Development and Use of Electronic Schematic Capture in the Specification and Simulation of a Structured-Custom ASIC

ASIC designers must sometimes provide the ASIC vendor with documentation describing the data path of the chip and its relationship to the control portion. This paper describes a method and attendant tools that facilitate the employment of commonly available electronic schematic capture software to ensure that the documentation given to the ASIC vendor always matches the Verilog HDL descriptions used by the ASIC designers for simulation.

by David A. Burgoon

This paper briefly recounts the development and use of schematic capture in the design of a structured-custom ASIC (application-specific integrated circuit). The ASIC was developed through a partnership of two teams of engineers, one from our laboratory and one from the ASIC vendor, which in this case was the HP Integrated Circuits Business Division. The team from our laboratory was responsible for defining the overall architecture of the chip, that is, the overall hierarchical block diagram consisting of data path and control, and the functional description of each module. The vendor team's main contribution was custom design of the data path and implementation of the ASIC's infrastructure, such as pads, scan chain, JTAG† circuitry, clocking network, and the like. Our team was responsible for conveying the logical architecture and functional description of the chip to the vendor team, and was also responsible for verifying the functional design through (primarily) Verilog simulation. The vendor team was responsible for turning the Verilog HDL (hardware description language) functional descriptions into an equivalent physical design.

It soon became clear that if this division of labor was going to succeed, the quality and accuracy of the documentation used to convey the chip's functional behavior to the vendor had to be ensured. Most of our designers preferred to document the overall architecture graphically, and had often done so by drawing block diagrams with their favorite graphics editor. However, to verify the functional design through simulation, Verilog HDL descriptions have to be developed. These descriptions are also used for synthesis of the chip's control logic via the Synopsys toolset (from Synopsys, Inc.). This presented an apparent dilemma: the best way to convey the high-level functional design to the vendor was through schematic graphics, but the only way to describe the functionality precisely was through textual HDL models. If our designers attempted to maintain both forms of description, we faced duplication of effort (essentially describing the block diagram twice, once graphically and once textually), and the possibility of differences between the functional schematic block diagram given to the vendor and the Verilog HDL used for simulation (see Fig. 1).

Goals and Tactics

One way to solve this dilemma would be to develop tools to compare the topology represented by the schematic documentation to that of the Verilog models, and manually reconcile any differences. Another approach would be to try to convert the Verilog models into schematic diagrams programmatically. This approach was discarded because it entailed a lack of control over the aesthetic form of the drawings. The goal we ended up pursuing was to find a way to simulate the documentation, that is, to remove the possibility of disparity between the schematic documentation and the Verilog HDL.

We chose three tactics to achieve this goal:

- Use electronic schematic capture to produce the documentation and the Verilog HDL models. Develop a set of tools and processes that ensure that the hierarchical block diagrams and the Verilog netlists are products of one and the same database.
- Design the tools and processes to enforce a policy that prevents the extracted Verilog text files from being independently altered. All changes to the Verilog models must be effected through the schematics so that disparities are rigorously eliminated.

† JTAG is the Joint Test Action Group, which developed IEEE standard 1149.1, *IEEE Test Access Port and Boundary-Scan Architecture.*

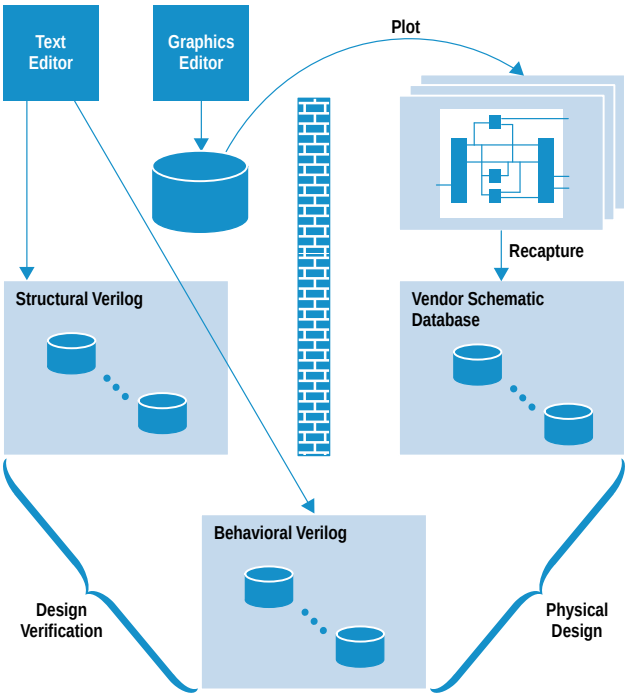

Fig. 1. The designers' dilemma: the block diagram had to be described twice, once graphically and once textually, and there was a possibility of differences between the block diagram given to the vendor and the Verilog description used for simulation. The brick wall symbolizes the lack of connection between the two forms of description.

- Design the tools and processes to be integrated smoothly with the existing simulation and regression testing environment. This allows the extracted Verilog models to be easily verified alongside the handwritten models.

Major Components of the Solution

Our goals were achieved by knitting together tools we had "lying around the house" by means of some tool enhancements, bug fixes, and shell scripts. This approach was in keeping with the grass-roots nature of the effort: we had neither the time, the funds, nor the inclination to search the commercial marketplace for a solution. We were committed to choosing the best tools from among those currently available in our laboratory and integrating them into a solution. Some pieces of the solution were internal and some were commercial. The major components of the solution are described in the following paragraphs (see Fig. 2).

Design Capture System. We chose the Design Capture System (DCS), version 5.10, from Zuken (formerly from HP) as our schematic capture tool. Some of the attributes of DCS that made it an attractive choice are:

- It is a true hierarchical schematic capture tool, not merely a graphics editor that has been coerced into capturing ASIC topology. It implements hierarchy naturally, and has a rich set of built-in features that promote circuit consistency and discourage wiring errors.
- All of our designers were proficient at using it because it had been the tool of choice for many years for the capture of printed circuit board designs.
- A Verilog extraction tool, dcs2ver, had already been written for it.

dcs2ver. DCS comes with a powerful language called DDL (Design Database Language) for accessing the database representing a schematic. Several years ago, a DDL program called dcs2ver was written by our productivity group to facilitate the functional verification of hierarchical printed circuit board schematics through Verilog simulation.[1] It was a natural choice for our application.

Briefly, dcs2ver traverses the hierarchy of the schematic and produces a Verilog module definition for each unique symbol (called a design) in the schematic. Each module definition contains the required port declarations, as well as module instantiations representing all instantiations of symbols on the circuit pages of the design. The result is a set of files, one module declaration per file, that represents the hierarchy of the ASIC as a component-oriented netlist expressed in Verilog HDL.

NgleTalkII. NgleTalkII is a name given to our laboratory's latest generation of simulation, configuration, debugging, and regression testing tools.[2,3] It is listed here because it facilitated the regular testing of dcs2ver-extracted Verilog modules against a mature suite of test scripts written in the

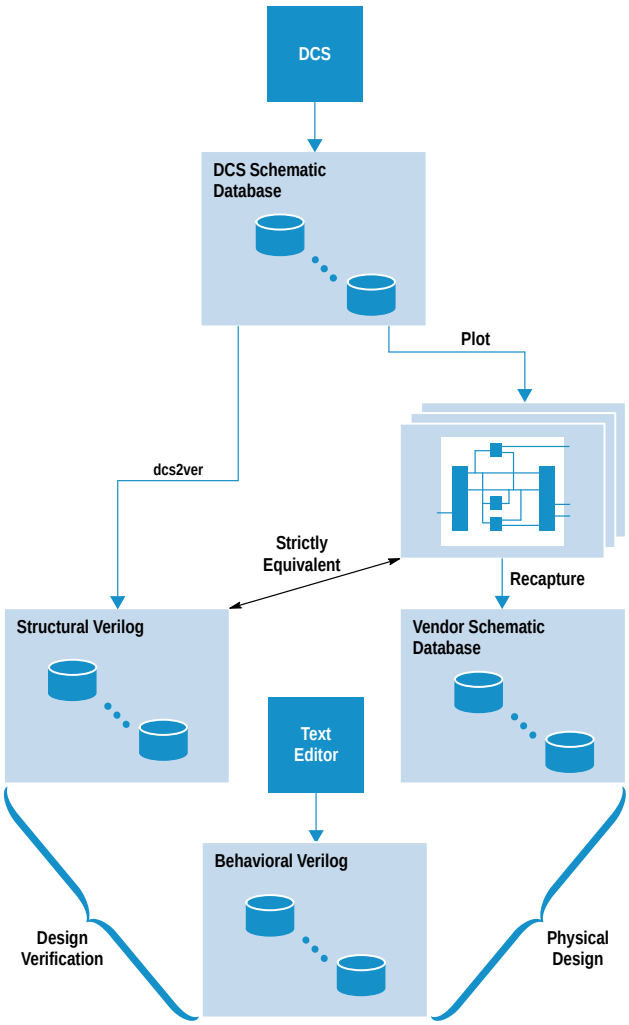

Fig. 2. The solution forces the graphical and textual descriptions to be equivalent.

NgleTalkII language. In other words, NgleTalkII made it easy to "simulate the documentation" on a daily basis.

History Management System. The History Management System (HMS), the internal predecessor of HP's SoftBench CM product, is a set of tools that manage versioned files in a networked environment.[4] It provides mutually exclusive edit access and revision control through a set of client commands reminiscent of the standard RCS (Revision Control System) commands of the UNIX® operating system.

The consistent and universal use of HMS by our design team on all design files made it easy to enforce the policy of disallowing independent modification of the DCS files and the derived Verilog files. This was achieved simply by checking in (placing under HMS control) the original DCS database files, and purposely not checking in the dcs2ver-extracted text files. We used HMS to help ensure that the Verilog embodied in the DCS schematics was extracted strictly from those schematics. Since the Verilog files were absent from the HMS server, the only way to get them (and be sure they were correct) was to play by the rules and use dcs2ver. Thus a typical design change to the high-level structural Verilog required locking the necessary DCS files, making the necessary graphical edits via DCS, checking in the modified DCS files, and running dcs2ver.

Glue and Enhancements

The major components of our solution were integrated through a set of "glue" programs and procedures and enhancements to the existing components.

dcs2ver Enhancements. As mentioned previously, dcs2ver was originally written for board simulation. Our use exposed several weaknesses and bugs, which we fixed. Some of the more notable enhancements were:

- The ability to allow DCS net aliases and splitters with alias labels.†
- The ability to use DCS net "bundles" (heterogeneously named buses),† and to control whether a given bundle is translated to a Verilog concatenation or a set of scalar port connections.
- Control over whether a module is instantiated with ports connected by position or by name.
- Detection of duplicate module names.

HMS Enhancements. As mentioned above, HMS was a key component in our solution. Unfortunately, at the time we were developing our solution, HMS did not "version" (keep old revisions of) non-ASCII files. Not willing to lose this capability for our DCS files, we wrote a set of Korn shell scripts, called rawfci, rawfco, rawfutil, and rawfhist, on top of the similarly named standard HMS client commands. These scripts allowed the DCS files to be fully versioned in a manner that is transparent to the user. This enhancement saw wide use for other types of non-ASCII files, and has since been incorporated into standard HMS.

† Net aliases are a DCS construct; they allow a net to be referred to by another name, an alias. Splitters with alias labels are a means of aliasing net names when splitting off elements of buses. Bundles are like buses, except that each element of a bundle can have a name that is not related to any other element.

make netlists. Not all users of our dcs2ver-extracted Verilog files had the training, license, or inclination to run DCS to extract the netlist files. To accommodate these users, scripts were developed whose invocation was initiated from a make(1) command. This approach was congruent with the existing NgleTalkII environment, in which users were accustomed to performing the sequence

```
fupdate; make; vsim
```

which causes new copies of out-of-date files to be fetched, C-language models to be compiled if necessary, and a Verilog simulation to be started. We added a netlists target to the makefiles which was referenced by the default target.

Briefly, the netlists target conditionally causes a shell script named extract_netfiles to be started which runs dcs2ver under the terminal form of DCS, called DDAS (Design Database Access System). If the user has a license to run DDAS, dcs2ver is executed locally; otherwise, an HP Task Broker[5] job is submitted to a DDAS server. In either case, the needed dcs2ver-extracted files are deposited on the user's machine in a few minutes.

Observations

Our approach to the use of schematic capture in the design of this ASIC was a tolerant one: our designers were free to use DCS as much or as little as desired. The typical designer did not capture a DCS design for every module in the hierarchy of the ASIC, but only went a few levels below the top level and then switched over to hand-generated textual Verilog for those modules that were predominantly behavioral as opposed to structural. One designer had a depth of zero (as measured from the top level) in the DCS hierarchy. Another used DCS only for the data path, leaving the control to a text editor, vi(1). The other two designers fully embraced the experiment, having a depth of five in some places.

Our use of DCS for high-level hierarchical capture was generally successful at meeting our goals. The most significant complaint was directed at our strict enforcement of the policy of not checking extracted netlists into HMS. The path to get extracted Verilog was somewhat complex in its implementation, especially for those who relied on the remote DDAS service. We had some occasional downtime because of licensing and HP Task Broker administration problems. In retrospect, it probably would have been more reasonable to check the extracted Verilog into HMS as nonversioned raw files, and use a cron(1m) job to do a daily make netlists to keep them up to date, thereby also ensuring that any direct textual edits were obliterated.

Another flaw in our approach was that it entailed a duplication of schematic capture effort. The vendor designers normally captured schematics in their environment from textual Verilog supplied by us. In the case of this ASIC, they found themselves capturing schematics whose top levels were essentially identical to the corresponding DCS schematics. As a result, we are already taking the next obvious evolutionary step: our designers are now directly capturing top-level schematics using the vendor's environment and are sharing files with the vendor via HMS.

Acknowledgments

Our ASIC design team consisted chiefly of Frank Bennett, Larry Mahoney, Bryan Prouty, and the author. The vendor team from the Integrated Circuits Business Division (ICBD) consisted of John Pessetto, Brian Miller, and John Morgan. The design team would like to thank our ICBD partners as well as our peers for having patience with us while we developed our tools and processes for schematic capture. We would also like to thank Tim Carlson, the original author of dcs2ver, and all those who enhanced the tool significantly, especially George Robbert.

References

1. L. Mahoney, "GTD Board and ASIC Simulation Tools: The New Generation," *Proceedings of the 1991 HP Design Technology Conference,* 1991, pp. 519-524.

2. R. Jayavant, *NgleTalkII Users Guide, Version 1.5,* internal HP document, May 27, 1993.

3. R. Jayavant, *NgleTalkII Environment Reference, Version 1.0,* internal HP document, May 27, 1993.

4. S.A. Kramer, "History Management System," *Proceedings of the Third International Workshop on Software Configuration Management (SCM3),* June 14, 1991, page 140.

5. T.P. Graf, et al, "HP Task Broker: A Tool for Distributing Computational Tasks," *Hewlett-Packard Journal,* Vol. 44, no. 4, August 1993, pp. 15-22.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open® Company Limited.

X/Open is a registered trademark and the X device is a trademark of X/Open Company Limited in the UK and other countries.