N94-71115

Characteristics of a Semi-Custom Library Development System

M. Yancey
Product Technology Manager
R. Cannon
SCLD Section Manager
Gould AMI Semiconductors
2300 Buckskin Road Pocatello, ID 83221

Abstract– **Standard cell and gate array macro libraries are in common use with workstation CAD tools for ASIC semi-custom application and have resulted in significant improvements in the overall design efficiencies as contrasted with custom design methodologies. Similar design methodology enhancements in providing for the efficient development of the library cells is an important factor in responding to the need for continuous technology improvement. This paper presents the characteristics of a library development system that provides design flexibility and productivity enhancements for the library development engineer as he provides libraries in the state-of-the-art process technologies. An overview of Gould's library development system ("Accolade") will also be presented.**

1 Introduction

Over the past decade significant advances have been made in CAD technology to support the design and layout of ASIC circuits. Today, many systems are available to assist the design engineer with such tasks as schematic capture, logic simulation, automatic placement and routing, design rule checking, back-annotation and post-layout simulation. With the advent of such CAD tools and the standardization of libraries and design methodologies engineering design productivity has been greatly enhanced. Design cycle times have been reduced from months to weeks or days. Design engineers are now in a position to focus on innovation and value-added engineering in other aspects of their system design.

Key to many current design methodologies is the availability standard cell and gate array libraries in the latest process technologies for the various design environments. Continuous improvement in process technologies creates a need for the library development system to be flexible and efficient in incorporating the ever changing process requirements into the available cell libraries. A review of the status of tools currently available to support the library development activity plus suggested characteristics of an integrated library development system to further enhance the productivity of the library development engineer and an overview of Gould AMI's library development system known as "Accolade" are presented.

2 Library Development Tools

Various tools exist for completing the layout portion of the cell libraries. Some of these tools are computer aided drafting, symbolic layout and compaction, layout synthesis, and module generators. Computer aided drafting systems or hand layout are the simplest of the tools [3],[4]. Basically the designer draws colored polygons that are mapped to specific mask layers for use in the silicon fabrication process. This process is time consuming and is prone to design rule errors.

Symbolic layout and compaction are another set of layout tools [3],[4], [5]. The designer uses a subset of primitives (mainly transistors, wires, and contacts) to formalize the topology of the cell in a symbolic layout. The primitives may be parameterized. As an example, transistors may have parameters for width, length, and the number of contacts to the source and drain. This symbolic layout contains no information about the actual spacing or distance between objects. Once the symbolic layout is defined, the compaction tool using a set of design rules for the process technology then moves the objects as close together as possible without violating the process design rules.

Layout synthesis takes a transistor level schematic representation (present tools use a netlist) and converts it into a symbolic layout using place and route algorithms [2], [9]. The symbolic layout can then be compacted to generate mask data compatible with the design rules. This results in a significant time savings as the layout of cells is completely handled by the synthesis system. Of course this method will benefit from human interaction to control how the cell is built, especially for analog cells where capacitance and crosstalk is of great concern.

Module generators are composed of procedural code that defines the cells. The code is parameterized to handle possible variation of the cell such as the number of inputs, types of control logic, the number of memory elements, etc. Some sophisticated generators take into account delay, and area considerations. Module generators require coding experts or programmers and are difficult to debug and maintain [6]. On the other hand they work well for structures containing regular repetitive blocks such as RAMs, ROMs, and PLAs. Also this tool makes an attempt at integrating the whole design task of layout, extraction, verification and data generation.

As mentioned previously, other tools exist to aid in the design of the cells. Gate sizing algorithms are used to optimize performance and area parameters. One such algorithm is given in [7]. Mask data extraction tools are a valuable aid in the library development process. These extractors are useful for design rule checking, layout versus schematic checks, netlist extraction and the extraction of electrical parameters for use with circuit simulators [3], [4], [8].

Extracted data such as cell netlists and capacitance loading information can also be useful in the development of the library datasheets. Propagation delay values for the cell to be used for verification, logic model generation, and datasheets can be obtained using circuit simulators [3].

The circuit simulators contain models that allow the user to model the behavior of transistors and other physical devices in silicon for a given process technology. A netlist

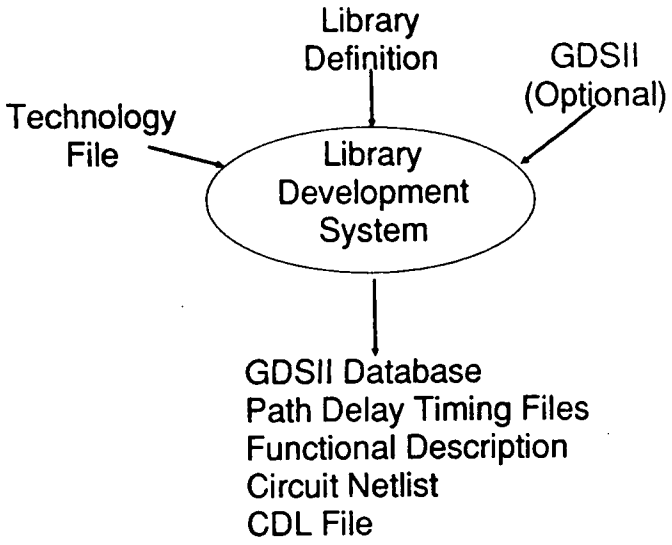

Figure 1: Library development System

of the cell including extracted parasitic capacitances and waveform stimuli can be sent to the simulator. The simulator then performs an incremental transient analysis yielding information about the changes in the node voltage and device currents as a function of time. From this data the delay from a switching input to a change in the output can be determined. Adding different load capacitances to the circuit can be used to characterize the propagation delay as a function of capacitance loading.

3 Library Development System Features

The CAD tools reviewed in the previous section have yielded significant productivity gains for the library development engineer. Further productivity gains can be realized as these tools are integrated into a single development system optimized to meet the specific needs of the library development engineer. Figure 1 illustrates the objectives of such a system. The inputs to the system would include a technology file and the library definition. Optionally, a GDSII reader would be desirable to permit data for cells that were not developed on this system to be entered into the system. The library development system using data from the technology file together with the cell library definition would then output the data required to support a library released for use with various engineering workstations. The output data to support the libraries would include the following:

1. Physical layout data in GDSII format,

2. Datasheets including cell schematics and icons, timing diagrams and cell parameter specifications, etc.,

3. Workstation path delay timing files,

4. functional description,

5. *Circuit netlist, and*

6. CDL (Circuit Description Language) file for use in Layout versus Schematic (LVS) checks.

In the following sections we will review some of the desirable characteristics and features of this library development system.

3.1 Input Features

The user interface is one of the more important characteristics of the system. This interface should be designed with the human being in mind and adapted to the human style of doing the task [1]. Therefore it is proposed that graphical data entry be used for most of the interfacing. This allows IC design engineers to focus on the engineering aspects of the library development task instead of the procedural code to implement the library cell generators, etc. Efficiencies gained through graphical data entry permit the design engineer to efficiently perform "What-If" analysis in evaluating alternative library designs.

The schematics for the circuit can be captured graphically using a mouse and screen. Various primitives can be selected such as transistors and a parameter menu can be pulled up to add values of width, length, and other values including variables related to a sizing table or expressions. This schematic now defines the data submitted to the layout synthesis tool. The schematic data can also be used for the logic comparison with the mask layout data as well as for generating cell simulator formats for use in system simulations.

The option should also be provided to allow the designer to enter symbolic layouts using a graphical entry style similar to that used by layout editors. The symbolic layout as entered does not include geometric data. Sizes can be automatically transferred to the symbolic layout from the schematic of the cell.

Icon definition is an important feature to allow for hierarchical definitions of cells. As megacell generators are developed, the capability to design blocks using a graphical interface which can automatically be converted to parameterized code is desirable.

A graphical interface should also be devised for specifying the characterization circuit netlist, input signal conditioners / generators, and the instrumentation probe points to automatically obtain the cell characterization data using a circuit simulator. Several instances of the cell (icon) can be placed with various waveform generators and loads connected to them to setup the characterization conditions.

Delay probe pairs can be placed on nodes to designate timing measurements. These probes can have parameters defining voltage level and direction (rising or falling) of the waveforms to compute the desired measurements. The probes can be assigned names so that the measurements can be identified in the characterization database. Text can also be added to set variables equal to expressions that use the delay measurements from the characterization database.

The circuit simulator netlist and control information can then be extracted from the graphically entered data. This data is then used by the circuit simulator to setup and perform the transient analysis. Results are then extracted from the circuit simulation output and specified computations are completed and returned to the characterization database.

Graphical interfaces should also be developed to provide network and icon information required for datasheets and logic models. Some of the output files may be strictly textual and may only need a textual template. All of the graphical interfaces should be menu driven. Some functions could benefit by the implementation of both menu and keyboard drivers.

Integration of these graphical front-ends with the various design engineering tools to provide for user-friendly input to the library development system can significantly improve engineering productivity.

Another important data input consideration is that the process technology inputs and the cell library inputs should be isolated to provide true process independence. A consistent set of standard library cell definitions and models is important. Once the library data has been entered for one technology subsequent libraries can be developed to support different process technologies by updating the process technology file and then using the library development system to produces the new library of cells.

3.2 Operational Features

In operation the library development system is an integration into a single package of many of the different tools which are available. For example, this means combining such tools as the following into a single design environment: gate sizing algorithms, layout, verification, simulation, and data generation and formatting tools. The preferred layout tools for library cell development are layout synthesis and compaction tools. Layout synthesis provides for the auto place and route of the symbolic layouts. Several modes may be needed to control the layout synthesis and compaction of the cells. Decisions have to be made as to whether the cell will use double or triple layer metal and which metal levels will be used for routing the signal and power ports. It is also desirable to be able to place layout constraints on the cell such as preventing certain signal lines from crossing.

Since instances will arise where the designer needs control over the placement and routing of certain cells, a symbolic layout editor needs to be provided. The compactor is useful in producing area efficient layouts for a given technology file or set of design rules. Another option for the compactor is to be able to define symbols for existing mask blocks which can be placed into the symbolic layout. These blocks will then be placed into the cell mask data without being compacted. The process independence created by using the symbolic layout reduces the amount of rework required when migrating cells to a new technology. A hierarchical compactor is required to handle the large number of objects involved in creating such megacells as RAMs, ROMs and PLAs. Another useful feature of the system is the ability to use the symbolic layout generators to develop the different memory sizes.

Tools for the verification algorithms also need to be integrated into the system. A design rule checker should be added to verify the correctness of the mask data according to the design rules contained in the technology file. A layout versus schematic checker to verify that the mask data agrees with the schematic should also be included.

Access to various simulation tools is an important feature of the library development system's design environment. For example, a circuit simulation updated to reflect layout parameters such as capacitance loading is needed to provide data for the logic models and cell datasheets. Graphical interfaces to setup the netlist and control functions as discussed previously in the input features section greatly enhances the design engineers productivity. This interface should permit the specification of waveform generators, output loading, nodes for delay measurements, voltage levels that designate valid level changes for the node, and calculations of the required delay measurements. A software routine should then translate the setup and control information and the parasitic parameter data extracted from the mask into a transistor level format readable by the circuit simulator. As the simulation is completed, a delay extractor retrieves the requested data, performs necessary calculations and stores the data in the characterization database. Other cell data such as node capacitances, area, device count, leakage currents, power consumption can be stored in the characterization database.

The system should provide for both interactive and batch mode operation. Since the tools being integrated in this system can run independently, the interactive mode is easily implemented. This mode is most useful for the development of new designs. The batch mode is used for the production of previously defined libraries in a new process technology. A restart capability at various points through the library development process is important. For example, with the restart capability an existing library can be re- characterized with new models without having to re-synthesize and compact all of the library cells. Similarly, if the datasheet templates are changed for a library, new datasheets can be provided without a new compaction or characterization. This can provide a significant time saving for many library maintenance tasks. With restart capability comes the ability to read in existing mask data and then use the library development system to characterize these cells and provide updated datasheets.

3.3 Output Features

The final tool to be added to the system is the data format generators. These generators are used to format the cell data into the various formats required to complete the formal release of the library. The mask data needs to be written out in a common format such as GDSII along with the pin locations which can be read into the place and route tool. In additions to the GDSII layout database, the circuit netlist and control data should be included as a system output file to be released with the library. This will allow library users to perform additional simulations on the cell as required by the application.

It is also necessary to update the logic simulation models. Some logic simulators accept an update file (capacitances/delays) that incorporates the delays into the simulator models. These update files can be generated directly from the characterization database.

If the external logic simulator requires a more complex update file then a template can be designed with variable designators that can be filled in from the characterization database. Other simulators require manual entry of the delay information and yet these files will be beneficial to the IC design engineer performing the data entry.

Another important set of forms is the datasheets. Again users can define a template with variables for the cell specifications which the system can then replace with symbols and values as the cell is created in a given technology. The objective is to have the library development system output the layout data, simulation models and all supporting data files in the proper format required by the user of the library.

3.4 Open System Features

The library development system needs to be user-extendible because of the many circuit and logic simulators and other external as well as internal design and layout tools that can use data from or be part of this system. The user needs access to the internal databases to be able to format data for use by other tools. Also as improvements are made in various design tools it is desirable to have an open architecture that allows for the integration of different and/or additional tools.

As these changes are made the user needs the capability to create the necessary graphical interfaces and/or textual templates to maintain the user-friendly environment. High level functions should be available to make this task easy. The integration of new tools suggests the need for a programming language capability with high level functions such as program caller and data retrieval functions. A standardized programming language such as LISP or C is preferred.

4 The Accolade System

The Accolade library development system is an integration of various commercially available CAD hardware and software products together with Gould AMI Semiconductor's proprietary software. This system realizes many of the features outlined above. A high level block diagram of the Accolade System is shown in Figure 2. Accolade does not currently support layout synthesis but it does have a one dimensional constraint graph compactor. Plans are underway to add a hierarchical one dimensional constraint graph compactor.

Data entry is accomplished using graphical entry for the schematic, icon, symbolic layout, timing schematic (the circuit simulator interface), datasheet symbols, and the datasheet schematic. A technology file defines design rules, electrical parameters, and transistor level models. A gate size file is created using algorithms to update transistors on the cell schematics by referencing variable names. The symbolic layouts are then updated from the schematics.

Accolade has schematic and mask data extractors to perform design rule checks and logic comparisons. As previously mentioned, the timing schematic defines the delay charac-

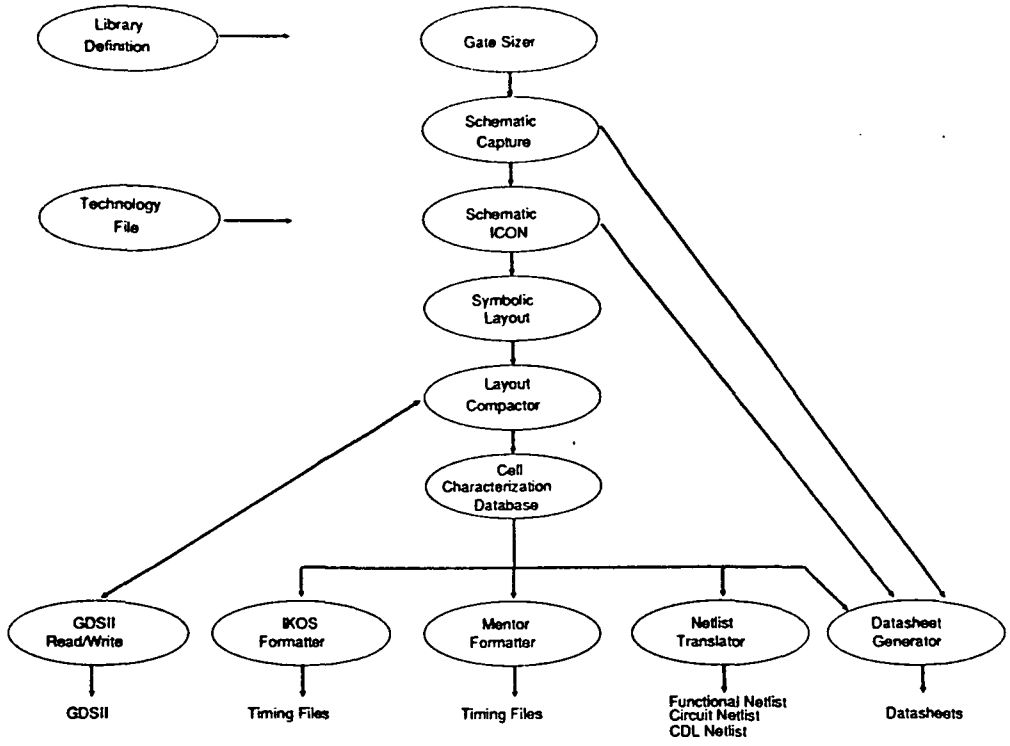

Figure 2: Accolade System Diagram

terization file for the circuit simulation, delay measurement extraction and timing calculations. Instead of using the probe pairs suggested earlier the Accolade delay measurements are described in textual form. After the simulation and delay extraction, a characterization database is created.

The schematic and mask extractors are used to develop a generic network database from which Accolade can create spice-like cell definitions as well as the functional cell definitions for use in creating a chip netlist. Accolade outputs the layout data in GDSII format. The system also can read a GDSII database and extract electrical parameters for use in characterizing the cell and providing datasheets. Logic model update files are formatted for IKOS simulator as well as Mentor's simulator with the 8.0 operating system. The Mentor file is produced with the aid of a text template.

Datasheets use a template where the datasheet schematic and data from the characterization database are merged with the template to produce the actual cell datasheet.

Restart points in the program are made by saving the mask data and the characterization database to files. The mask data can then be retrieved to re-characterize the cell if changes are made in the electrical parameters. Similarly the characterization database can be retrieved to permit a reformatting of the datasheet without the need to start at the beginning of the process.

User extensions to add interfaces, to add new data generators, or to integrate programs are implemented using the Common LISP [10] language.

5 Summary

This paper reviews various tools available to support the library design engineer and suggests desirable features of an integrated library development system. Gould AMI's Accolade library development system has incorporated many of the suggested features. Productivity gains of a factor of two have been realized using the Accolade system as compared to the previous library development system. Predominant factors in the realization of the productivity gains are as follows:

1. graphical data entry versus development of procedural code,
2. system integration from technology file inputs to GDSII and datasheet outputs, etc.,
3. process independent library definition and
4. system restart capability.

Due to Accolade's open architecture/database and the objected-oriented language implementation the systems lends itself to future upgrades.

References

[1] McFarland, Michael C., "The Social Implications of Computerization," in Proc. of the 26th Design Automation Conference, pp. 129-134, June 1989.

[2] Chen, Chao C., Chow, Shaw-Lin, "The Layout Synthesizer: An Automatic Netlist-to-Layout System," in Proc. of the 26th Design Automation Conference, pp. 232-238, June 1989.

[3] Mavor, J., Jack, M.A., and Denyer, P.B., *Introduction to MOS LSI Design,* Addison-Wesley Publishing Company, pp. 166-178, 219- 220, 1983.

[4] Preas, Brian, "Introduction to Physical Design Automation," in Physical Design Automation of VLSI Systems, edited by Brian Preas, and Michael Lorenzetti, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, pp. 1-29, 1988.

[5] Wolf, Wayne H., and Dunlop, Alfred E., "Symbolic Layout and Compaction," in Physical Design Automation of VLSI Systems, edited by Brian Preas, and Michael Lorenzetti, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, pp. 211-281, 1988.

[6] Lacroix, Didier, and Menkis, Sarah, "An Interactive Graphical Approach to Module Generator Development," in Proc. of the Custom Integrated Circuits Conference pp. 30.1.1-30.1.5, May, 1990.

[7] Richman, Bruce A., Hansen, James E., and Cameron, Kelly, "Deterministic Algorithm for Automatic CMOS Transistor Sizing," IEEE Journal of Solid-State Circuits, Vol. 23, No. 2, pp. 522 - 526, April 1988.

[8] Szymanski, Thomas G., and Van Wyk, Christopher J., "Layout Analysis and Verification," in Physical Design Automation of VLSI Systems, edited by Brain Preas, and Michael Lorenzetti, The Benjamin/Cummings Publishing company, Inc., Menlo Park, California, pp. 347-407, 1988.

[9] Ong, Chong-Leong, Li, Jeong-Tyng, and Lo, Chi-Yuan, "GENAC: An Automatic Cell Synthesis Tool," in Proc. of the 26th Design Automation Conference, pp. 239-243, June 1989.

[10] Steel, Guy L. Jr., *Common LISP The Language,* Digital Equipment Corporation, 1984.