Flexible Model for Delay and Power

Jay Abraham
Silicon Integration Initiative (Si2), Inc.
4030 West Braker Lane, suite 550
Austin TX 78759, USA
Jabraham@si2.org

Sanjay Churiwala
CADWorx Consulting Inc
504 Valley Way
Milpitas CA 95035, USA
Sanjayc@cadworx.com

Abstract

Increased demand for library accuracy and consistency across design flows for System On Chip (SOC) using deep submicron process has led to the development of an open architecture named Open Library API (OLA). OLA is a comprehensive Application Procedural Interface (API) that can be used by EDA tools for the determination of cell and interconnect timing, power and characteristics of deep submicron ICs (Integrated Circuits). OLA has been endorsed and approved by the ASIC Council[*] and other standardization bodies. OLA consists of two open and public standards; the Advanced Library Format (ALF), an OVI (Open Verilog International) standard [1], and Delay Calculation Language (DCL) and Procedural Interface (DCL-PI), an eminent IEEE standard [2].

1 Introduction

As device geometry shrinks, silicon device characteristics that were previously ignored are now becoming dominant. Though, smaller geometry have also decreased interconnect delays, the percentage of interconnect delay in a delay path has increased. In fact interconnect delay can be as much as 60%-70% of total timing in large multi-million gate designs. As a result of this, interconnect delay calculation takes a higher priority. With increase in popularity and use of hand-held device applications ranging from mobile computing to wireless communication systems, managing and controlling power also takes on an important role [3]. Besides extended battery life, low power designs have several other advantages. Low power devices often run at lower junction temperatures and this leads to higher reliability and lower cost cooling systems. This paper discusses the use of OLA as a means by which multiple application EDA tools (which may also be from different vendors) can achieve rapid convergence with the use of a single library enabling accurate and consistent calculation and modeling of delay, power, and other silicon device characteristics.

[*] A Council of ASIC companies operating under the auspices of Si2 Inc. and includes: IBM, LSI, Lucent, Motorola, NEC, Texas Instruments and VLSI Technology

2 Motivation for OLA

The motivation for OLA is that today there is no consistent standard for the modeling and/or calculation of timing and power. Formats in existence are either inadequate or proprietary. An added problem is that there are a variety of formats in use today. There is no format, which can represent IC library information for all parts of the design flow. Thus, an IC company has to generate several different library formats for the same technology family, for different tools to be used in the different parts of the design flows (see figure 1).

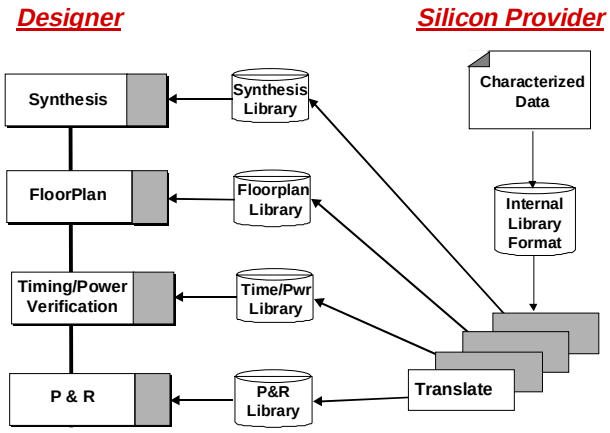

Fig 1: Library systems in use today

The section titled "Silicon Provider" describes the activities that a typical IC vendor would have to perform in order to generate the required libraries for design of an IC. This includes the characterization and generation of timing and power data and then the conversion of this data to the various library formats for supported tools. Furthermore the various library formats will have different levels of accuracy, mostly limited by the capabilities of the library format of the tool itself. This results in inconsistent timing and power results for the designer as he/she operates different tools required in the design flow (ranging from simulation, synthesis, floorplan, place and route, and final timing and power analysis). This inconsistency in different

views is one major reason for a large number of iterative loops that a designer has to go through.

3 OLA Architecture

The OLA architecture was chosen with the following requirements:

1. A unique library representation to EDA application software
2. Allowing scalability from ASIC cells to IP (Intellectual Property) blocks and cores
3. Protection of proprietary information

The choice of ALF and DCL very easily meet these requirements.

3.1 Advanced Library Format

ALF is an ASCII based file format that contains functional, timing, power, test, and gate level silicon device property information. ALF provides for the organization of library data in a clear hierarchical format form and can model gate device properties with simple constant data to complex arithmetic models including tables and equations. The hierarchical nature of ALF lends itself to model individual gates as well as complex soft and hard IP cores. Cell and core functionality is described in canonical form that is reminiscent of the Verilog hardware description language. State tables for function is also allowed. ALF is a balloted and approved OVI standard (version 1.0).

One of the most powerful features of ALF is the concept of Arithmetic Model. Any characteristic of the device can be modeled/represented with arbitrary order of complexity, using, an arbitrary deep level of nesting of combinations of equations, and/or tables.

3.2 Delay Calculation System

Delay and Power Calculation System (DPCS) can be used for computing timing and power for cells, macros and IP cores. DPCS consists of two components: a language – DCL and a procedural interface (PI). DCL is a calculation language with which one has the capabilities of modeling and calculating timing and power. The language inherently is extremely flexible and extensible. Its extensibility lends itself very easily for its use in OLA. EDA applications do not directly parse DCL source files for accessing library information. Instead a DCL compiler is used to compile the source form to a compiled executable binary module. Applications interact with the library module via the DCL PI. DPCS is a balloted and approved OVI/Si2 standard (version 1.0.2) and is expected to become an IEEE (1481) standard.

Besides having its own syntax, DCL also allows calculation routines to be written in “C”, which can interface (i.e. can be called by, or, can call) very easily with DCL routines. Thus, DCL is as powerful as “C”.

3.3 Merging of ALF and DCL

ALF and DCL are very complementary to each other. ALF lacks a procedural interface and calculation capability, but has excellent representation of IC characteristics. DCL does not provide a means by which to describe function and generic properties, but does contain a procedural interface and calculation capability. When merged, one has the benefits of the two and none of the individual disadvantages. An OLA Task Group studied the feasibility of merging these two forms, with a mission to demonstrate the effectiveness and viability of OLA with participation by leading EDA and Silicon vendors, and came up with the following recommendation:

- All libraries will use ALF and/or DCL as the primary source of representing characterized data
- There needs to exist a means for generating DCL data from ALF source data
- The IEEE defined DCL PI needs to be extended to include ALF constructs (for function, properties, arithmetic models etc.)

The framework in which the above recommendations can be implemented is shown in figure 2. The silicon vendor has the choice of modeling all of his/her IC characteristics in ALF, DCL, or a hybrid of both forms. If ALF source code exists, it is then pre-processed with the use of the ALF to DCL code generator. The code generator creates DCL tables and models, which can then be compiled with the DCL compiler. If special and proprietary calculation algorithms are to be used (e.g. for interconnect delay etc.), it will have to be entered in native DCL form, C/C++ or other linkable object form.

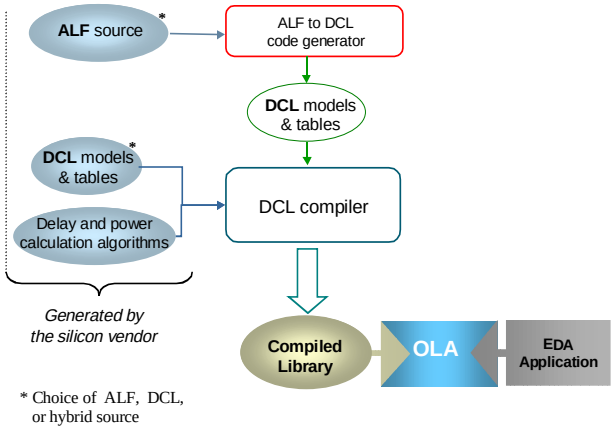

Fig 2: OLA architecture

Since DCL has the capability to interface easily with “C”, one can link his/her existing computation routines with the

DCL executable, if one so desires. This would be particularly helpful to those who have already invested a lot of effort in writing their own accurate delay computation routines.

The compiled library satisfies the original requirements of 'unique representation to applications', 'scalability from cells to cores', and 'protection of proprietary data'. The standards defined API provides unique representation. The use of ALF and DCL as hierarchical and extendable formats satisfies the scalability requirement, and the compiled form meets IP protection requirements. In addition to this, the overall goal of convergence is accomplished with the use of a single library that can be used by multi-vendor EDA tools. Each individual vendor EDA tool is presented with the same data and algorithms that will allow for convergence.

With OLA, the Silicon Provider creates a single library that contains very accurate timing and power modeling information (either in ALF or DCL, or both forms). The ALF portion of the source library is converted into DCL. This ASCII data is then compiled to produce a library known as the Compiled OLA Library. The compiled library is a binary executable module that contains function, properties, etc. as well as capability to compute delay and power. This library (binary executable module) gets dynamically loaded into the EDA application at run time. Information about function, properties, attributes, timing, and power etc. is extracted from the library by the application via a Procedural Interface (PI). This single library has all timing and power information including detailed interconnect delay calculation. The end result is a system that can compute consistent (across any tool) timing and power (see figure 3).

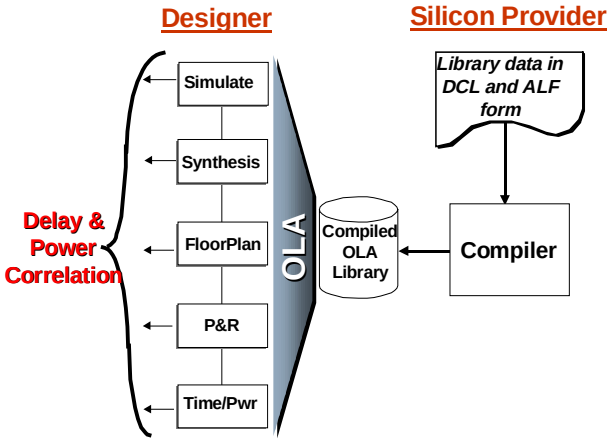

Fig 3: OLA design methodology

4 Interface Details

The existing IEEE 1481 DPCS standard does not support all of the required APIs for function and attribute data. As a result of this, extensions were needed for OLA. The Task Group identified the following areas for extensions:

- Function description
- Library, cell, pin, and path properties and attributes
- API for access to ALF arithmetic models

4.1 Extensions for function

Cell and complex function information in OLA will be available in two forms to an application. The application will have the ability to extract a string in the ALF syntax describing function in the form of boolean expressions, ALF operators or, state tables. In this case, it is up to the application to parse this function string, and interpret the functionality. The second method is a parsed structure of the function in a control or data graph format. The long-term plan is to support only the control or data graph format, so that an application will not need to parse and understand the semantics of the ALF string. Instead, it merely traverses a data structure describing function of a cell or block. Further, this method also ensures that function representation is also unique across all the applications.

The IC vendor still puts the functional description in the source ALF library in the ALF format. These functional descriptions in ALF format will get translated to the corresponding data structure representing a control/data graph, along with translation of timing/power etc. from ALF to DCL tables.

4.2 Extensions for properties and attributes

Property and attribute data is compiled into the OLA library. The application will be given a handle to a data structure containing the property/attribute data. The data structure is a linked list and follows the hierarchy given in the original source ALF. It is up to the application to understand the meanings of the properties. Property and attribute data will be available at the library, cell, pin and path (as identified by a combination of input and an output pin) levels.

4.3 Extensions for ALF arithmetic models

ALF allows for generic arithmetic models to represent/model any of the device characteristics, such as drive strengths, pin capacitance, resistance etc. (extensions are planned for signal integrity in revisions of the ALF OVI specification). Additional APIs had to be added to the DPCS PI set, in order for an application to make use of this information, which is already characterized and available in ALF library. This was accomplished with several arithmetic model specific APIs. The OLA library computes the arithmetic value for the requested model and returns a floating-point number to the calling application.

5 Library and Tool Interaction

Under the OLA architecture, the delay and/or power computation is done entirely by the library, while, the tool does solely what it is primarily supposed to do. However, the library does not have any information about the design

connectivity etc. Some of the connectivity or design related information might be needed by the library in order to compute the delay. The tool to the library provides this information, as and when requested by the library. This two-way communication between the library and the application tool takes place through a series of call-back routines, using the interface defined by IEEE 1481, and its extensions as mentioned in section 4 of this paper. Let us consider a very simple example using an inverter (only timing portion):

The application first requests the library to "model" the inverter. The library returns back all the timing paths, and constraints for this inverter. The application now requests the library to compute the delay from the input pin to the output pin. The library sees that this delay is dependent on output load. Thus, the library asks the application to send back information about output load. The application can only see the connectivity, but it does not know the input pin capacitance of the gates that are being driven by the output of this inverter. Thus, the application sees from its connectivity database that this inverter is driving an AND gate, for example. So, it again requests the library to compute and send back the information about the pin capacitance of the corresponding pin of the AND gate. The library sends back the pin capacitance for the AND gate to the application. The application then sends back this capacitance value as the output load on the inverter. The library can now compute the delay for the inverter since it now knows the load that the instance of the inverter is driving.

Thus, in effect, any computation is achieved through a serpentine sequence of callbacks, which get evaluated in a stack kind of operation. It should be understood that these callback mechanisms are conceptually different from subroutine based systems. In subroutine based systems, the value of all the parameters that are needed to execute the subroutine are already known at the time of making the call to the subroutine.

If the application or the library feels that some of the information might be needed very frequently, then, the particular information can be "cached" in memory. This will save the need for recomputing (and hence, speed execution time), when the information is needed again.

6 Advantages of using OLA

Some of the major advantages of using OLA are:

- IC companies need to generate only one library format, which can be used by all the supported tools. This also reduces a lot of overhead related to library management, and ensuring consistency between all different libraries.
- The same library is given to all the tools in the design flow. Hence, there is consistency among the timing and power data reported/seen by the different applications being used at various stages of the design. This should reduce a lot of iterations in the design-flow. Under ideal situations this has the potential to have a single-pass design methodology.
- In the conventional style, the delay computation is done by the application tool, while, the library data is generated by IC company, thus, leaving a lot of room for errors in interpretation of the library data by the tool, or, of the tool-algorithm by the library developer. But, with OLA, the computation routines are also written by the library developers themselves, thereby, leaving no room for confusion, or erroneous interpretations.
- Since the library data and the computation routines are all in compiled form, it provides protection of the IP within large macros, or cores.
- Since the library is an executable module, the provider can insert licensing requirements into portions or the entire library. For example, access could be restricted to certain cells, cores and even algorithms. Licensing can be in the form of the popular FLEXlm™ license manager or other licensing software.
- OLA provides level playing field for all EDA applications. If a company wants to develop a new tool, they need not worry about the availability of the library for the format required by them. They can immediately begin supporting OLA libraries, which would be available from most leading IC vendors. I.e. all EDA tools will have equal access to IC vendor libraries. In present conditions, each EDA tool needs to individually tuned to work with the respective IC vendor to ensure support of their library.
- OLA has the required capability to describe very accurate models for timing and power, with the capability to handle deep sub-micron device characteristics.
- OLA is scalable from cells to larger cores.

7 OLA Support

Among IC companies, some of the large vendors (IBM, LSI, Lucent, Motorola, NEC, VLSI Technology etc.) have strongly indicated that they will migrate to OLA. All of these companies have also publicly claimed that they will generate OLA compatible libraries by first quarter of 1999. Among EDA companies also, most of the large vendors (Ambit, Cadence, Mentor Graphics etc.) have agreed to support OLA and are already working on tools that can support OLA. OLA (including both its components: DCL and ALF), is a public standard. Since a particular company does not own it, future enhancements to the standard can be

proposed, and made by working with the standards committee.

8 Future Work

All said and done**,** a lot can is in need of enhancements. The ASIC Council's OLA Task Group works hand in hand with OVI-ALF and IEEE-DCL committees to resolve any potential issues seen by EDA or ASIC companies. The Task Group is working along with industry leaders to extend OLA in order to support tools for signal-integrity analysis and physical design attributes. ALF already supports most of the information needed for signal integrity analysis.

9 Conclusion

A proof of concept exhibition/demonstration was successfully presented at DAC (Design Automation Conference) 1998, using libraries from different IC vendors, and applications from different EDA vendors. OLA based commercially available tools are expected early next year from Ambit, Cadence and Mentor Graphics. In addition several large Silicon providers have indicated that they will ship OLA libraries in their design kits by first quarter of 1999. The OLA Task Group is scheduled to release a draft and version 1.0 of the OLA specification later this year. OLA is the key technology that will enable Silicon and EDA vendors to solve the convergence issues faced by so many designers today. Often times, projects are canceled because of divergence between synthesis and place & route tools. With OLA a designer need not rely on a single EDA vendor to provide a suite of tools for a design methodology with zero backward and forward annotation loops.

10 References

[1] OVI "Advanced Library Format for ASIC Cells and Blocks," *Open Verilog International*, Version 1.0, November 1997.

[2] IEEE 1481-draft specification "Delay and Power Calculation System," unpublished.

[3] "Parallel Algorithms For Power Estimation", Victor Kim & Prithviraj Banerjee, Proceedings of Design Automation Conference, 1998, Pg. 672-677

Further information on OLA can be obtained from the OLA Task Group website at http://www.si2.org/ola. Information on ALF can be obtained from the OVI website at http://www.ovi.org, and information on DCL/DPCS can be obtained from the Si2 website http://www.si2.org/dcl.