CAD Methodology for the Design of UltraSPARC™-I Microprocessor at Sun Microsystems Inc.

A. Cao, A. Adalal, J. Bauman, P. Delisle, P. Dedood, P. Donehue, M. Dell'OcaKhouja, T. Doan, M. Doreswamy,
P. Ferolito, O. Geva, D. Greenhill, S. Gopaladhine, J. Irwin, L. Lev, J. MacDonald, M. Ma, S. Mitra,
P. Patel, A. Prabhu, R. Puranik, S. Rozanski, N. Ross, P. Saggurti, S. Simovich, R. Sunder,
B. Sur, W. Vercruysse, M. Wong, P. Yip, R. Yu, J. Zhou, G.Zyner.
SPARC Technology, Sun Microsystems, Inc. 2550 Garcia Avenue, Mountain View, CA 94043

Abstract - The overall CAD methodology for the design of UltraSPARC-I microprocessor at Sun is described in this paper. Topics discussed include: CAD flow strategy, tool development and integration strategy, and design infrastructure. The importance of concurrent design style, modular CAD flow environment, incremental design verification and early design quality checking is strongly emphasized in this paper.

I. Introduction

The aggressive schedules of today's high-performance microprocessors dictate the need for a very high degree of automation during the design. This need applies to all of the design phases starting from high-level architectural design and performance evaluation, all the way down to physical design and verification.

Consequently, a wide variety of CAD tools are to be used by a large number of designers working on an unified design database. In order to support this effort, a reliable software environment and efficient CAD methodologies and tools are needed. A data management system is also required to handle large amount of data generated during the design process. In this paper, we focus our attention on the following CAD methodology issues:

1) general CAD strategy for the automation of different design tasks
2) tool development and integration
3) efficiency and reliability of the software environment
4) design information management
5) compute resource management

32nd ACM/IEEE Design Automation Conference ®

TABLE 1. UltraSPARC-I specifications

Architecture	SPARC V9 RISC
Device count	4.6 million
Die size	17.7 x 17.8
Clock freq	167 MHz
Power	30 watts
Technology	CMOS 4 metal layers
Power Supply	3.3 V

In section 2 UltraSPARC-I's design methodology is described. We present the concept of the CAD flow in section 3. An example of a CAD flow is presented in section 4. Tool integration issues are discussed in section 5. Section 6 presents CAD flow used for custom design needs. Section 7 describes the CAD tool development approach established. Section 8 analyzes design information management requirements. Network based software environment is discussed in Section 9. Section 10 concludes the paper.

II. UltraSPARC-I's Design Methodology

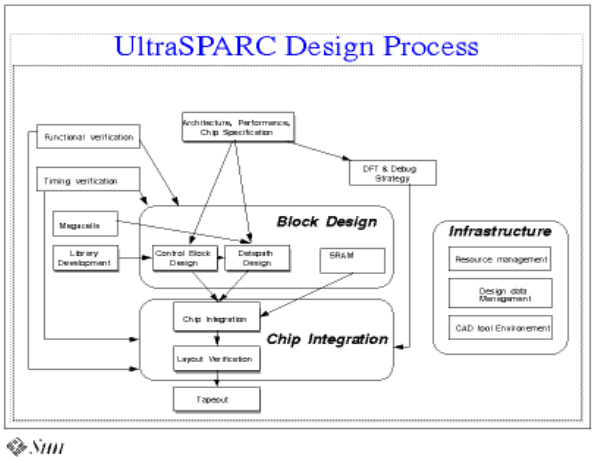

Figure 1. UltraSPARC-I's design process

Achieving the UltraSPARC-I's design specifications (refer to Table 1) in a limited time was a challenging task. The design started with architectural modeling, critical path analysis and SPECmark performance evaluation.

Once the architectural specification of the design was stabilized, the design was partitioned into structural units based on timing, functional and area constraints. For the purposes of logic synthesis, simulation, and verification, each of the structural units were coded in the Verilog hardware description language (HDL). We refer to HDL representation of a unit as functional RTL (register-transfer-level) description. The gate-level description of a unit was synthesized from its RTL description both automatically and manually depending on performance requirements. Within a unit, control and datapath logic were designed separately.

Custom blocks, i.e. megacells and memory cells in Figure 1, were designed in parallel with the RTL design. The initial physical design iterations were performed using layout bounding boxes generated from schematics. Each of the above steps involved extensive verification. After the assembly, unit layouts were extracted and timing analysis was performed for each of the units.

Concurrently with unit and block design activities, standard cell and datapath libraries were implemented using special programs for automatic generation of schematic and layout. Input to a generator program is a text file describing desired cell parameters (e.g., number of inputs, output strength etc.). These generators allowed our cell libraries to be highly tolerant of the evolving CMOS technology design rules. Additionally, the entire library was characterized and verified automatically.

After each individual unit was designed and verified, chip-level assembly was performed followed by layout extraction and full-chip timing analysis. Paths which violated timing requirements were carefully examined, troubleshooting options were considered and the redesign was performed. The entire design went through numerous iterations of the above steps until the target performance criteria were met.

To increase efficiency and reduce design time, a need arose to automate execution of different design steps associated with the same tasks. For example, parasitic extraction of a unit's layout may involve preprocessing different layouts for extraction (e.g. collecting all the necessary pieces of the unit's layout), running the extractor program, post-processing extracted netlist (e.g., for timing analysis), and verification of extracted netlist's connectivity. Often, not all of the steps associated with one task can be handled by a single CAD tool. Furthermore, different tools needed in different steps are often not available from the same tool developer (EDA vendor or in-house source).

In general, the problem we faced was providing a general CAD model capable of handling automation of an arbitrary design task, using an arbitrary combination of appropriate CAD tools.

III. CAD Flow Concept

CAD flow is a concept which allows a series of steps associated with a single task to be performed in an automated fashion. The basic premises of the CAD flow are the following:

1) A set of independent CAD tools may be used. Because they are independent, we assume that different tools do not necessarily have to conform to the same software standards (e.g. operate on the same type databases). Interaction between different tools is provided through the flow.

2) Incremental redesign is provided. If a change is made in the design, only these redesign steps which are affected by the change should be performed, not the entire design task.

3) Standard design procedures are provided through the flow, thus making the overall design process more consistent, i.e. different units are designed in the same fashion.

Similar to standardizing design procedures through a CAD flow, CAD flows were standardized themselves. Strict rules for the flow design, implementation, regression test and release were established and enforced. In this fashion, efficiency, modularity and reusability of software were maximized.

IV. An Example of a CAD Flow

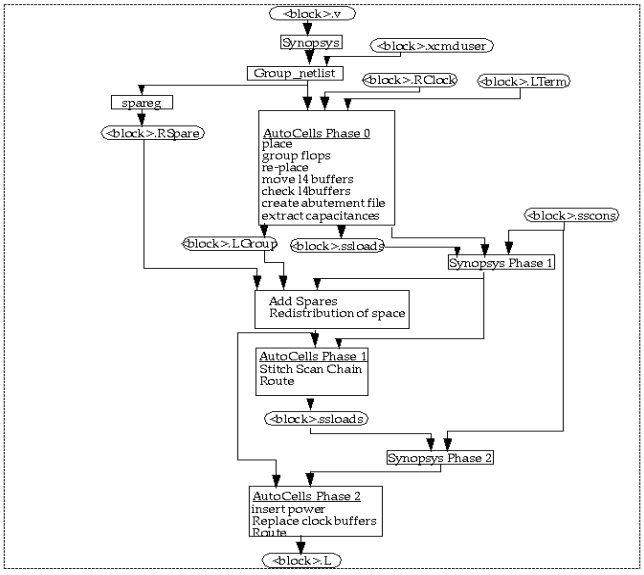

Figure 2. Control block design flow overview

We implemented all of the UltraSPARC-I's CAD flows using the Unix utility make. This appeared to suit the above requirements to enable incremental redesign, futhermore, they were designed to facilitate a wide variety of customization of the flow for specific block constraints. Tools that were used in

the flows came from different EDA vendors (Mentor Graphics, Synopsys, Cadence, Parsec Software, Meta-software, Sunrise, Chronologic, Design Acceleration, etc.) as well as from in-house developers. Further in the text, we describe one of the CAD flows in detail.

The Control Block (CB) layout flow provides a robust link between logic synthesis and cell place and route (P&R) by utilizing both EDA vendor and in-house tools (refer to Figure 2).

Starting from the RTL description of a control block, the CB flow iterates through several steps of placement and synthesis before converging to the cell final placement. The designer can interrupt the flow at any stage, hand edit intermediate results (e.g. synthesis results) and then continue automatic execution.

Besides invoking different vendor tools and providing interfaces between them, the CB flow performs a series of tasks specific to UltraSPARC-I design methodology which the vendor tools were unable to perform. For example, UltraSPARC-I specific requirements for clock tree design impose the need for a specific clock buffer placement. In this regard, the flow augmented the vendor tool capabilities to meet our design methodology requirements. Other UltraSPARC-I -specific tasks handled by the CB flow include power network design and buffer sizing.

V. CAD Tools Integration Strategy

As mentioned before, one of the main CAD flow tasks is handling the diversity of tools used across the design process. Different vendors use different netlist and data formats. Providing a smooth and continuous flow between formats and tools is a major challenge.

Whenever possible, we used industry standards (e.g. HDL and SDF) to pass data from one tool to another. Many EDA vendors also provide built-in Application Programing Interface (API) that accelerates the development of translators. However, becoming overly dependent on EDA vendor languages, formats or methodologies can make switching to new tools expensive.

At the beginning of the project we concluded that it was critical to address the data transformation problem by creating internal datamodels (a collection of data structures) and by creating a corresponding set of data readers and writers. The datamodels handle various design data (e.g netlist connectivity, behavioral descriptions and parasitic information) and the corresponding API allows easy manipulation and retrieval of data.

In cases where we were not able to develop our own solutions, we worked closely with the vendors to have them implement enhancements to the current version of the tools integrated in our CAD flow.

The UltraSPARC-I CAD design architecture changed quickly as design methodologies evolved to meet design constraints. While the cost of planning, developing and debugging the data models and the corresponding APIs was high, the datamodels enabled us to switch to new point tools and make major methodology changes with a limited amount of effort. We expect to highly leverage the datamodel software for future projects.

VI. CAD Flow for Custom Design

In addition to our highly automated CAD flows, we implemented a set of CAD point tools to enable the designers to achieve the performance goals, including:

- ERC (electrical rule check) for early detection of illegal circuit structures or questionable design practices.
- Parasitic estimator embedded into schematic netlister for critical path simulation to enable early timing optimization.
- Automatic power grid generation for 4th layer metal for faster iteration during top level assembly
- Clock net extraction to perform skew analysis in order to reduce clock design iteration.
- Custom tilers to perform automatic SRAM block assembly with faster turnaround.

These tools represented an addition to the traditional circuit design environment of schematic entry, simulation and physical layout design.

VII. CAD Development Model

The initial effort of the CAD group was to establish key engineering processes, such as internal software development, vendor interaction and design group communication.

First, the software development process was defined by initial design methodology specifications, the feasibility and trade-off analysis from the CAD standpoint, the complete development plan and the final release schedule. In order to increase developer productivity, this software development process was supported by generic software libraries (e.g netlist parsers, datamodels), code management tool, code style guidelines, and software quality metric.

Second, the vendor interaction model was defined by reviewing our design methodology with their R&D group, analyzing potential enhancements, and communicating for tool usage issues and bug tracking.

Third, the design group communication was channeled through several primary CAD flow owners responsible for developing and supporting the entire solution (e.g. timing, floorplan, library, RTL, CBF, datapath, megacell, etc.).

VIII. Design Information Management

Considering the large size of the UltraSPARC-I design team, the massive amount of information produced and the high rate of design changes, our solution to this information management challenge was to develop a Central Data Management System (CDMS).

CDMS is defined as a data management system that provides access to design files via an easy to use "check-in/ checkout" mechanism. In addition, CDMS provides a powerful and flexible release mechanism, permitting the controlled interchange of data among designers and design groups, as well as facilitating the formal product release and tapeout process. Each CDMS user has available a local copy of the files needed from the central database for design activities. All the CAD flows have been tightly interfaced to CDMS in order make this data management system as transparent as possible.

Finally, another requirement for effective design information management is to track design changes and bugs during any iteration of the design process. This is implemented by application software from Scopus Technology.

IX. Network Integrated Computing Environment for UltraSPARC-I

The computing requirements associated with the design of UltraSPARC-I called for a computing environment that would support high productivity in a cost-effective manner. This led to the design of the Network Integrated Computing Environment (NICE), where a large network of computing resources (600 computers/1000 processors, 2 terabytes of disk space) was unified to give network administrators, engineers, and managers the amount of control required to get the most out of the hardware at hand.

At the first layer of NICE, the architecture of the network is defined to support high performance, high availability, easy administration, and transparency of operation.

The second layer is focused on fault (failure detection and resolution) and performance (usage statistic/reports) management to ensure timely response to problems and provide all data necessary for proper capacity management and planning.

At the third layer, overall performance of the computing environment is improved through transparent and intelligent sharing of the network resources. DReAM, the Distributed Resources Allocation Manager, has been developed internally to provide these sharing services. DReAM is unique in the sense that it is programmable, adapting to the set of rules and policies that the organization desires to enforce for the sharing of network resources.

Finally, at the top layer, applications take advantage of the services provided by the other layers to truly harness the power of the network.

X. Conclusion

In this paper, we presented the CAD methodologies developed for UltraSPARC-I design. We emphasized the importance of automation and need for an efficient CAD software environment. Our CAD flow allows flexible and efficient automation for a specific design task. In addition, we describe how we achieved maximum usage of our compute resources to meet UltraSPARC-I productivity and design data management requirements. Finally, this methodology has proven itself by delivering first silicon that meet UltraSPARC-I functional and timing goals.

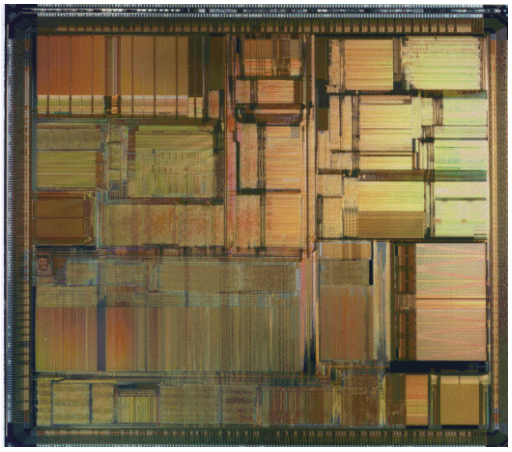

Figure 3. UltraSPARC-I microprocessor

References

[1] Larry Yang "System Design Methodology of UltraSPARC-I", 1994 (submitted for publication)

[2] Marc Tremblay "A fast and flexible performance simulator for Microarchitecture trade-off analysis on UltraSPARC-I ", 1994 (submitted for publication)

[3] Palnitkar, Saggurti "Finite State Machine Coverage program" Open Verilog International Conf. 1994

[4] J. Gateley "UltraSPARC-I Emulation", 1994 (submitted for publication)

[5] Slobodan Simovich, et al, "Timing verification Methodology for the Design of UltraSPARC-I Microprocessor at SUN", 1994 (submitted for publication)