Olivier Tardieu. Efficient code generation from SHIM models. In *Proceedings of Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 125–134, Ottawa, Canada, June 2006. 21/83 = 25%.

[36] Cristian Soviani, Ilija Hadžić, and Stephen A. Edwards. Synthesis of high-performance packet processing pipelines. In *Proceedings of the 43rd Design Automation Conference*, pages 679–682, San Francisco, California, July 2006. 180/865 = 20%.

[37] Cristian Soviani, Olivier Tardieu, and Stephen A. Edwards. Optimizing sequential cycles through Shannon decomposition and retiming. In *Proceedings of Design, Automation, and Test in Europe (DATE)*, pages 1085–1090, Munich, Germany, March 2006. 233/834 = 28%, **Best paper award.**

[38] Osama Neiroukh, Stephen A. Edwards, and Xiaoyu Song. An efficient algorithm for the analysis of cyclic circuits. In *Proceedings of the Symposium on VLSI (ISVLSI)*, pages 303–308, Karlsruhe, Germany, March 2006. 64/151 = 42%.

[39] Jia Zeng and Stephen A. Edwards. Separate compilation for synchronous modules. In *Proceedings of the 2nd International Conference on Embedded Software and Systems (ICESS)*, volume 3820 of *Lecture Notes in Computer Science*, pages 129–140, Xi'an, China, December 2005. 140/360 = 39% overall, 63/360 = 17% for proceedings.

[40] Olivier Tardieu and Stephen A. Edwards. Approximate reachability for dead code elimination in Esterel*. In *Proceedings of the Third International Symposium on Automated Technology for Verification and Analysis (ATVA)*, volume 3707 of *Lecture Notes in Computer Science*, pages 323–337, Taipei, Taiwan, October 2005. 33/95 = 35%.

[41] Stephen A. Edwards and Olivier Tardieu. SHIM: A deterministic model for heterogeneous embedded systems. In *Proceedings of the International Conference on Embedded Software (Emsoft)*, pages 37–44, Jersey City, New Jersey, September 2005. 25/88 = 28%, 10 citations on Google Scholar.

[42] Stephen A. Edwards and Olivier Tardieu. Deterministic receptive processes are Kahn processes. In *Proceedings of the International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 37–44, Verona, Italy, July 2005. 17/47 = 36%.

[43] Christopher L. Conway, Kedar S. Namjoshi, Dennis Dams, and Stephen A. Edwards. Incremental algorithms for inter-procedural analysis of safety properties. In *Proceedings of the 17th International Conference on Computer-Aided Verification (CAV)*, volume 3576 of *Lecture Notes in Computer Science*, pages 449–461, Edinburgh, Scotland, June 2005. 32/123 = 26%.

[44] Stephen A. Edwards. The challenges of hardware synthesis from C-like languages. In *Proceedings of Design, Automation, and Test in Europe (DATE)*, pages 66–67, Munich, Germany, March 2005. 176/825 = 21%. 17 citations on Google Scholar., Invited.

[45] Jia Zeng, Cristian Soviani, and Stephen A. Edwards. Generating fast code from concurrent program dependence graphs. In *Proceedings of Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 175–181, Washington, DC, June 2004. 28/120 = 23%.

[46] Christopher L. Conway and Stephen A. Edwards. NDL: A domain-specific language for device drivers. In *Proceedings of Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pages 30–36, Washington, DC, June 2004. 28/120 = 23%.

[47] Stephen A. Edwards. Making cyclic circuits acyclic. In *Proceedings of the 40th Design Automation Conference*, pages 159–162, Anaheim, California, June 2003. 152/628 = 24%. 13 citations on Google Scholar.

[48] Stephen Jan, Paolo de Dios, and Stephen A. Edwards. Porting a network cryptographic service to the RMC2000: A case study in embedded software development. In *Designers' Forum: Design Automation and Test in Europe Conference and Exhibition*, pages 150–155, Munich, Germany, March 2003. 98 long + 54 short + 36 designer's forum/590 = 32%, Also appears as Chapter 13 of *Embedded Software for SoC*, Jerraya, Yoo, Verkest and Wehn eds., Kluwer, 2003.

[49] Sandeep Shukla, Stephen A. Edwards, Jean-Pierre Talpin, and Rajesh K. Gupta. Tutorial: High level modeling and validation methodologies for embedded systems: bridging the productivity gap. In *Proceedings of the 16th International Conference on VLSI Design*, pages 9–14, New Delhi, India, January 2003.

[50] Stephen A. Edwards, Tony Ma, and Robert Damiano. Using a hardware model checker to verify software. In *Proceedings of the 4th International Conference on ASIC (ASICON)*, pages 85–90, Shanghai, China, October 2001.

[51] Stephen A. Edwards. Compiling Esterel into sequential code. In *Proceedings of the 37th Design Automation Conference*, pages 322–327, Los Angeles, California, June 2000. Association for Computing Machinery. 154/445 = 35%, Cited by 47 in Google Scholar.

[52] Gitanjali Swamy, Stephen Edwards, and Robert Brayton. Efficient verification and synthesis using design commonalities. In *Proceedings of the Eleventh International Conference on VLSI Design (VLSI'98)*, pages 542–551, Chennai, India, January 1998.

[53] Robert K. Brayton, Gary D. Hachtel, Alberto L. Sangiovanni-Vincentelli, Fabio Somenzi, Adnan Aziz, Szu-Tsung Cheng, Stephen A. Edwards, Sunil P. Khatri, Yuji Kukimoto, Abelardo Pardo, Shaz Qadeer, Rajeev K. Ranjan, Shaker Sarwary, Thomas R. Shiple, Gitanjali Swamy, and Tiziano Villa. VIS. In *Proceedings of Formal Methods in Computer-Aided Design (FMCAD)*, volume 1166, pages 248–256, Palo Alto, California, November 1996.

[54] Robert K. Brayton, Gary D. Hachtel, Alberto Sangiovanni-Vincentelli, Fabio Somenzi, Adnan Aziz, Szu-Tsung Cheng, Stephen Edwards, Sunil Khatri, Yuji Kukimoto, Abelardo Pardo, Shaz Qadeer, Rajeev K. Ranjan, Shaker Sarwary, Thomas R. Shiple, Gitanjali Swamy, and Tiziano Villa. VIS: A system for verification and synthesis. In *Proceedings of the 8th International Conference on Computer-Aided Verification (CAV)*, volume 1102 of *Lecture Notes in Computer Science*, pages 428–432, New Brunswick, New Jersey, July 1996. Springer. 32/93 = 34%, 367 citations on Google Scholar.

Workshop Papers

All workshop papers were peer-reviewed. Those at IWLS have limited distribution.

[55] Stephen A. Edwards. Concurrency and communication: Lessons from the SHIM project. In *Proceedings of the Workshop on Software Technologies for Future Embedded and Ubiquitious Systems (SEUS)*, volume 5860 of *Lecture Notes in Computer Science*, pages 276–287, Newport Beach, California, November 2009. Springer. Invited.

[56] Stephen A. Edwards, Sungjun Kim, Edward A. Lee, Hiren D. Patel, and Martin Schoeberl. Reconciling repeatable timing with pipelining and memory hierarchy. In *Proceedings of the Workshop on Reconciling Performance with Predictability (RePP)*, Grenoble, France, October 2009.

[57] Stephen A. Edwards and Jia Zeng. Static elaboration of recursion for concurrent software. In *Proceedings of the Workshop on Partial Evaluation and Program Manipulation (PEPM)*, pages 71–80, San Francisco, California, January 2008. 20/74 = 27%.

[58] Cristian Soviani and Stephen A. Edwards. FIFO sizing for high-performance pipelines. In *Proceedings of the International Workshop on Logic Synthesis (IWLS)*, San Diego, California, June 2007.

[59] Olivier Tardieu and Stephen A. Edwards. Instantaneous transitions in Esterel. In *Proceedings of the Workshop on Model-Driven High-Level Programming of Embedded Systems (SLA++P)*, Braga, Portugal, March 2007. 9/16 = 56%.

[60] Becky Plummer, Mukul Khajanchi, and Stephen A. Edwards. An Esterel virtual machine for embedded systems. In *Proceedings of Synchronous Languages, Applications, and Programming (SLAP)*, Electronic Notes in Theoretical Computer Science, pages 1–14, Vienna, Austria, March 2006.

[61] Jia Zeng, Chuck Mitchell, and Stephen A. Edwards. A domain-specific language for generating dataflow analyzers. In *Proceedings of the Sixth Workshop on Language Descriptions, Tools and Applications*, Vienna, Austria, April 2006. 7/21 = 33%.

[62] Stephen A. Edwards. Using program specialization to speed SystemC fixed-point simulation. In *Proceedings of the Workshop on Partial Evaluation and Program Manipulation (PEPM)*, pages 21–28, Charleston, South Carolina, January 2006. 17/29 = 59%.

[63] Cristian Soviani, Stephen A. Edwards, and Angelos Keromytis. Adding a flow-oriented paradigm to commodity operating systems. In *Proceedings of the Workshop on Interaction between Operating System and Computer Architecture (IOSCA)*, pages 1–6, Austin, Texas, October 2005.

[64] Marcio Buss, Stephen A. Edwards, Bin Yao, and Daniel Waddington. Pointer analysis for source-to-source transformations. In *Proceedings of the 5th International Workshop on Source Code Analysis and Manipulation (SCAM)*, pages 139–148, Budapest, Hungary, September 2005. 18/48 = 38%.

[65] Cristian Soviani, Olivier Tardieu, and Stephen A. Edwards. High-level optimization by combining retiming and Shannon decomposition. In *Proceedings of the International Workshop on Logic Synthesis (IWLS)*, pages 16–23, Lake Arrowhead, California, June 2005. 33/67 = 49%.

[66] Cristian Soviani and Stephen A. Edwards. Challenges in synthesizing fast control-dominated circuits. In *Proceedings of the International Workshop on Logic Synthesis (IWLS)*, pages 326–332, Lake Arrowhead, California, June 2005. 34 posters/67 = 51%.

[67] Stephen A. Edwards. SHIM: A language for hardware/software integration. In *Proceedings of Synchronous Languages, Applications, and Programming (SLAP)*, Electronic Notes in Theoretical Computer Science, Edinburgh, Scotland, April 2005. 9/17 = 53%.

[68] Stephen A. Edwards. SHIM: A language for hardware/software integration. In *Proceedings of SYNCHRON*, Schloss Dagstuhl, Germany, December 2004.

[69] Stephen A. Edwards. The challenges of hardware synthesis from C-like languages. In *Proceedings of the International Workshop on Logic Synthesis (IWLS)*, pages 509–516, Temecula, California, June 2004. 33 talks/70 = 47%.

[70] Stephen A. Edwards, Vimal Kapadia, and Michael Halas. Compiling Esterel into static discrete-event code. In *Proceedings of Synchronous Languages, Applications, and Programming (SLAP)*, volume 153(4) of *Electronic Notes in Theoretical Computer Science*, pages 107–121, Barcelona, Spain, March 2004. Elsevier Science. 7/10 = 70%, 12 citations on Google Scholar.

[71] Stephen A. Edwards. High-level synthesis from the synchronous language Esterel. In *Proceedings of the International Workshop on Logic Synthesis (IWLS)*, New Orleans, Louisiana, June 2002. 22 long talks/80 = 28%. 14 citations on Google Scholar.

[72] Stephen A. Edwards. ESUIF: An open Esterel compiler. In *Proceedings of Synchronous Languages, Applications, and Programming (SLAP)*, volume 65(5) of *Electronic Notes in Theoretical Computer Science*, page 71, Grenoble, France, April 2002. Elsevier Science. 13/16 = 81%.

[73] Stephen A. Edwards. Compiling Esterel into sequential code. In *Proceedings of the 7th International Workshop on Hardware/Software Codesign (CODES)*, pages 147–151, Rome, Italy, May 1999. Association for Computing Machinery. 20/90 = 22%.

[74] Gitanjali Swamy, Stephen Edwards, and Robert Brayton. Efficient verification and synthesis using design commonalities. In *Proceedings of the International Workshop on Logic Synthesis (IWLS)*, Tahoe City, California, May 1997.

[75] Arlindo L. Oliveira and Stephen Edwards. Limits of exact algorithms for inference of minimum size finite state machines. In *Proceedings of the Seventh Annual Workshop on Algorithmic Learning Theory (ALT)*, volume 1160 of *Lecture Notes in Computer Science*, pages 59–66, Sydney, Australia, October 1996. Springer-Verlag. 16 long + 8 short/41 = 59%.

Theses

[76] Marcio Buss. *Summary-Based Pointer Analysis Framework for Modular Bug Finding*. PhD thesis, Columbia University, New York, New York, USA, February 2008. CUCS–013–08.

[77] Jia Zeng. *Partial Evaluation for Code Generation from Domain-Specific Languages*. PhD thesis, Columbia University, New York, New York, USA, November 2007. CUCS–048–07.

[78] Cristian Soviani. *High Level Synthesis for Packet Processing Pipelines*. PhD thesis, Columbia University, New York, New York, USA, October 2007. CUCS–041–07.

[79] Stephen Anthony Edwards. *The Specification and Execution of Heterogeneous Synchronous Reactive Systems*. PhD thesis, University of California, Berkeley, 1997. 44 citations on Google Scholar, Available as UCB/ERL M97/31.

[80] Stephen Edwards. An Esterel compiler for a synchronous/reactive development system. Master's thesis, University of California, Berkeley, June 1994. Available as UCB/ERL M94/43.

Technical Reports

[81] Sungjun Kim, Hiren D. Patel, and Stephen A. Edwards. Using a model checker to determine worst-case execution time. Technical Report CUCS–038–09, Columbia University, Department of Computer Science, New York, New York, USA, September 2009.

[82] Devesh Dedhia. Example application under PRET environment — programming a MultiMediaCard. Technical Report CUCS–005–09, Columbia University, Department of Computer Science, New York, New York, USA, January 2009.

[83] Stephen A. Edwards. Retrocomputing on an fpga: Reconstructing an 80's-era home computer with programmable logic. Technical Report CUCS–003–09, Columbia University, Department of Computer Science, New York, New York, USA, January 2009.

[84] Keerti Joshi and Delvin Kellebrew. A MPEG decoder in SHIM. Technical Report CUCS–057–08, Columbia University, Department of Computer Science, New York, New York, USA, December 2008.

[85] Nishant R. Shah. Memory issues in PRET machines. Technical Report CUCS–059–08, Columbia University, Department of Computer Science, New York, New York, USA, December 2008.

[86] David Lariviere and Stephen A. Edwards. uClinux on the Altera DE2. Technical Report CUCS–055–08, Columbia University, Department of Computer Science, New York, New York, USA, December 2008.

[87] Ravindra Babu Ganapathi and Stephen A. Edwards. SHIM optimization: Elimination of unstructured loops. Technical Report CUCS–054–08, Columbia University, Department of Computer Science, New York, New York, USA, December 2008.

[88] Dave Aaron Smith, Nalini Vasudevan, and Stephen Edwards. Static deadlock detection in SHIM with an automata type checking system. Technical Report CUCS–053–08, Columbia University, Department of Computer Science, New York, New York, USA, December 2008.

[89] Nalini Vasudevan, Olivier Tardieu, Julian Dolby, and Stephen A. Edwards. Analysis of clocks in x10 programs (extended). Technical Report CUCS–052–08, Columbia University, Department of Computer Science, New York, New York, USA, December 2008.

[90] Ben Lickly, Isaac Liu, Sungjun Kim, Hiren D. Patel, Stephen A. Edwards, and Edward A. Lee. Predictable programming on a precision timed architecture. Technical Report UCB/EECS-2008-40, University of California, Berkeley, April 2008.

[91] Marcio Buss, Daniel Brand, Vugranam Sreedhar, and Stephen A. Edwards. A new abstraction for summary-based pointer analysis. Technical Report RC24104, IBM, New York, 2007.

[92] Chen-Chun Huang, Javier Coca, Yashket Gupta, and Stephen A. Edwards. An implementation of a Renesas H8/300 microprocessor with a cycle-level timing extension. Technical Report CUCS–051–06, Columbia University, Department of Computer Science, New York, New York, USA, December 2006.

[93] Nalini Vasudevan and Stephen A. Edwards. A JPEG decoder in SHIM. Technical Report CUCS–048–06, Columbia University, Department of Computer Science, New York, New York, USA, December 2006.

[94] Smridh Thapar, Olivier Tardieu, and Stephen A. Edwards. Arrays in SHIM: A proposal. Technical Report CUCS–047–06, Columbia University, Department of Computer Science, New York, New York, USA, December 2006.

[95] Stephen A. Edwards and Edward A. Lee. The case for the precision timed (PRET) machine. Technical Report UCB/EECS-2006-149, EECS Department, University of California, Berkeley, November 2006.

[96] Neesha Subramaniam, Ohan Oda, and Stephen A. Edwards. Macshim: Compiling matlab to a scheduling-independent concurrent language. Technical Report CUCS–038–06, Columbia University, Department of Computer Science, New York, New York, USA, September 2006.

[97] Olivier Tardieu and Stephen A. Edwards. Specifying confluent processes. Technical Report CUCS–037–06, Columbia University, Department of Computer Science, New York, New York, USA, September 2006.

[98] Olivier Tardieu and Stephen A. Edwards. Scheduling-independent threads and exceptions in SHIM. Technical Report CUCS–036–06, Columbia University, Department of Computer Science, New York, New York, USA, September 2006.

[99] Marcio Buss, Stephen A. Edwards, Bin Yao, and Daniel Waddington. Pointer analysis for C programs through AST traversal. Technical Report CUCS–028–05, Columbia University, Department of Computer Science, New York, New York, USA, 2005.

[100] Christopher L. Conway, Kedar S. Namjoshi, Dennis Dams, and Stephen A. Edwards. Incremental algorithms for inter-procedural analysis of safety properties. Technical Report CUCS–018–05, Columbia University, Department of Computer Science, New York, New York, USA, 2005.

[101] Stephen A. Edwards and Chun Li. Determining interfaces using type inference. Technical Report CUCS–052–04, Columbia University, Department of Computer Science, New York, New York, USA, December 2004.

[102] Cristian Soviani, Jia Zeng, and Stephen A. Edwards. Sequential challenges in synthesizing Esterel. Technical Report CUCS–051–04, Columbia University, Department of Computer Science, New York, New York, USA, December 2004.

[103] Stephen A. Edwards. Design and verification languages. Technical Report CUCS–046–04, Columbia University, Department of Computer Science, New York, New York, USA, 2004.

[104] Hanoril Estevez and Stephen A. Edwards. Live CD cluster performance. Technical Report CUCS–037–04, Columbia University, Department of Computer Science, New York, New York, USA, 2004.

[105] Cristian Soviani, Jia Zeng, and Stephen A. Edwards. Improved controller synthesis from Esterel. Technical Report CUCS–015–04, Columbia University, Department of Computer Science, New York, New York, USA, 2004.

[106] Stephen A. Edwards. Design languages for embedded systems. Technical Report CUCS–009–03, Columbia University, Department of Computer Science, New York, New York, USA, 2004.

Professional Activities

Professional Society Memberships

Senior Member, IEEE	2006–
Member, ACM	2006–
Member, IEEE	1994–2006

Standarization Committees

Vice Chair, IEEE P1778 Esterel Standardization Committee, 2007–

Journal Activities

- IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems
  Top journal in my area.

Associate Editor	2006–
Guest Editor, special section on the Intl. Workshop on Logic and Synthesis	May 2006
Reviewer	1994, 2001–2003, 2006–2008

- ACM Transactions on Embedded Computer Systems

Associate Editor	2008–
Reviewer	2004, 2006–2007, 2009

- IEEE Transactions on Industrial Informatics

Associate Editor	2007–
Reviewer	2009

- EURASIP International Journal of Embedded Systems

Associate Editor	2004–
Reviewer	2007–

Conference/Workshop Activities

- Design Automation Conference (DAC)
  Top conference in my area; 15%-20% paper acceptance rate

Technical Subcommittee Chair, *Managed four TPC members and 50+ papers.*	2006–2007
TPC Member, *Responsible for 30+ paper reviews per year.*	2004–2006
Reviewer	1996–2004, 2008–

- Design, Automation, and Test in Europe (DATE)
  Second-to-top conference in my area.

Topic Committee Member	2002–2004, 2007

- International Workshop on Logic and Synthesis (IWLS)
  Main workshop for logic synthesis, approx. 100 attendees

Program Chair	2006
General Chair	2005
Publicity and Publications Chair	2003–2004
TPC Member	2003–2009

- Embedded Systems Week
  Local Arrangements Chair 2005
  Publicity Chair (EMSOFT conference) 2003–2004
  TPC Member (EMSOFT conference) 2004–2006
  TPC Member (CODES+ISSS conference) 2008, 2009
  Reviewer (EMSOFT conference) 2008

- Synchronous Languages, Applications, and Programming (SLAP)
  Steering Committee Member 2006–
  TPC Member 2002–2006

- Memocode conference
  Program Chair 2007–2008
  Publicity Chair 2003–2004, 2006
  TPC Member 2003–2007, 2009 Panel Organizer 2009

- Languages, Compilers, and Techniques for Embedded Systems (LCTES)
  TPC Member 2006

- IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)
  TPC Member 2005, 2006

- Embedded and Ubiquitous Computing (EUC)
  TPC Member 2006–2008

- International Conference on Computer Design (ICCD)
  TPC Member 2004–2005

- Applications of Concurrency to System Design (ACSD)
  TPC Member 2004–2008

- Workshop on Modeling, Validation and Heterogeneity (MoVaH)
  TPC Member 2008

- ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM)
  TPC Member 2008

- International Conference on Hybrid Systems: Computation and Control (HSCC)
  TPC Member 2008

- Language Descriptions Tools, Analysis (LDTA)
  TPC Member 2009

- International Conference on Software Language Engineering (SLE)
  TPC Member 2009

- Real-time Systems Symposium (RTSS)
  TPC Member 2009

Grant Review Panels

NSF, 2002, 2003, 2008.

Paper Reviews

ACM Transactions on Programming Languages and Systems (TOPLAS) 2001
Computer-Aided Verification (CAV) 1996, 2001
Correct Hardware Design and Verification Methods (CHARME) 1999
Design Automation and Test in Europe (DATE) 2001
Formal Aspects of Computing (FAC) 2002
Formal Methods for Industrial Critical Systems (FMICS) 2001
IEEE Transactions on Computers 2003, 2007–2009
International Conference on Computer-Aided Design (ICCAD) 1999–2003
International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 1996
IFIP Workshop on Logic & Architecture Synthesis (IWLAS) 1994
Semantic Foundations of Engineering Design Languages (SFEDL) 2004
Sigmetrics 2004
Formal Methods in System Design (FMSD) 2004, 2007
Integration—The VLSI Journal 2004
International Conference on Embedded Software and Systems (ICESS) 2005, 2008, 2009
Code Generation and Optimization (CGO) 2005
Architectural Support for Programming Languages and Operating Systems (ASPLOS) 2006
Principles of Programming Languages (POPL) 2007
Fundamenta Informaticae 2007–2008
Programming Language Design and Implementation (PLDI) 2008
IEEE Design and Test of Computers 2008
International Colloquium on Automata, Languages and Programming (ICALP) 2008
IEEE Conference on Automation Science and Engineering (CASE) 2008
Electronics and Telecommunications Research Institute (ETRI) Journal 2008
The 6th IFIP Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS) 2008
IEEE Transactions on Very Large Scale Integration Systems (TVLSI) 2008, 2009
IEEE Transactions on Design Automation of Electronic Systems (TODAES) 2008, 2009
Compiler Construction (CC) 2008

Invited Talks

Conferences/Other

What Do We Do With 10^{12} Transistors? The Case for Precision Timing — February 21, 2008
Presented at the DSRC TeraChip Workshop, Stanford, California.

Verification Challenges in the SHIM Concurrent Language — May 18, 2007
Invited talk at the Third Northeast Verification Seminar, NEC, Princeton, New Jersey.

Verification: What Works and What Does Not? — May 18, 2007
Panel at the Third Northeast Verification Seminar, NEC, Princeton, New Jersey.

Using and Compiling Esterel — July 11th, 2005
Invited Tutorial, Memocode conference, Verona, Italy.

The Future of Embedded Linux. June 30, 2005
Panel at C3Expo, New York, NY.

Languages for Embedded Systems August 2–6, 2004
Week-long course at National Chiao Tung University, Hsinchu, Taiwan.

Linux for EDA November 2003
Tutorial at the International Conference on Computer-Aided Design (ICCAD), San Jose, California.

High-Level Modeling and Validation Methodologies for Embedded Systems:
Bridging the Productivity Gap January 4, 2003
Presented at VLSI Design 2003, New Delhi, India.
With Sandeep K. Shukla, Jean Pierre Talpin, and Rajesh K. Gupta.

System-on-a-chip and the Coming Design Revolution November 1, 2002
Invited talk at the Second Annual Emerging Information Technology Conference (EITC), Princeton, New Jersey.

Scaling the Abstraction Cliff: High-level Languages for System Design March 2001
Tutorial A2 at the Design, Automation and Test in Europe (DATE 2001) Munich, Germany.

Universities/Industry

Programming Shared Memory Multiprocessors with Deterministic Message-Passing Concurrency: Compiling SHIM to Pthreads August 8, 2008
National Taiwan University, Taipei, Taiwan

What Do We Do With 10^{12} Transistors? The Case for Precision Timing February 20, 2008
Google, Mountain View, California

Precision-Timed (PRET) Machines January 9, 2007
Altera, San Jose, California

Precision-Timed (PRET) Machines July 6, 2007
National Taiwan University, Taipei, Taiwan

SHIM: A Scheduling-Independent Concurrent Language for Embedded Systems May 10, 2007
Princeton University, New Jersey

SHIM: A Scheduling-Independent Concurrent Language for Embedded Systems April 27, 2007
University of Pennsylvania, Philadelphia

SHIM: A Scheduling-Independent Concurrent Language for Embedded Systems March 16, 2007
MIT, Boston, Massachusetts

SHIM: A Scheduling-Independent Concurrent Language for Embedded Systems March 13, 2007
CEA, Grenoble, France

SHIM: A Scheduling-Independent Concurrent Language for Embedded Systems November 8, 2006
University of California, Berkeley

The Challenges of Hardware Synthesis from C-Like Languages September 18, 2006
ECSI-UBS Workshop on High Level Synthesis, Darmstadt, Germany.

SHIM: A Deterministic Language for Embedded Systems August 28, 2006
SpringSoft, Hsinchu, Taiwan.

SHIM: A Deterministic Language for Embedded Systems August 28, 2006

National Chiao Tung University (NCTU), Hsinchu, Taiwan.

SHIM: A Deterministic Language for Embedded Systems — August 23, 2006
Microsoft Research, Bangalore, India.

SHIM: A Deterministic Language for Embedded Systems — August 11, 2006
Tsinghua University, Hsinchu, Taiwan.

SHIM: A Deterministic Language for Embedded Systems — August 10, 2006
National Taiwan University, Taipei.

SHIM: A Deterministic Language for Embedded Systems — August 4, 2006
Seoul National University, Korea.

SHIM: A Deterministic Language for Embedded Systems — July 21, 2006
University of Kiel, Germany.

SHIM: A Deterministic Model for Heterogeneous Embedded Systems — December 9, 2005
Verimag, Grenoble, France.

SHIM: A Deterministic Model for Heterogeneous Embedded Systems — November 10, 2005
University of California at Berkeley.

SHIM: A Deterministic Model for Heterogeneous Embedded Systems — November 9, 2005
Xilinx, San Jose, California.

SHIM: A Deterministic Model for Heterogeneous Embedded Systems — October 7th, 2005
National Instruments and the University of Texas at Austin.

SHIM: A Deterministic Model for Heterogeneous Embedded Systems — August 16th, 2005
Tsinghua University, Hsinchu, Taiwan.

Deterministic Receptive Processes are Kahn Processes. — June 22, 2005
INRIA, Sophia-Antipolis, France.

SHIM: A Language for Hardware/Software Integration. — April 7, 2005
University of California, Irvine.

Using and Compiling Esterel — August 17, 2004
National Chung Cheng University, Chai-Yi, Taiwan.

Making cyclic circuits acyclic — March 3, 2003
Carnegie Mellon, Pittsburgh.

Compiling Esterel — January 13, 2003
Indian Institute of Technology, Delhi.

Compiling Esterel into Better Circuits and Faster Simulations. — November 5, 2002
Intel, Hillsboro, Oregon.

Compiling Esterel — October 10, 2002
Cambridge University, UK.

Compiling Esterel — September 5, 2002
University of California, Berkeley.

Compiling Esterel — August 26, 2002
University of Calgary, Alberta, Canada.

Compiling Esterel — August 19, 2002

Microsoft Research, Redmond, Washington.

High-level Synthesis from the Synchronous Language Esterel — August 8, 2002
Intel, Hillsboro, Oregon.

An Overview of the Electronic Design Automation (EDA) Field — July 16, 2002
Yuan Ze University, Chungli, Taiwan.

Compiling Esterel — July 8, 2002
National Taiwan University (Taida), Taipei, Taiwan.

Compiling Esterel — April 2002
A discussion of my first Esterel compiler along with ongoing work on ESUIF.
Princeton, New Jersey.

ESUIF: An Open Esterel Compiler — March 2002
A work-in-progress description of the ESUIF Esterel compiler.
IRISA/INRIA Rennes, France.

Esterel and Other Projects — October 2001
A summary of existing Esterel work and future plans
Intel, Hillsboro, Oregon.

Compiling Esterel into Sequential Code, — April 28, 1999
University of California, Berkeley, CAD Group Seminar.

Synchronous Reactive Systems. — February 1997
University of Texas, Austin.

Departmental

Ph.D Students

Baolin Shao	Fall 2008–present
Sungjun Kim	Spring 2008–present
Kristina Chodorow	Fall 2007
Nalini Vasudevan	Spring 2007–present
Marcio Buss	Spring 2004–Spring 2008
Jia Zeng	Fall 2002–Spring 2008
Cristian Soviani	Fall 2002–Fall 2007
Christopher L. Conway	Summer 2002–Spring 2006

Postdoctoral Researchers

Olivier Tardieu	Spring 2005–Fall 2006

External PhD Committees

Francesco Leonardi, University of Trento, Italy	March 2009
Jan Lukoschus, University of Kiel, Germany	July 2006
Matthieu Moy, Verimag, Grenoble, France	December 2005
Olivier Tardieu, Ecole des Mines, Sophia-Antipolis, France	October 2004
"Jacky" Dumitru Potop-Butucaru, Ecole des Mines, Sophia-Antipolis, France	November 2002
Sean Gibb, University of Calgary, Alberta, Canada	August 2002

Internal Dissertation Defenses

Cheng-Hong Li (Luca Carloni)	Jul 22, 2009
Cheoljoo Jeong (Steve Nowick)	October 22, 2007
Cristian Soviani	August 30, 2007
Jia Zeng	August 30, 2007
Krysta Svore (Al Aho)	March 31, 2006
Saravanan Rajapandian (Ken Shepard, EE)	May 20, 2005
Tiberiu Chelcea (Steve Nowick)	December 8, 2003
Yu Zheng (Ken Shepard, EE)	June 9, 2003
Henry Li (Charles Zukowski, EE)	Febriary 1, 2002
Michael Theobald (Steve Nowick)	December 18, 2001

Internal Thesis Proposal Committees

Cheng-Hong Li (Luca Carloni)	December 3, 2007
Mark Eaddy (Al Aho)	December 13, 2006
Marcio Buss	April 27, 2006
Jia Zeng	May 2, 2006
Cristian Soviani	May 2, 2006
Krysta Svore (Al Aho)	March 31, 2006
Chris Conway	May 12, 2006

Committees

PhD Committee	2003–
Academic Commitee	2003–
Recruiting	
Hosted one candidate	2002
Hosted two candidates	2004
Hosted three candidates	2007
Hosted one candidate	2008
Visibility Committee	2005
Revamped CS department web page	

Undergraduate/Masters Research Projects Supervised

Student	Work resulted in		Year
	Tech. Report	Published Paper	
Devesh Dedhia	•		2008
Delvin Kelleybrew	•		2008
Keerti Joshi	•		2008
Nishant R. Shah	•		2008
Ravindra Babu Ganapathi	•		2008
Dave A. Smith	•		2008
Phong Pham			2007
Haim Cohen		•	2006
Chen-Chun Huang	•		2006
Javier Coca	•		2006
Yashket Gupta	•		2006
Wei Chung Hsu	•		2006
David Lariviere	•		2006
Smridh Thapar	•		2006
Nalini Vasudevan	•		2006
Neesha Subramaniam	•		2006
Ohan Oda	•		2006
Bryan Gwin			2006
Mukul Khajanchi		•	2006
Rebecca Plummer		•	2005
Nicholas Ip		•	2005
Chun Li	•		2004
John Shick			2004
Erin Adelman			2004
Hanoril Estevez	•		2004
Clarke Landis			2003
Noel Vega	•		2003
Michael Anikin			2003
Michael Halas		•	2003
Vimal Kapadia		•	2003
Seema Gupta			2003
Miquad Mohammad			2003
Thomas Heydt-Benjamin	•		2003
Mikhal Litvin			2003
Avi Shinnar		•	2002
Jose Brunheroto			2001

Teaching

I developed CSEE W4840: Embedded System Design—a new "capstone" lab course for Computer Engineering and Electrical Engineering majors. Seniors are required to take one of three such courses. In it, students design and implement embedded systems using state-of-the-art FPGA boards. Projects have included video games, music synthesizers, and robots.

Semester	Sp 2004	Sp 2005	Sp 2006	Sp 2007	Sp 2008
Enrollment	57	45	40	34	25

Student comments:

"Prof. Edwards is an amazing teacher. I have always enjoyed his lectures." (Spring 2007)

"His in class presentation is the best I have seen. He is approachable by students and always gives a straight answer to his students. Finally he takes an interest in what the student is doing though out the course." (Spring 2007)

"Professor Edwards is a very knowledgeable teacher who keeps the classroom interesting and moreover is very help in one-on-one lab settings. He has a genuine interest in what you are trying to accomplish and this is a first that I have seen among all Columbia professors during my 4 years here. To me he seems like the epitome of what a teacher should be." (Spring 2006)

"I've always found Prof. Edwards to bring passion and energy to his lectures. He makes it easy to want to learn more." (Spring 2006)

"For the time and effort he puts into his presentations and labs. Edwards and his TAs do almost as much work as the students." (Spring 2005)

I also teach COMS W4115: Programming Languages and Translators, a senior/masters compilers class:

Semester	Sp 2002	Sp 2003	F 2003	F 2004	F 2005	F 2006	Sp 2007	F 2007
Enrollment	107	86	58	52	59	55	58	68

Student comments:

"Professor Edwards was absolutely incredible and I almost failed." (Fall 2006)

"Professor Edwards is the most accessible instructor I've ever had - and he teaches a class about compilers. He's an amazing man." (Fall 2006)

"His classes are extremely enjoyable the best course I have taken so far." (Fall 2006)

"Of the four courses that I took this semester all had good professors but Stephen Edwards was outstanding in his delivery, his ability to use wit [made] his material interesting, and in the quality of the notes and management of the course. I would therefore give him my vote for the Distinguished Faculty award." (Fall 2006)

"This is a course that I did not really want to take and have little aptitude for. Professor Edwards turned something which could have been very uninteresting into something lively and interesting—this is something I really appreciate." (Fall 2006)

"Great professor made subject interesting and explained difficult concepts well. I felt like I even learned from the exams." (Fall 2005)

"Now I know why people call [him] one of the best professors. His wit in the class just make everyone fall into the class." (Fall 2005)

"Prf. Edwards is one of the few lecturers I've met who actually makes the classes very interesting. He's always inserting funny anecdotes and ties in stories with concepts really well. It s been a pleasure to take PLT with him. Even outside the classroom Prof. Edwards is very friendly and fun to talk to. Would definitely be happy to see him get the SEAS Distnguished Faculty Award." (Fall 2005)

"Edwards was a great lecturer a great motivator and created a great class. The class was so formative in my CS educational career that I would consider it the most valuable and one of the most interesting." (Fall 2005)

"His demeanor inside and outside of the classroom is truly one of the greatest I have seen in my four years at Columbia. It is one of the few classes in the engineering school I look forward to going to everyday just to see what he'll talk about next." (Fall 2005)

"Prof. Edwards is a great lecturer and really knows his stuff. The class is really well organized." (Fall 2004)

"One of the best and most useful classes I have ever taken in the CS department." (Fall 2004)

"I really enjoyed the classroom delivery. The professor did a great job in explaining the concepts. I never felt lost in the material." (Fall 2004)

COMS W4995–02: Languages for Embedded System Design
Graduate-level class of my own devising based on my book.

Semester	F 2001	F 2002
Enrollment	23	16

Student Teaching

University of California, Berkeley 1995
Head teaching assistant.
Supervised seven teaching assistants overseeing a 200-student upper-division logic design course (CS 150, taught by Richard Newton). Designed and wrote labs. Conducted weekly recitation section, gave substitute lectures to entire class, graded tests.

California Institute of Technology 1990–1992
Teaching assistant.
Wrote assignments and conducted a recitation section for the 30-student microprocessor design course (CS 51, 52, and 53). Designed sound I/O circuitry and PC board for class project.

EXHIBIT C

Claim #42	Infringement Analysis	Infringement by Software Developers Using Courgette Code
A method for generating a compact difference result between an old data table and a new data table;	"Rather than push out a whole new 10MB update, we send out a diff that takes the previous version of chrome and generates the new version." *Chromium Developer Documentation,* http://dev.chromium.org/developers/design-documents/software-updates-courgette (hereinafter *CDD*). Here, the old data table and new data table consist of the old and new versions of Google chrome, respectively.	A software developer, or anyone with a C++ compiler and little experience, would be able to use the code posted by Google at http://src.chromium.org/viewvc/chrome/trunk/src/courgette/ and the comments and instructions provided to generate a compact difference result between versions of his or her own products. The old data table would be a prior version of the developer's program and the new data table would be the new version.
each data table including reference entries that contain reference that refer to other entries in the data table;	"The compiled code is full of internal references where some instruction or datum contains the address (or offset) of another instruction or datum." *CDD.*	A compiled version of any program would be full of internal references where some instruction or datum contains the address (or offset) of another instruction or datum.
the method comprising the steps of:		
(a) generating a modified old data table utilizing at least said old data table;	"Courgette transforms the program into a primitive assembly language and does the diffing at the assembly level: . . . asm_old = disassemble(original)" *CDD.* Therefore, the modified old data table is generated when Courgette "transforms" the program into a "primitive assembly language."	The posted Courgette code can be compiled into an executable following the instructions posted at http://www.chromium.org/developers/how-tos/get-the-code. The executable generated in part from file courgette_tool.cc is used to disassemble an old version of a program. The method PrintHelp in courgette_tool.cc instructs on usage of the tool. This method lists the valid commands for the tool. For example the command:

Claim #42	Infringement Analysis	Infringement by Software Developers Using Courgette Code
		"courgette -dis <executable_file> <binary_assembly_file>" will cause the disassemble method to be executed, where <executable_file> is the old data table and <binary_assembly_file> is the disassembled output of the process.
(b) generating a modified new data table utilizing at least said new data table,	"Courgette transforms the program into a primitive assembly language and does the diffing at the assembly level: . . . asm_new = disassemble(update); asm_new_adjusted = adjust(asm_new, asm_old)." *CDD.* Therefore, a modified new data table is generated when Courgette "transforms" the program into a "primitive assembly language."	The user is instructed to disassemble and adjust using the courgette executable by invoking it using a different switch, -disadj. This parses the new version of the program, and creates a disassembled new version.
said modified old data table and modified new data table have at least the following characteristics:		

Claim #42	Infringement Analysis	Infringement by Software Developers Using Courgette Code
(i) substantially each reference in an entry in said old data table that is different than corresponding entry in said new data table due to delete/insert modifications that form part of the transition between said old data table and new data table are reflected as invariant references in the corresponding entries in said modified old and modified new data tables;	As part of the above "disassemble(original)" step, method DefaultAssignIndex in file assembly_program.cc creates new references in program "m" (the old "data table") by assigning indices to labels. Method AssignOne of adjustment_method.cc copies these references to the corresponding labels of program "p" (the new "data table"), thereby rendering them invariant.	The user is instructed to use the –dis (disassemble process) on the old program, and the –disadj (disassemble and adjust process) on the new version of the program, in order to scan the programs and encode them. Comments in the Courgette source code state that an encoded program is a set of tables that contain a simple 'binary assembly language' that can be assembled to produce a sequence of bytes, for example, a Windows 32-bit executable. (*See* encoded_program.h and related files). During the generation of the encoded program, references are replaced with labels, which are invariant references where the references refer to corresponding entries in the modified old and modified new data tables.
(c) generating said compact difference result utilizing at least said modified new data table and modified old data table.	"Courgette transforms the program into a primitive assembly language and does the diffing at the assembly level: . . . asm_diff = bsdiff(asm_old, asm_new_adjusted)" *CDD*. Therefore, the difference calculation operates on the modified old program (asm_old) and the modified new program (asm_new_adjusted). This is done in method GenerateEnsemblePatch which calls method GenerateSimpleDelta which in turn calls the CreateBinaryPatch method of bsdiff, a binary difference utility.	To generate the difference result, the user of the courgette code is instructed to run the courgette executable with the argument -gen specifying the old version, new version, and location for the patch file to be generated.

Claim 43	Infringement Analysis	Infringement by Software Developers Using Courgette Code
(d) transmitting said compact difference result over a communication network.	"Rather than push out a whole new 10MB update, we send out a diff that takes the previous version of chrome and generates the new version." (*CDD*). "We want smaller updates because it *narrows the window of vulnerability*. If the update is a tenth of the size, we can push ten times as many per unit of bandwidth. We have enough users that this means more users will be protected earlier. A secondary benefit is that a smaller update will work better for users who don't have great connectivity." (CDD) The difference results are transmitted from a server to Chrome users and this transmission invariably occurs over a communication network.	The references in the CDD to "bandwidth," "connectivity," and "push out" make explicit to a person of ordinary skill in the art that Courgette is intended for transmitting difference results over a communication network.

Claim 44	Infringement Analysis	Infringement by Software Developers Using Courgette Code
The method of claim 43, wherein said network includes the Internet.	"Google Chrome is a web browser that runs web pages and applications with lightning speed." (http://www.google.com/chrome/intl/en/features.html) "The web" is understood by a person of ordinary skill in the art to operate on the Internet.	The references in the CDD to "bandwidth," "connectivity," and "push out" make explicit to a person of ordinary skill in the art that Courgette is intended for transmitting difference results over a communication network, such as the Internet.