Toward CAD-IP Reuse: A Web Bookshelf of Fundamental Algorithms

Andrew E. Caldwell
Simplex Solutions

Igor L. Markov
University of Michigan

Andrew B. Kahng
University of California at San Diego

The free reuse of code and other types of CAD intellectual property can help EDA vendors, captive CAD organizations, and researchers address the design technology productivity gap. The Marco GSRC Bookshelf, a medium for CAD-IP reuse, is now operational and accessible to the general public over the Internet.

ACCELERATING CHANGES in process technology, system implementation platforms, and electronics markets make future design technology requirements uncertain. A new design technology's time to market—from understanding the design problem to integrating a new solution in mainstream flows—can span several process generations and the life of entire electronics markets. Thus, for designers, obtaining the right design technology at the right time is difficult. This situation demands that the entire design technology community (commercial vendors, captive CAD organizations, and academic researchers) focus on improving the delivery of design technology. Put another way, the community must address the well-known design productivity gap by addressing the less well known design *technology* productivity gap.

Today's design technology landscape shows many signs of the design technology productivity gap. Phrases such as "design productivity gap" (for example, in the *International Technology Roadmap for Semiconductors*) and the existence of the Marco Design and Test Focus Research Center reflect a perceived need to improve the effectiveness of CAD-algorithm research. In 1999, the Design Automation Conference found it necessary to institute a new topic area: Fundamental CAD Algorithms. At a more technical level, research fragmented across too many subcommunities indicates a need to clarify the technological leading edge. And, at the level of individual problem formulations (specifically, hypergraph bipartitioning), we have noticed cases of silent, undocumented implementation decisions that change result quality by more than 400%.[1] We also know of two papers published since April 1998 that report more than 1,000% differences in solution costs returned by implementations of the same well-known algorithm.[1]

To help the CAD community address these problems, we have developed a new medium for the reuse of CAD intellectual property, under the auspices of the Marco Gigascale Silicon Research Center (http://www.gigascale.org/). This new medium, the Marco GSRC Bookshelf of Fundamental CAD Algorithms, is an Internet clearinghouse and repository for CAD IP. The Bookshelf—along with a technol-

0740-7475/02/$17.00 © 2002 IEEE

ogy extrapolation system (GTX) and a measurement infrastructure (Metrics)—is one of three initiatives in the GSRC's "Calibrating Achievable Design" research theme (http://vlsicad.ucsd.edu/GSRC/) that seek to improve design technology productivity.

Why reuse?

The usual criteria for technology delivery are time to market and result quality achieved within given resource constraints. These criteria strongly motivate a culture of reuse throughout the design technology community.

With respect to time to market, better software development processes can improve design technology productivity. Expected time to market grows with project risk, which in turn grows with the amount of code that must be written from scratch. Cheaply reusing existing codes with documented performance and a history of successful reuse can reduce risk and shorten time to market even if new features or interfaces must be added.

Even to measure result quality, designers must thoroughly understand algorithms and software tools, including common benchmarks, evaluation methodologies, and known-good performance results. The analysis of hypergraph partitioning in our earlier work shows what can happen when any aspect of a sophisticated technology is unclear.[1] Such risks increase as the research domain matures and the literature expands—in other words, as a problem gains importance and attention. When individual researchers can no longer keep track of all relevant research, there is a risk they will poorly reinvent the wheel. Designers and researchers need an infrastructure that clearly identifies the best results in the CAD field at any given time. This infrastructure should reuse accumulated knowledge about standard benchmarks, evaluation methodologies, and performance comparisons. To build such an infrastructure, we must move beyond mere code reuse and consider a generalized form of reuse—that is, the reuse of CAD IP.

CAD research tends to focus on narrow optimizations—for example, efficient manipulation of binary decision diagrams in logic synthesis or netlist hypergraph partitioning in physical design. If this narrowness is accompanied by the inability to evaluate new results in the context of full design flows, the usefulness of research suffers. (In another article, we describe instances of academic research continuing to use outdated context for specialized optimizations long after industry changed to a different context.[2]) A shared public infrastructure for the evaluation of specialized algorithms would highly benefit the design technology community, at a small amortized cost.

Finally, with respect to resource constraints, the short supply of EDA engineers is widely lamented. Commonly proposed solutions focus on increasing the supply—for example, through outreach to graduate-level education programs. At the same time, many companies and research groups reimplement basic software components such as format readers and writers, delay calculators, circuit partitioners, and technology mappers. For mature technologies whose details (apart from their existence and correctness) do not affect their competitiveness in the marketplace, reimplementation incurs tremendous waste. Resource imperatives alone will force data models, polygon database implementations, placers and routers, and so forth to evolve into commoditized (but nearly free) foundation IP, following the course of operating systems, data structures, and graphical-user-interface components before them. This evolution is consistent with both the culture of "coopetition" (collaboration among competitors), as seen in the histories of semiconductor manufacturing and consortia such as Sematech, and the impetus toward disaggregation (which allocates limited resources to the areas where they add the greatest value). Even for those who don't embrace these trends, reuse has benefits: When a technology becomes freely available, companies no longer need to raid competitors to reimplement it, and the ability to produce reusable code benefits internal research and development.

Free reuse of IP (including software) has a positive effect on all aspects of design technology productivity: time to market, result quality, and effective use of R&D resources. The design technology community requires an adequate level of IP reuse, in the form of convenient and

Design- and CAD-IP reuse

The key goals of VLSI design are to meet time-to-market and quality requirements. Challenges to design success arise from the increasingly complex electrical engineering, optimization, and constraint satisfaction problems inherent in the design process. Mistakes are costly yet almost inevitable. To make matters worse, verification and test complexity grows with the complexity of other design tasks. The classic response to these challenges is design-IP reuse, which includes

- modular design IP specifically created for reuse,
- formal or informal certification and socketization of modular IP,
- matching services that connect IP providers with prospective users, and
- IP protection (for commercial applications).

For CAD-tool providers, the goals and challenges are strikingly similar to those faced by VLSI designers. The complexities of optimization, constraint satisfaction, and software engineering hamper providers from meeting time-to-market and result-quality goals. Accounts of seven-year development cycles from research to production tool are common. Verification and test of software systems and optimization heuristics are poorly understood, so mistakes can be a showstopper. To successfully respond to these challenges, the CAD community must adopt CAD-IP reuse. Since the Marco GSRC Bookshelf's inception in 1999, the concept of CAD-IP reuse has gained considerable momentum in the EDA industry; notable examples include the LEF/DEF and OpenAccess source-code releases.

consistent evaluation methodologies, to identify the best available solutions for important problems. Without this infrastructure, the field is hobbled by barriers to research progress and the inability to identify and adopt appropriate research results for use in production design tools and methodologies. The "Design- and CAD-IP reuse" sidebar presents further motivation for a reuse infrastructure.

What is CAD IP?

CAD IP encapsulates the theory and know-how behind the creation, evaluation, and use of CAD tools. We can contrast CAD IP with design IP, which is typically captured, produced, and managed by CAD tools. Just as design IP includes many components (for example, test harnesses, simulation vectors, technology files, constraints, and layout geometries), so does CAD IP. The following are CAD-IP components[3]:

- data models that provide consistent semantics and data structuring throughout the design flow (ideally with an accompanying canonical application programming interface);
- mathematical problem formulations for optimization and constraint satisfaction that isolate the fundamental difficulties in particular design tasks and encourage solver reuse;
- use models and context descriptions for problem formulations;
- test cases with corresponding high-quality solutions—for both individual problem formulations and integrated tool flows;
- descriptions and theoretical analyses of algorithms;
- executable implementations, including not only solvers (algorithms), but also interchange-format parsers and converters, legality checkers, and cost-function evaluators;
- leading-edge-performance results, as well as standard algorithm comparison and evaluation methodologies, to ensure that newly proposed methods are improvements over previous methods;
- software design and implementation methodologies; and
- implementation source code.

Neither source code nor executables, nor the traditional medium of written algorithm descriptions (typically, six pages of 9-point, two-column format in a proceedings), dominates CAD IP. All the CAD-IP components are critical to design technology progress.

Background

A number of earlier developments have enabled CAD-IP reuse: electronic publishing, implementation software repositories, benchmarking sites, algorithm evaluation methodologies, openness, and software engineering.

Electronic publishing

Electronic publishing was established to convert existing journals into electronic form to reduce production costs, achieve faster dissemination, and provide continual availability. Early

interest in electronic publishing was fueled by the potential for cheap, fast, low-quality publications to drive established publications out of business (see an article by Odlyzko, for example[4]) and had detrimental effects on the research community. The scholarly community urged peer review to increase quality, and by 1997 several organizations (the American Mathematical Society, for example) had established high-quality, low-volume online journals. Today, high-quality online journals flourish in hundreds of research communities, and many are competitive with paper-based publications. In recent years, articles on electronic publication have addressed economics, IP issues, competition between media types, and electronic libraries. Noting new information types, use models, and requirements for dissemination of scholarly information, the literature concludes that the new types and larger amounts of research-related information require new forms of presentation.

Implementation software repositories

With the growing popularity of the World Wide Web, algorithm researchers are tempted to make their codes available through a home page. This form of publication allows unlimited appendices to publications that would have length limits in conference proceedings and journals, as well as additional cross-referencing and indexing through hyperlinks and Internet search engines. A more systematic approach to Web publishing is comprehensive implementation repositories, or portals, that attempt to collect all material relevant to a particular domain. Examples in the CAD-IP domain are http://www.mrc.uidaho.edu/vlsi/, http://microsys6.engr.utk.edu:80/ece/msn/, and http://openeda.org (a portal for open-source implementation IP with goals similar to those of the Bookshelf project).

Examples in the optimization domain are Netlib, a collection of mathematical software, papers, and databases at http://www.netlib.org/index.html; the Stony Brook Algorithm Repository at http://www.cs.sunysb.edu/~algorith/; and the Decision Tree for Optimization Software at http://plato.la.asu.edu/guide.html. These repositories, which represent their domains as tree diagrams, provide reasonably good coverage. The Stony Brook Repository, perhaps the most refined of the three, identifies and encourages high-quality implementations of well-known algorithms. Far smaller than Netlib, Stony Brook can afford to rank every entry and provide brief reviews by maintainers. Typically, standards for inclusion in these repositories are rather minimal, with no concern for interoperability and reuse (for example, codes in Netlib for a given problem may be in both C and Fortran and take inputs in different data formats). The repositories trust implementations to compute cost functions and give little attention to evaluation and comparison methodologies. Overall, implementation repositories tend to be informal and lack descriptions of implementation techniques and algorithm details. Furthermore, the content of such repositories is often detached from any specific applications.

Benchmarking sites

In addition to algorithms, the Decision Tree for Optimization Software references benchmark instances and performance data. Several other sites are entirely dedicated to benchmarking: the Combinatorial Algorithm Test Sets (CATS) at http://www.jea.acm.org/CATS/; the Collaborative Benchmarking Laboratory (CBL) at http://www.cbl.ncsu.edu/www/; and the OR-Library at http://mscmga.ms.ic.ac.uk/info.html. Benchmarking sites appear more fragmented than implementation repositories, and we are not aware of comprehensive cross-disciplinary work or surveys on this subject. Whereas CATS and OR-Library primarily aim for wide coverage, CBL specializes in particular physical design topics for VLSI CAD. CBL benchmarks fueled partitioning, placement, and routing research for many years. Brglez, the CBL site maintainer, makes recommendations on the statistical aspects of benchmarking in his publications[5] and illustrates them using benchmarks at the site. In particular, he attempts to ensure that reported results are statistically significant by validating them on mutated benchmarks—benchmarks sufficiently different from the originals to expose fragile coincidences but preserving the essential features affecting reported results.

On the other hand, several key aspects of benchmarks are addressed only by ad hoc,

episodic activity. In the partitioning community, Alpert noted that as of 1997, the VLSI CAD circuit benchmarks posted at CBL no longer reflected modern technologies, and proposed a new suite of more difficult and varied benchmarks for VLSI hypergraph partitioning, based on IBM-internal circuits.[6,7] Nevertheless, there has been no consistent effort to maintain the latest partitioning benchmarks in a single location. Another aspect of benchmarking, common data formats, has also failed to receive proper attention. Formats still in use in academia, such as the .netD partitioning format, are unnecessarily difficult to read, counterintuitive, or restrictive. For example, cells in the .netD format are numbered from 0, whereas pads are numbered from 1. Such a triviality causes common format violations and variants of standard benchmarks, most likely with differing optimal solutions. Other formats correspond to outdated use and data models.

Algorithm evaluation methodologies

Barr et al.'s pivotal technical report, which analyzes the ways a heuristic optimization method "makes a contribution," spurred attention to algorithm evaluation methodologies. The authors urged that such a heuristic be fast, accurate, robust, simple, high-impact ("solving a new or important problem faster and more accurately than other approaches"), generalizable, and/or innovative. They held that high-impact research in heuristics must necessarily address a problem that abstracts or otherwise reflects a real-world application. To this end, a research experiment should be driven by a clear statement of "the questions to be answered and the reasons that experimentation is required."[8]

Particular experimental reporting methodologies for optimization algorithms are typically agreed on by researchers and become part of the community culture. A popular reporting style uses "best-so-far curves," which plot the solution cost the algorithm is expected to achieve in a multistart regime versus the given CPU time budget.[8-10] This yields a speed-dependent ranking that depicts regions of dominance in the 2D plane (instance size and CPU time) for each heuristic compared. Our earlier article points out the need for use-model-aware comparisons against the leading edge and the inherent risks of failing to make them.[1] In other words, if you're evaluating an apple peeler, you don't want to use it on oranges; moreover, you want to compare it with the best apple peelers available.

Openness

Errors are inevitable in implementations, evaluations, and reporting. Their average cost grows with algorithm complexity and slows progress in the field. A natural solution is to leverage community resources to verify performance claims and implementations, so that every researcher does not have to replicate the evaluation effort. The two most common ways to ensure community participation are to require that any modification or extension of publicly available software be made available on the same terms and to publish source code (either approach can be effective without the other). The Free Software Foundation (http://www.fsf.org/) and the Open Source Initiative (http://www.opensource.org/) represent these approaches. Both see quality issues as being closely tied to IP issues. Rosenburg provides comprehensive references and detailed discussion.[11]

Successful projects such as the Linux operating system, the GNU (GNU's Not Unix) compiler collection, and the K Desktop Environment (KDE) window manager for Linux (http://www.kde.org) used the Open Source model. The model lets users fix bugs, add features, and submit the patches as candidate changes to the model's maintainers. One of the best-publicized studies of open software argues that it provides a robust business model.[12] All submitted changes become immediately available to everyone, but very few changes make it into official releases. Thus, the Open Source is a hybrid of refereed and unrefereed electronic publication. All submissions immediately appear in the unrefereed section, and high-quality submissions are published in the refereed section.

The Berkeley model is an interpretation of Open Source that makes source codes available for any purpose, including commercial use. The only fee is a nominal shipping and handling charge. This model is credited for the success of several large software projects at the

University of California at Berkeley, such as BSD Unix and various VLSI CAD tools (Spice, Magic, Octtools, and MIS/SIS). For more details, see http://www.opensource.org/licenses/.

Software engineering

Engineers with no formal training in computer science can write optimization codes. They are not required to follow good coding standards or accepted software practices. This tendency is especially strong in academia, where better implementations traditionally have little perceived additional value over poor ones. Hence, any software repository should establish educational mechanisms, such as returning submissions for revision, and enforce software standards by rejecting poor implementations and confusing source code. Our caveat is that draconian requirements or unfamiliar software practices can turn off potential authors and block potentially useful development. We recommend the application of Occam's razor to all prospective rules and regulations.

Related issues include poor maintenance and unavailable development tools, again particularly in academia. Student-written codes may be virtually unusable after the student graduates or moves on to another project. Some authors cannot produce clean code because they have limited expertise or because commercial memory debuggers are unavailable. The community must allocate resources to evaluating such implementations and improving the promising ones.

New CAD-IP reuse medium

The Marco GSRC Bookshelf project has collected relevant experiences from electronic publishing, software repositories, algorithm evaluation methodologies, benchmarking, software engineering, and the Open Source and Free Software organizations. It has also proposed mechanisms for more-efficient CAD-IP reuse and research in fundamental CAD algorithms.

To date, the primary result of these efforts is our proposal of a two-pronged framework for IP reuse, including

- a new electronic publishing medium with support for implementations of fundamental VLSI CAD algorithms alongside textual works, and
- common open standards for data representations and evaluation methodologies.

The proposed framework emphasizes ease of use, resource savings stemming from reuse, and inclusive rather than exclusive mind-sets.

The Bookshelf project maintains an active presence on the Web (http://gigascale.org/bookshelf) in the form of a prototype CAD-IP reuse medium, focusing on algorithm implementations, evaluation, and related information. This medium is similar to a refereed scholarly electronic periodical or conference proceedings but also has software repository features. The site contains high-quality, reviewed, archival, citable publications (both text and implementations) that are relatively difficult to change once published. In its "Preview" section, authors can publish new contributions, which remain visible regardless of the editors' decision to publish or not to publish them in the refereed, or "Golden," section. In contrast to traditional scholarly publications, we place the most value on advances in implementations, as well as improvements and innovations conducive to such advances.

The Bookshelf's charter mandates the following tasks:

- collecting and disseminating leading-edge knowledge on optimization algorithms;
- providing comprehensive information about the target domain of VLSI CAD, including as many relevant subproblem types as possible;
- serving as an institutionalized community memory of past approaches, benchmarks, and results, to facilitate their continued use;
- providing an infrastructure for CAD-IP reuse, including a consistent data model and a framework that encourages a uniform appearance of contributions, in each type of CAD IP and in interfaces, documentation, and common testing and evaluation methods;
- rewarding those who publish leading-edge implementations by making it citable just as text publications are, thus compensating for any perceived loss of competitive advantage (this entails acceptance of new areas of expertise);

Bookshelf copyrights

Motivated by the success of the Berkeley model for open software release and distribution, the Marco GSRC Bookshelf supports free and open distribution of all its entries. We also strongly support the practice of freely providing data models and data interchange formats, including both the syntax and the application programming interface. Implementation releases (in library, executable, or source code form) should also be freely usable. The following is the MIT License template (also known as the X Consortium License), which we use when we distribute Bookshelf software (including source code and documentation), and which is distributed with many open-source software titles, including those shipped today with every Linux and Solaris distribution:

Copyright (c) {year} {copyright holders}

Contact author(s): {address} and/or {email} Affiliations: {company} and/or {institution}

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

The Software is provided "as is," without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages, or other liability, whether in an action of contract, tort or otherwise, arising from, out of, or in connection with the Software or the use of or other dealings in the Software.

This license allows arbitrary—including commercial—uses and does not require notifying authors of any particular types of use. The main requirement is that the license itself be copied without modification.

- lowering barriers to entry caused by implementation complexity; and
- assisting in the maturation of the domain.

Unlike traditional conferences and journals, we accept and encourage submissions that have been made public earlier, as long as there are no copyright conflicts. Items published elsewhere can be referenced in publications on our site but not stored there if they don't conform to the copyright policies explained in the "Bookshelf copyrights" sidebar.

Like traditional conferences and journals, we do not require that authors maintain their submissions after publication; rather, they may submit significant revisions as new items. This weaker type of version control simplifies quality control and reuse mechanisms. By not moving substandard publications to the "Golden" area, we encourage authors to meet published standards. Exceptions are generally made through peer review rather than by a fixed committee.

Structure

At the highest level, an exemplar of the new medium covers a domain. Our exemplar covers the domain of optimization algorithms for VLSI CAD, particularly physical design and logic synthesis. Individual areas in the domain are called *slots*. The Bookshelf includes slots for graph coloring, Boolean satisfiability, technology mapping, hypergraph partitioning, block shaping and packing, global routing, scheduling, and so on. Individual submissions are represented by entries embedded in slots. There are four main types of entries:

Problem definitions provide the means for encapsulating optimization problems and solutions. Typical entries include

- standard file formats,
- integrated I/O, including parsers of standard formats,
- standard benchmark instances, and
- reasonably good or interesting solutions.

Reference solver implementations allow developers of new solvers to make relevant comparisons and developers of larger applications to leverage existing work. These solvers should

- solve problems given in one of the standard formats or in-memory classes,
- produce results equivalent to a published, high-quality approach, and
- support modifications and performance analysis.

Independent evaluators act as reliable tools for measurement of solution quality.

Heuristic evaluation and comparison methodologies include

- descriptions of testing procedures and best-known results,

- precepts for experimental evaluation of metaheuristics, and
- references to relevant optimizers and benchmarks.

These four categories are sufficient to contain a wide variety of information types—for example, bibliographies and other links of resources, expositions, experimental studies, and statistical descriptions of real-world problems and their solution spaces. Experimental-research papers, and even theoretical papers with a clear relevance to creating better implementations, fit well in this framework.

Additionally, the Bookshelf accepts many data types that don't fit well in traditional publication media. These include generic and domain-specific standard data formats, detailed statistics describing benchmark instances and their solution spaces, and several types of executables. Common executables such as test case generators, optimizers, evaluators, and consistency and constraint checkers can be independent entries in the new medium.

Management and evolution

We distinguish two types of standards, which apply equally to submissions, but which we enforce in different ways. Individual reviewers verify the availability of all submission components (for example, on the Web), the submission's suitability for particular purposes, the submission's consistency, and the presence of documentation, usage examples, and regression tests. The steering committee ensures correct labeling, focus and utility, compatibility with common data formats, and ease of evaluation with published mechanisms. The committee also makes sure the submission complies with copyright law and fairly acknowledges contributors, prior publication, and funding sources. For each submission, the committee interprets what belongs to the domain and what does not, formally introduces new slots, solicits necessary entries to improve coverage, organizes peer review, and makes acceptance decisions.

We attempt to prevent the so-called maintenance wars that become possible with continually updated Web pages by limiting support for versioning, thus avoiding the temptation for authors to publish untested code.

We designed our medium to capitalize on existing implementations and increase their availability. Therefore, in its early stages it resembles a library (hence, the name "Bookshelf") more than a journal or proceedings. Because most software is independent, the Bookshelf currently takes multiple, nearly independent entries in each slot. This contrasts with open-source practices that foster high modularity and implementation reuse. For example, we allow duplication when it reduces dependencies between entries—that is, if B depends on A, it is easier to include a copy of A in B than to track dependencies between versions of A and B. The lack of versioning support also simplifies dependencies.

With the inclusion of large amounts of software, the question of code reuse arises. Even if an instance at first does not explicitly support reuse, policies and user interfaces should be scalable so that support for code reuse can be added in the future. For pragmatic reasons, implementations of the new medium will likely introduce features only when actually needed—and only after establishing a reliable author and user base. Another consideration depending on critical mass is the provision of both refereed and unrefereed sections of the publication medium. We believe that a refereed section driven by a steering committee must come first. While our prototype builds momentum, we are including only refereed entries, though the inclusion criteria are somewhat relaxed.

Slot contents

Slots in the Marco GSRC Bookshelf are listed at http://gigascale.org/bookshelf/Slots/. Content is available in 28 slots. Four popular examples include Placement, Single Interconnect Tree Synthesis, Clock Skew Scheduling and Clock Topology Generation, and Gate Sizing for Performance Optimization. The slot contents were created by researchers from 26 leading universities around the world, as well as system and EDA companies. The four examples just mentioned were developed at the University of California, Los Angeles (UCLA), the University of Illinois at Chicago, the University of Pittsburgh, and the University of Wisconsin.

UCLA Physical Design Tools

The Bookshelf hosts the UCLA Physical Design Tools set (http://vlsicad.cs.ucla.edu/software/PDtools), released 2 June 2000. The open-source distribution provides source codes from the Placement and Partitioning slots and contains more than 118,000 lines of C++ code in 35 packages, released under the MIT license shown in the "Bookshelf copyrights" sidebar.

Included are implementations of advanced industrial-grade VLSI physical design tools: Capo placer, MLPart partitioner, the object-oriented UCLA DB database with the Library/Design Exchange Format (LEF/DEF) parser and all supporting libraries, and regression tests. (The LEF/DEF interchange formats from Cadence Design Systems are the de facto standard for physical library and layout-design information. The source code for reader/writer and C API documentation is at http://openeda.org.)

All codes conform to the ANSI C++ standard and extensively use the Standard Template Library and other recent C++ features. The distribution is self-contained in the sense that no additional libraries are needed to compile it. Installation procedures consist of downloading one tar.gz file, uncompressing it, and running an installation script. The script adapts the packages to the user's system, runs compilation, builds all libraries and regression tests, and then runs regression tests to verify that installation was successful. Users have reported successful installations with g++2.95.2, SunPro CC5.1, and MSVC++6.0 on Intel, Sun Microsystems, IBM, and Hewlett-Packard hardware, running Linux, Solaris, or MS Windows.

The UCLA Physical Design Tools are also freely available at http://openeda.org, http://dacafe.org, and http://opencollector.org. Within the first few months of release, the toolset had been downloaded several hundred times—to most major EDA companies worldwide and to academic and industrial sites in 10 countries.

Some leading-edge implementations are freely available on the Bookshelf, accompanied by performance results on standard benchmarks. The "UCLA Physical Design Tools" sidebar describes one open-source release available on the Bookshelf.

In addition to Bookshelf maintenance, our work focuses on interoperability among slots and research into design flows based on the contents of individual slots. By the end of 2002, we plan to release executable Bookshelf extensions (Bookshelf.exe) that will automate several routine operations associated with empirical research in algorithmics (automatic processing of simulation results) and assembly of experimental VLSI design flows, such as fast prototyping with scripting languages. Bookshelf.exe will offer computational resources and easy access to Bookshelf content through several Web- and e-mail-based interfaces. Recent industry initiatives such as OpenAccess (see http://openeda.org) may facilitate interoperability among slots.

THE BOOKSHELF shares its goal of providing open-source CAD software with more recent efforts such as OpenEDA. The goal of CAD-IP reuse is by no means new; indeed, a number of companies exist because of the demand for reusable tool components such as hardware description language front ends, waveform viewers, schematic-capture editors, and format converters. Notable examples include Interra (http://www.interra.com), Engineering DataXpress (http://www.dataxpress.com), and Artwork Conversion Software (http://www.artwork.com). Nevertheless, the Bookshelf offers a unique combination of emphases: algorithmic CAD IP, advancement of leading-edge design technology, free reuse, comparison and evaluation methodologies, and common data modeling.

Through our ongoing work, we hope to build the Bookshelf's content and breadth and to make the Bookshelf a part of the research process in the design technology community. We will also build new linkages among Bookshelf slots, common data models, and industrial data interchange formats to enable vertical benchmarking and more complete evaluation of algorithm innovations. ■

Acknowledgment

This work was supported by the Marco Gigascale Silicon Research Center and by a grant from Cadence Design Systems.

■ References

1. A.E. Caldwell, A.B. Kahng, and I.L. Markov, "Hypergraph Partitioning for VLSI CAD: Methodology for Heuristic Development, Experimentation and Reporting," *Proc. ACM/IEEE Design Automation Conf.* (DAC 99), ACM Press, New York, 1999, pp. 349-354.
2. A.E. Caldwell, A.B. Kahng, and I.L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?" *Proc. ACM/IEEE Design Automa-*

tion Conf. (DAC 00), ACM Press, New York, 2000, pp. 477-482.
3. A.E. Caldwell, A.B. Kahng, and I.L. Markov, "VLSI CAD Bookshelf," 1999, http://gigascale.org/bookshelf/.
4. A.M. Odlyzko, "Tragic Loss or Good Riddance? The Impending Demise of Traditional Scholarly Journals," *Int'l J. Human-Computer Studies*, vol. 42, no. 1, Jan. 1995, pp. 71-122.
5. F. Brglez, "ACM/SIGDA Design Automation Benchmarks: Catalyst or Anathema?" *IEEE Design & Test of Computers*, vol. 10, no. 3, 1993, pp. 87-91.
6. C.J. Alpert, "The ISPD-98 Circuit Benchmark Suite," *Proc. ACM/IEEE Int'l Symp. Physical Design* (ISPD 99), ACM Press, New York, 1998, pp. 80-85; http://ww1.acm.org/pubs/contents/proceedings/dac/274535. See errata at http://vlsicad.cs.ucla.edu/~cheese/errata.html.
7. C.J. Alpert, "Partitioning Benchmarks for the VLSI CAD Community," http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html.
8. R.S. Barr et al., "Designing and Reporting on Computational Experiments with Heuristic Methods," *J. Heuristics,* vol. 1, no. 1, 1995, pp. 9-32.
9. G.R. Schreiber and O.C. Martin, "Procedure for Ranking Heuristics Applied to Graph Partitioning," *Proc. 2nd Int'l Conf. Metaheuristics*, Laboratoire Prism, Université de Versailles-Saint Quentin en Yvelines, France, 1997, pp. 1-19.
10. G.R. Schreiber and O.C. Martin, "Cut Size Statistics of Graph Bisection Heuristics," *SIAM J. Optimization*, vol. 10, no. 1, Jan. 1999, pp. 231-251.
11. D. Rosenburg, "The Open Source Software Licensing Page," 1998, http://www.stromian.com/Open_Source_Licensing.htm.
12. E. Raymond, "The Cathedral and the Bazaar," 1997, http://tuxedo.org/~esr/writings/cathedral-bazaar/.

Andrew E. Caldwell is a staff engineer at Simplex Solutions, Sunnyvale, California. His research interests include large-scale circuit partitioning, placement, and routing. Caldwell has an MS in computer science from the University of California, Los Angeles, and is on leave from the UCLA PhD program in computer science. He is a member of the IEEE and the ACM.

Andrew B. Kahng is a professor in the departments of Computer Science and Engineering and Electrical and Computer Engineering of the University of California, San Diego. He also leads the Calibrating Achievable Design activity, which includes the Bookshelf project, in the Marco Gigascale Silicon Research Center. His research interests include VLSI physical layout design and performance analysis, combinatorial and graph algorithms, and large-scale heuristic global optimization. Kahng has an AB in applied mathematics (physics) from Harvard College and an MS and a PhD, both in computer science, from the University of California, San Diego. He is a member of the IEEE and the ACM.

Igor L. Markov is an assistant professor of computer science and engineering at the University of Michigan, Ann Arbor. His research interests include combinatorial optimization with applications in IC design and quantum computing. Markov has an MA in mathematics and a PhD in computer science, both from UCLA. He is a member of the IEEE and the ACM.

■ Direct questions and comments about this article to Andrew B. Kahng, UCSD CSE and ECE Depts., La Jolla, CA 92193-0114, and to Igor L. Markov, Univ. of Michigan EECS Dept., Ann Arbor, MI 48109-2122; abk@ucsd.edu and imarkov@eecs.umich.edu.

For further information on this or any other computing topic, visit our Digital Library at http://computer.org/publications/dlib.