High-Level Synthesis: Current Status and Future Directions

Gaetano Borriello

Department of Computer Science, FR-35
University of Washington
Seattle, WA 98195

Ewald Detjens

Exemplar Logic, Inc.
Berkeley, CA 94703

Abstract

The third High-Level Synthesis Workshop was held in January of this year at Orcas Island, Washington*. What distinguished this workshop from its predecessors was that participants were asked to submit papers describing the application of their systems and algorithms to a set of benchmarks. This proved to be a successful method for comparing the extensive research being conducted in this area of CAD. In this paper, we will briefly describe the benchmarks and use them as a foundation for outlining the major themes in the research work presented at the workshop. The paper concludes with a summary of the discussions held during the course of the workshop on the future development of the high-level synthesis benchmark suite.

1. Introduction

High-level synthesis is the design and implementation of a digital circuit from a behavioral description. A behavioral description typically does not include structural information (e.g., registers, busses, etc.) but describes the function to be performed by the circuit in an algorithmic form. In essence, a high-level specification is a program describing the behavior of the circuit. Over the last decade there have been literally hundreds of papers published on methods and algorithms for high-level synthesis. Unfortunately, due to the differing cultures (both within academia and within industry) these achievements are difficult to compare.

The 1988 High-Level Synthesis Workshop was the latest in a series of gatherings to bring together CAD researchers in the area of automatic digital circuit synthesis. It was organized primarily by the authors; Ewald Detjens was Workshop Chair and Gaetano Borriello was Technical Chair. The purpose of this workshop was to begin the process required to develop a foundation that will permit comparison of high-level synthesis techniques. It was decided, by the organizers and an informal program committee, that a set of examples could begin to form the common framework needed for comparison of both systems and their component algorithms. The motivation for this effort came primarily from the frustration felt by many of the participants at the previous workshop (held in May 1986 in Santa Barbara, CA) with the difficulty in comparing the quality, applicability, practicality, and originality in the work that was presented. To address this frustration, we decided that for the 1988 workshop there would be a set of examples against which participants would relate the results of their work. The use of benchmarks for high-level synthesis systems allows determination of how "real" the reported work is - whether there is anything that can be put to immediate practical use or must be further refined and extended. Furthermore, as the benchmarks were familiar to most of those in attendance, the discussion focused on the algorithms and results rather than the details of the examples.

This paper is organized into five sections. This brief introduction is the first section. The next section describes the process by which the benchmarks were chosen and how they changed the character of the workshop. The third section is an outline of the major research directions being pursued in this area. The fourth section is a summary of the discussions held at the workshop on the future of the benchmarking effort and the problems of multiple design representations and metrics. The last section describes the facilities being put in place for evolving the benchmark suite and the objectives of the next workshop (to be held in the fall of 1989).

2. Current Status

2.1. Benchmarks

The benchmarks were selected at a meeting held at the 1987 Design Automation Conference. A group of about twenty persons from both industry and academia met and agreed to disagree on what a perfect set of benchmarks would be. However, it was decided that such an effort was important and that at least a first cut should be made in time for the 1988 workshop.

There were five criteria applied to the selection of the examples. First, there must exist a representation of the circuit that is either already machine readable or can be made so very easily. Second, it did not really matter what

*The 1988 High-Level Synthesis Workshop was sponsored by the ACM Special Interest Group on Design Automation (SIGDA) and by the IEEE Design Automation Technical Committee (DATC). It was held at the Rosario Resort Hotel in Eastsound, Orcas Island, WA on January 24-27, 1988. Special thanks are due Judi Taylor of ACM for so competently handling the many organizational details.

that representation was as long as it was well-documented. Third, the examples should not all be so large that manual translation would be an impossibility. Fourth, for each of the examples, it should be possible to obtain relevant data sheets from the designer or manufacturer. This is so participants can have access to more information than is available in the distributed descriptions, obtaining it directly from the source. Last, the examples should span a wide range of circuit classes and not focus on any particular application area (e.g., all microprocessors).

Four examples were finally chosen: a serial-line controller (Intel8251), a small microprocessor (MC6502), a digital signal processing (DSP) filter (an elliptical wave filter from a DSP textbook),[1] and a large microprocessor (MC68000). The reasons for choosing these examples were as follows. The first three designs should be small enough to run on most systems. The 68000 was used as a large example (an extreme data point) so that more robust systems could be differentiated. The Intel8251 was included as an example of a small circuit with asynchronous concurrent components. The elliptical wave filter has features that made it attractive for comparing scheduling algorithms (it was already used for this purpose)[2] and exhibiting potential pipelined organization.

Two of the examples were available in ISPS from CMU (MC6502 and MC68000). The Intel8251 was described in ISPS by J. Nestor of the Illinois Institute of Technology especially for the workshop. Finally, a one-page data-flow graph was provided for the filter example. It was felt that the filter would be trivial to translate into whatever language was required for input into a synthesis system. The MC68000 was by far the largest description, requiring 2426 lines of ISPS. The ISPS for the Intel8251 and MC6502 consisted of 533 lines and 596 lines, respectively.

A distribution package was made available to participants that included the ISPS for three examples and an English description of the filter. The hope was that various groups would translate the ISPS descriptions into their own favorite hardware description languages (HDL) and submit the new description for redistribution to other interested parties. The set of examples was distributed to over 100 sites worldwide via both electronic and surface mail and MS/DOS floppies.

2.2. Metrics

It is difficult to compare the results of high-level synthesis systems even if they are run on the same example. There are many transformations and algorithms to which the circuit description is subjected and it is difficult, if not impossible, to arrive at a single metric that can truly be used to declare one system better than another. Therefore, in the call for participation for the workshop, participants were asked to provide data on their systems for a whole range of metrics. These ranged from counts of functional units and delay along the critical path to the size of the final layout and power requirements. For comparing performance it was necessary to decide on a standard parts list for which delay numbers were available. The CMOS3 standard cell library, published by Addison-Wesley was chosen for this purpose.[3] The Caltech Intermediate Form (CIF) for this library is available through USC/ISI's MOSIS service and could be used to estimate actual circuit area by those systems that generate chip layout as their final output. The cells have been characterized with respect to timing at various temperatures.

Some participants felt that evaluating high-level synthesis on low-level constructs such as mask layout was wrong and that the appropriate measures were the classical metrics of cycle time and number of cycles per instruction, where cycle time is based on a fixed library of cells. Others strongly believed a final implementation was necessary for real comparison. We decided that a major topic for discussion at the workshop would be what constitutes a reasonable set of metrics (see section 4.5).

2.3. Use of the Benchmarks

There were a total of 18 presentations at the workshop, 6 from industry and 12 from university researchers. Of these, 11 of the presentations included discussion related to the benchmark suite. The filter example was the most common (7), followed by the Intel8251 (5) and the MC6502 (4). Only two presentations included results for the largest example, the MC68000. The popularity of the filter example is probably due to its straightforward data-flow representation. It was simple enough for everyone to digest and transcribe into a new language in an hour, with a good probability of correctness.

While many people were put off by translating the benchmarks to another HDL, others found it tedious but not impossible. In one case, D. Baldwin of the University of Rochester, translated all the examples into a new HDL he is developing, called RASP-SL. The times he gave for translation ranged from 1 hour for the filter example to 24 hours for the MC68000.

Given this order of effort for translation, everyone should have had ample time to translate the examples into their own HDL or data-flow format. It was hoped in organizing the workshop that participants would perform many translations and provide the new descriptions for redistribution. In this manner, an entire library of descriptions could be collected for each example and increase the probability that future participants would find a description that their systems could parse. We believe the limited response to translation was due to the mundane nature of the task and the difficulty in showing that a translation was performed correctly.

The reaction to the benchmark suite was as expected. Only one presentation had results for all the examples. Difficulty with the Intel8251 arose from the description being broken into three concurrent processes. Many of the systems had problems representing the synchronization and communication between the three pieces. The

MC68000 was simply too large for many systems and was too laborious to translate from the ISPS. It was clear from the discussions that few of the systems are ready for a design of that size. By far the most complete results were reported on the filter and MC6502 examples. Both of these examples have straightforward data-paths and the size of the descriptions was small enough to be translated and massaged before input to the synthesis system. In fact, the filter was criticized for not being a true high-level description for its domain. Filters are described by their frequency response, not by the data-path or algorithm that will be used in the implementation.

3. Workshop Themes

For the person who desires some background information on high-level synthesis, there are several good survey and introductory readings. We do not attempt to provide an extensive bibliography in this paper. The article by Parker [4] gives a helpful introduction to high-level synthesis. It is a useful starting point for the novice in this area. For a more advanced description (especially of the CMU work) the chapter by Thomas in Goto [5] is very good. Shiva [6] presents an overview of synthesis and surveys some high-level synthesis systems, but his article does not cover the latest research.

The various synthesis systems and research that were described at the workshop brought up a number of common themes. The following subsections are a summary of most of the workshop presentations, grouped into the theme that they most directly addressed.

3.1. Design Representation

The first such theme was design representation, which was a popular topic given the benchmarking effort. Research groups use many different high-level description languages including ISPS and VHDL, as well as developing new ones to meet new requirements. One such requirement is the addition of timing constraints to the behavioral specification. J. Nestor of the Illinois Institute of Technology addressed this issue by adding constraint specification to ISPS as did D. Baldwin by developing a new Lisp-based HDL called RASP-SL that supports constraint specification as well as concurrent tasks. Among the new languages was one by D. Knapp of the University of Illinois who presented a synthesis system that allowed the user to specify partial structure in the specification of the design.

C. Sequin of the University of California at Berkeley made a proposal to develop a standard internal representation for all aspects of the synthesis problem. His *B-graphs* would combine descriptions of data-flow, control, constraints and structural information into a single graph rather than the multiple graphs currently used. Along these lines, C. Papachristou of Case Western Reserve University presented a formal basis for applying transforms to control data-flow graphs (CDFG). He is trying to develop a "complete" set of transforms and optimization strategies based on them.

3.2. Logic Synthesis

Logic synthesis is much more mature than high-level synthesis. It is only natural that the results from this area would be applied to the higher levels. Three presentations addressed this issue. G. De Micheli of Stanford presented an iterative high-level synthesis system that uses feedback from low-level logic synthesis tools to get delay and area information and make better design space tradeoffs. This is accomplished by using fast versions of the logic synthesis algorithms to provide estimates. How good these estimates need to be has not yet been determined. He plans to incorporate user constraints into the same framework.

M. Lightner from AT&T described the applicability of low-level synthesis as a back-end to high-level synthesis and emphasized the need for the explicit specification of "don't care" information to the description as the key to better optimization. R. Camposano of IBM described the synthesis of the IBM801 using the Yorktown Silicon Compiler. The twist here was that the control logic and data-path are combined and logic optimization is performed on the entire circuit.

3.2.1. User Control

The need for improved user control of the synthesis process has long been recognized. T. Blank of Stanford talked about ILSP, a synthesis system emphasizing the interactive manipulation of a CDFG. A designer will be able to view the results of changing the CDFG on the schedule of operations and the synthesized structure. D. Knapp took this a step further. His system will permit the designer to specify a partial structure and use the synthesis system to complete the design. F. Brewer of the University of California at Irvine described CHIPPE, a system that uses a "knobs and dials" approach to exploring design tradeoffs and specifying constraints. The user limits the design space and a feasible design is generated that fits within the constraints.

D. Gajski, also from UC Irvine, suggested that we sometimes spend too much time on parts of the design process that designers do not want automated. He cited the case of R. Camposano having a difficult time convincing IBM engineers to use his scheduler. C-J. Tseng of AT&T described how the Bridge system uses an interactive scheduler to give the user limited control over this part of the synthesis process.

3.3. Systems

The systems described at the workshop either fell into general purpose systems, or exhibited some special purpose nature. There are a number of existing general purpose systems. T. Kowalski of AT&T talked about the real world problems of shoving the benchmarks through BUD and DAA. These general purpose systems took the benchmarks all the way to silicon layout. E. Dirkes

presented a method of clustering operations in the input description to improve allocation; this work is part of the "System Architect's Workbench" at Carnegie-Mellon University. W. Rosenstiel from the University of Karlsruhe talked about running the benchmarks on the CADDY system. One of the original high-level synthesis systems, MIMOLA,[7] is still in use. P. Harper of Honeywell talked about how they were taking VHDL as a front end language and converting into the MIMOLA language.

A few talks emphasized special purpose systems. The success of high-level synthesis in signal processing, as demonstrated by G. Goossens talk on the IMEC work, brought out the suggestion for designing more specialized compilers. Pipeline synthesis work being done in the Maha system at the University of Southern California was described by R. Jain. He talked about the effect of pipeline latency on area/time tradeoffs. J. Rajan of CMU described SUGAR, a system using delayed binding. SUGAR is specifically aimed at CPU design.

4. Workshop Conclusions

4.1. Benchmarks

Everyone agreed that the benchmark suite was not the best possible, and that it should be extended in several ways. There were two important points made. The first was that the current set of designs was not representative of the types of designs to which these tools will be applied, and the second was that they are not highly enough specified in a particular domain for specialized tools to be highlighted.

The presentations by W. Wolf of AT&T Bell Laboratories and T. Fuhrman of General Motors addressed the first point. The data from these two industries indicated that most designs are not pushing at the edges of the speed/power curve and are not limited by technology. Rather, most designs are communicating finite state machines with glue logic to interface them to the rest of the system. The designs were predominantly random logic chips, followed by protocol engines. There were very few processor or signal processing chips. In fact, ALU's were hardly ever seen on the more than 250 designs from both AT&T and GM designed over the last two years. The dominant recognizable component blocks were RAMs, ROMs, and buffers. Approximately 80% of the GM chips had some analog circuitry on them.

It was clear from these discussions that industry is most interested in using high-level synthesis tools to rapidly prototype non-aggressive designs to drive down the time needed to bring a new product to market. Unfortunately, the major thrust in the synthesis community has been on automating data-path and processor design rather than the control and interface circuitry that industry builds. The benchmarks reflected this bias.

The second point, brought up by the filter example, is that there are many interesting subdomains for which specialized synthesis tools can offer a substantial benefit. One example of this is the signal processing chip synthesis tools developed at IMEC in Leuven, Belgium. G. Goossens demonstrated that when the domain is restricted, complete tools can be developed that generate designs close to or better than what human designers can accomplish. Furthermore, with the domain restricted, the descriptions can be at even higher levels of abstraction. For example, a frequency response and sampling rate is all that is required for a digital filter specification.

The result of these two observations on the development of the benchmark suite was to suggest that there be two levels of examples, "compulsory" and "freestyle" benchmarks. The "compulsory" examples would be small designs, representative of the full range of applications to which high-level synthesis tools will be applied. "Freestyle" examples are those specific to an application domain (e.g., signal processing).

4.2. Design Representation

It was already known that a group of more than two people would not be able to agree on a hardware description language: witness the discussions that still rage over the future of VHDL. In fact, it was recognized from the beginning that agreement on a single HDL could never be reached. What was not apparent was that it would be equally difficult to get an agreement on a standard CDFG representation. After all, the logic synthesis community had little problem settling on a low-level description language (LIF) for distributing the MCNC benchmark suite. Since distributing benchmarks in multiple languages seemed to pose many problems (e.g., how would translation be performed and verified for equivalence), we attempted to find a CDFG representation that could become a standard for the benchmarks.

There was much disagreement over any proposed standard CDFG format. In fact, the presentation by C. Sequin on this issue caused the most animated discussion of the workshop. An obvious suggestion was made to standardize on the Value Trace (VT) representation from CMU. The complaints about the VT were that it was difficult to understand its nuances and the code used to manipulate it. Therefore, most people felt more comfortable and secure developing and using their own format and code for CDFGs rather than what may be a fixed, difficult to extend "standard". An important objection to distribution of CDFGs is that they are too difficult for humans to parse. A high-level language provides more insight for the evaluation and debugging of a synthesis system that is not yet production quality.

Other problems with standardizing on a single CDFG were that the CDFG chosen would reflect a certain style of representation. Each new CDFG that is developed has different pieces of information in it and is interconnected with the hardware structural description in a slightly different way. Furthermore, CDFGs are often annotated

with information that cannot be described in all HDLs (e.g., partial structure) and the format for this information is difficult to standardize. In general, people felt that compiler generators make CDFG generation from a high-level language trivial, so why not distribute benchmarks in some HDL rather than a CDFG. Some problems with CDFG and HDL representations that were not adequately addressed at the workshop include: whether control and data flow should be separate or combined graphs, how concurrent processes are to specified, and how to specify interface timing and electrical requirements.

It was agreed that we should continue collecting examples of well-documented designs so that people could obtain information that may not be represented in the distributed description. This was one of the reasons for the current benchmark set - all the examples are commercial parts for which data sheets are available from the manufacturer. People found these English language datasheets helpful, but were not willing to accept only data sheets as benchmark descriptions. It was finally decided that for the short-term future it would be necessary to distribute examples in multiple HDLs, along with data sheets where possible.

4.3. Testing and Verification

One conspicuously absent portion of the benchmark suite was the corresponding test vectors for each example to empirically verify the synthesized designs. Some bugs were found in the descriptions after they were distributed. Also, none of the presenters attempted to claim that the circuit synthesized by their system was actually correct. The most ambitious work along these lines was performed by T. Kowalski who compiled a C language version of the "hello world" program and tried unsuccessfully to simulate the resulting instructions on the 68000 that he synthesized. Unfortunately, it is difficult to determine whether this was due to a bug in the synthesis system or in the ISPS description. This vividly illustrates the need for test vectors, both before and after the synthesis process.

4.4. Design Constraints

There were no real-world design constraints included with the examples. All real designs have some strict timing, area, and power constraints, which restrict the space of possible implementation that a synthesis system can generate. As synthesis algorithms are becoming more general and robust, this area is now beginning to attract attention. However, it is still premature to expect a concensus on how constraints can be included in an HDL or CDFG description.

J. Nestor of the Illinois Institute of Technology demonstrated how timing constraints could be added to ISPS by using the Intel8251, the only example for which the constraints were defined (in the data sheet). Other presenters including C. Sequin of the University of California at Berkeley and D. Baldwin showed how timing constraints may be represented in a CDFG and HDL, respectively.

4.5. Metrics

Not all synthesis algorithms create layout as output, and even if a synthesis system does create layout, it is often desirable to look at measures at various levels of the design process. Synthesis consists of many separate processes, to which there are multiple approaches. We would like to compare the efficacy of the subtasks as well as the ultimate result of the entire synthesis process.

The metrics, then, are best defined separately for the different levels of description. In layout, it makes sense to speak of area, speed and power (for a particular process). At a logic level, that has not been placed and routed, we can talk about the number of latches, transistors, gates and their estimated delay. At a register transfer level (RTL) we can measure the number of registers, muxes, alus, control steps, the size of the control word and the estimated cycle time. Application specific levels of description can refer to Mips (computers and microprocessors) or signal-to-noise ratio (signal processing).

A cell library is necessary to discuss metrics at the more detailed levels of description. Any library needs to fulfill two needs: be readily available to the research and academic communities, and have a large selection of cells that have been fully characterized for speed. We had originally suggested using the area and speed numbers from the CMOS3 library, but many participants thought a better library would be one that is commercially available in a complete design system. People who do not have their own layout systems could then use the commercial chip assembly tools to produce a layout.

Another measure of a synthesis system is its throughput - how long it takes to run the examples. While most of the systems presented at the workshop ran in a short amount of time (on the order of minutes on a large minicomputer or mid-sized mainframe), this measure needs to be included in any complete discussion. Also, the total elapsed time is a useful measure for determining the level of usability and completeness of the system.

Finally, a synthesis system is not really useful if it cannot explore area/speed tradeoffs. We cannot compare a single run of one system with a single run of another system. We need to compare area/speed curves, not just single data points.

5. Future Directions

5.1. The Benchmark Clearinghouse

The idea of an electronic clearinghouse for sharing the current set and future collections of design examples was proposed at the workshop. This would allow researchers to obtain benchmarks via electronic mail and permit frequent updates of the benchmark set. This last capability is important since no general consensus was formed as to exactly what the benchmarks should be. This structure sets up a repository for design examples and results which can be accessed by everyone with a network connection.

The clearinghouse has been set up and is being maintained by Robert Mayo at the University of Wisconsin at Madison. For information on the clearinghouse and a list of currently available benchmarks, send electronic mail, via InterNet or UseNet, to:

HLSW@cs.wisc.edu or *ucbvax!uwvax!HLSW.*

If your return mail path is complex, include a line like the following in the body of your message:

"Net-Address: <your return path>".

Over the next year we hope to use the clearinghouse to facilitate the distribution and refinement of the benchmarks. A mailing list of parties interested in the benchmarking activities and future high-level synthesis workshops is also being maintained. The electronic mail address is:

HLSW-people@cs.wisc.edu or
ucbvax!uwvax!HLSW-people.

To be added or deleted from the list, send mail to:

HLSW-request@cs.wisc.edu or
ucbvax!uwvax!HLSW-request.

5.2. Towards the Next Workshop

The next High-Level Synthesis Workshop will be held in the fall of 1989. Gaetano Borriello, of the University of Washington, is workshop chair and Raul Camposano, of IBM T.J. Watson Research Center, is the program chair and a program committee has already been formed. It is expected that presentations will focus on three points: development of high-level synthesis itself, exploratory work at higher levels (system organization), and integration of high-level synthesis into design systems.

Formal approaches are needed to be able to assert correctness of high-level synthesis systems. Even though "correctness by construction" is only as good as the correctness of the software involved, the ability to prove the methods correct is needed. To synthesize real systems, more attention must be paid to design constraints (e.g., area, speed, cycle time, power, clocking schemes, interfaces).

Synthesizing an appropriate system organization is still a largely unsolved problem. System organization is the early partitioning of a circuit into sequential and concurrent modules. In the extreme, this requires automatic algorithm transformation. An example is the automatic synthesis of processor pipelines, which involves complex control, replication of processor state information, "bypassing" busses, etc. Its evaluation requires methods specific to the application (e.g., instruction traces for a pipeline). Furthermore, IC designs often cannot meet the packaging and power constraints imposed on a single chip or a board and must be physically partitioned. No system currently supports this adequately.

Incorporating high-level synthesis tools into design systems is a key issue. The design process must be studied to identify the appropriate point at which these tools can be applied. Their interaction with lower level design and synthesis tools must also be considered if high-quality designs are to be produced. Logic synthesis and layout considerations can have an enormous impact on the organization of a design. To this end, the benchmark suite is being developed to demonstrate the application of high-level synthesis systems to real circuits and their integration into the design process.

References

1. S. Y. Kung, H. J. Whitehouse, and T. Kailath, *VLSI and Modern Signal Processing,* Prentice-Hall (1985).
2. P. G. Paulin and J. P. Knight, "Force-directed Scheduling in Automatic Data Path Synthesis," *Proceedings of the 24th Design Automation Conference*, pp. 195-202 (July 1987).
3. D. V. Heinbuch, *CMOS3 Cell Library,* Addison Wesley (1988).
4. A. C. Parker, "Automated Synthesis of Digital Systems," *IEEE Design and Test,* pp. 75-81 (November 1984).
5. S. Goto, *Design Methodologies,* Elsevier North Holland (1986).
6. S. G. Shiva, "Automatic Hardware Synthesis," *Proceedings of the IEEE* 71 pp. 76-87 (January 1983).
7. P. Marwedel, "The Mimola Design System: Tools for the Design of Digital Processors," *21st Design Automation Conference Proceedings*, pp. 587-593 (1984).