

Steve Haynal

CONTACT INFORMATION	steve@softerhardware.com 503-781-2240	http://www.softerhardware.com <i>Mailing address available upon request</i>
OBJECTIVE	Full-time, part-time, contracting or consulting work that leverages my computer engineering expertise and allows me to telecommute from Walla Walla, Washington	
EXPERTISE	Front-end CAD tools and methodologies for VLSI design, High-level design and synthesis, RTL (Verilog, SystemVerilog, VHDL), FPGA-based design, VLSI design, Formal methods, Symbolic techniques, Fast Fourier Transform, Programming (Python, C/C++, functional languages)	
EDUCATION	Ph.D., Electrical & Computer Engineering, University of California, Santa Barbara, 2000 <ul style="list-style-type: none">• Dissertation: <i>Automata-Based Symbolic Scheduling for High-Level Synthesis</i> M.S., Electrical & Computer Engineering, University of California, Santa Barbara, 1997 B.S., Pacific Union College, Angwin, 1991	
PROFESSIONAL EXPERIENCE	Research Scientist, SofterHardware Researched & developed tools and techniques for generating and searching families of FFT algorithms	2010 – 2011
	Research Scientist, Intel Researched & developed CAD tools and methodologies, primarily for the front-end of VLSI design, RTL and higher, as detailed in the following topics: <i>SystemVerilog Refactoring and Rewriting</i> – Following the spirit of software refactoring, this work introduces both manual (with automatic test generation) and automated refactoring to register transfer languages, RTL, used to specify hardware design. Demonstrated applications include abstracting legacy RTL to improve human readability as well as preparing RTL for broader reuse and inclusion in an IP library. <ul style="list-style-type: none">• Abstracted the Execution Unit RTL from a legacy Pentium to 1/3 its size in terms of source lines of code. Each small rewrite step applied to the Pentium was verified with SAT-based FEV tools. The final abstracted RTL was validated by booting Linux on a FPGA-based emulation platform.• Developed automated refactoring tools to refactor chipset bus interface Verilog to meet specific stricter CPU coding guidelines. This automated refactoring prepares the chipset RTL for broader reuse and inclusion in a corporate IP library without months of manual rewrite effort.• All software development was done on Linux platforms with Python and C++. A commercial library from Verific, distributed at source-level, was substantially extended to enable this work. <i>Integrated Design and Validation System</i> – The Integrated Design and Validation system, or IDV, is a design by refinement system where small but formally verified changes are applied to a netlist model. Hundreds of these small transformations can refine a relatively abstract netlist model into a performance implementation. <ul style="list-style-type: none">• Developed the RTL import module. This module synthesizes RTL into a high-level IDV netlist model.• Integrated logic synthesis tools for pre-, post- and pocket-synthesis applications. Using IDV's ability to quickly and graphically carve out small "pockets" of a design, logic synthesis can be applied locally to preprocess the design and better direct traditional batch synthesis tools or postprocess the design to fix local problems after traditional	2001 – 2009

batch synthesis. Alternatively, traditional batch synthesis tools can be eliminated by using a combination of pocket synthesis and manual design with IDV.

- Applied IDV to real design to refine the methodology and direct tool development. This included specifying in reFLect, a functional language, a high-level netlist model of a IA-64 bus queue. A functional block from the IA-64 model was refined to a performance implementation. Pre- and post-synthesis gains were demonstrated on other IA-64 functional blocks.
- Developed automated rewrites and applied this to create an automated speculation transformation. This extended IDV's original classes of transformations (trusted automated and verified manual) to include complex but verified automated transformations. This was applied to create speculation transformations which are used to fix locally slow logic via duplicating logic and speculating on late signals.
- Implemented miniPBS, a light-weight production-based specification language to capture finite state machines in IDV. Several FSMs from a legacy Pentium were respecified in miniPBS in a more concise and malleable form.
- Created a project-wide regression suite and methodology.
- Developed an XML import module to import tables from microprocessor designs as an initial netlist model.
- All software development was done on Linux platforms with reFLect (a functional language), tcl/tk, and C++. A team of 5 to 10 people, with varying size through its history, worked on IDV.

Automata-Based Symbolic Scheduling – Using formal technologies (binary decision diagrams), Automata-Based Symbolic Scheduling or ABSS solves high-level synthesis scheduling problems exactly.

- Applied ABSS to a unit from the Pentium 4 processor. This demonstrated that given the design constraints to work under, the unit designer had chosen an optimal schedule for his unit.
- Applied ABSS to fine-grained scheduling of IA-64 math library routines. This showed that ABSS can be used for detailed software scheduling and identified optimal schedules for math routines under the unique constraints of a specific IA-64 architecture.
- Applied ABSS to a chipset graphics pipelines. This revealed implementations with lower latency given varying resource constraints.
- All software development was done on Linux platforms with Python and C++. An open source BDD library enabled this research.

Academic Mentoring and Service – A goal of Intel's Strategic CAD labs is to work with academics to enable better and more Intel-relevant research in VLSI CAD.

- Mentored CMU research in applying the Bluespec language to specify a simple IA-64. This design was emulated on an FPGA platform and also required writing Linux software and drivers for testing.
- Mentored UC Santa Barbara research in developing a design specification language, pyddl, based on synchronous elastic flow semantics. Pyddl was used to design a multithreaded, low-latency microcontroller for real-time control loop applications.
- Visited UC Santa Barbara for 3 months to work with professors on CAD problems of interest to Intel.
- Conference service: DAC 2003, DATE 2003 track committee member, ICCAD 2003 session co-moderator, ICCD 2006 track chair.

Owner, SofterHardware

1998 – 2001

Designed, prototyped and tested FPGA-based quadrature decoder PCI cards, Verilog IP cores, wrote Linux/Windows drivers and software libraries

Engineering Technician, Fischer Computer Systems

1989 – 1993

Designed, prototyped and tested quadrature decoder products as well as microcontroller-based paintball speed sensor

TEACHING EXPERIENCE	Contract Teacher, Walla Walla University 2010 <ul style="list-style-type: none"> • Taught course in Digital Logic for Engineering Department
	Contract Teacher, Technical Research Associates, Inc. 1998 – 2000 <ul style="list-style-type: none"> • Developed and taught intensive courses in Visual C++
	Teaching Assistant, University of California, Santa Barbara 1995 – 1999 <ul style="list-style-type: none"> • Developed and supervised labs in digital design, computer architecture • Supervised other teaching assistants as lead teaching assistant
	High School Teacher, San Fernando Valley Academy, Northridge 1993 – 1995 <ul style="list-style-type: none"> • Taught computer literacy, physics, chemistry, earth science, biology and math
PUBLICATIONS	<p>Haynal, S. and Haynal, H., “Generating and Searching Families of FFT Algorithms”, <i>Journal on Satisfiability, Boolean Modeling and Computation</i>, To appear, preprint at http://arxiv.org/abs/1103.5740, 2011.</p> <p>Haynal, S., “Refactoring to Prepare RTL for Reuse”, <i>IP’08</i>, http://www.design-reuse.com/articles/20119/refactoring-rtl-reuse.html, 2008.</p> <p>Haynal, S., et al., “A SystemVerilog Rewriting System for RTL Abstraction with Pentium Case Study”, <i>MEMOCODE 2008</i>, pp. 79-88.</p> <p>Brewer, F. and Haynal, S., “Symbolic NFA scheduling of a RISC microprocessor”, <i>IEEE Trans. VLSI Syst.</i>, vol. 10, no. 4, pp. 429-434, 2002.</p> <p>Haynal, S. and Brewer, F., “Automata-Based Symbolic Scheduling for Looping DFGs”, <i>IEEE Trans. Computers</i>, vol. 50, no. 3, pp. 250-267, 2001.</p> <p>Haynal, S. and Brewer, F., “Representing and Scheduling Looping Behavior Symbolically”, <i>ICCD 2000</i>, pp. 552-555.</p> <p>Haynal, S. and Brewer, F., “A Model for Scheduling Protocol-Constrained Components and Environments”, <i>DAC 1999</i>, pp. 292-295.</p> <p>Haynal, S. and Brewer, F., “Efficient encoding for exact symbolic automata-based scheduling”, <i>ICCAD 1998</i>, pp. 477-481.</p> <p>Haynal, S. and Parhami, B., “Arithmetic Structures for Inner-Product and Other Combinations Based on a Latency-Free Bit-Serial Multiplier Design”, <i>Proc. of the 30th Asilomar Conf. on Signals, Systems and Computers</i>, pp. 197-201, 1996.</p> <p>Haynal, S., “A Microcontroller Based Multimode Reader”, <i>QEX</i>, June 1991.</p>
INVITED TALKS	<p>Haynal, S., “VLSI Design at Intel”, Walla Walla University, Spring 2009 and Spring 2010.</p> <p>Haynal, S., “Integrated Design and Validation”, University of California at Santa Barbara, Fall 2005.</p> <p>Haynal, S., “Automata-Based Symbolic Scheduling”, Carnegie Mellon University, Fall 2002.</p> <p>Haynal, S., “Automata-Based Symbolic Scheduling”, University of Colorado at Boulder, Spring 2002.</p>
HONORS AND AWARDS	<p>Delco Defense Systems Operations Graduate Fellowship, 1998.</p> <p>Experienced category, 2nd place, Univ. of Michigan’s VLSI design contest, 1997.</p> <ul style="list-style-type: none"> • Paper, Slides <p>Teacher of the Year at San Fernando Valley Academy, 1995.</p> <p>Richard E. Fischer Award for excellence in technology, 1990.</p>
INTERESTS	<p>Linux, Open Source, Playing the Piano, Singing, Computer Music and Audio DSP, Amateur Radio, GPGPU, Electric Vehicles, Home Energy, Ethnic Foods, Travel, Gardening, Reading</p>
REFERENCES	<p><i>Available on request</i></p>