

An Incremental Technology Database Structure for Analog/Mixed-Signal Methodologies

David Kaplan (Cadence Design Systems, Inc.)
Sini Mukundan (National Semiconductor, Inc.)

Introduction

OpenAccess plays a wide range of roles in the EDA world, including interoperability enabler, communication tool, and advanced platform for software providers to develop powerful, cutting-edge applications. It is a new, powerful flexible database which gives the design community a chance to revisit its most basic design infrastructure methodologies. Probably one of the most challenging features in this respect is the increased flexibility that OpenAccess 2.2 offers in technology creation and partitioning.

There are two key aspects in which the OpenAccess 2.2 technology database contributes to increased flexibility and capability in the customer environment; The first is the basic structure of the technology file, which provides a flexible, organized method of grouping technology rules. The second is the ability to partition the technology file across several libraries. This feature, called "Incremental Technology Database" (ITDB) offers a very large degree of flexibility to users, while providing solutions to a number of challenges that designers and CAD groups may face. Both aspects of the OpenAccess technology file will be discussed here, and a sample implementation will be presented, as well as the results and challenges of designing with OpenAccess in an ITDB scheme.

In this paper, a collaborative effort involving National Semiconductor and Cadence is outlined, including a description of the ITDB model that was implemented, as well as lessons learned and challenges that were overcome. As a research project, an Incremental Technology Database implementation was created as the foundation of an interoperability flow for SOC Encounter and the Virtuoso suite of tools for IC6.1. This design environment was chosen to support the Digital-On-Top design flow that would incorporate analog/mixed signal blocks into a design that was primarily created using SOC Encounter.

The Design Environment

For the Digital-On-Top design flow, SOC 6.2 was selected for the digital design platform and was used for floorplanning, design partitioning, block design of digital sub-blocks, standard cell placement, and top-level block placement and routing. Virtuoso 6.1 was selected for analog/mixed-signal design, primarily for analog sub-blocks. Virtuoso 6.1.1 (and eventually Virtuoso 6.1.2) will be the

platform of choice for block creation, chip integration and chip finishing, including replacement of abstracts with layouts at the top level.

The inputs to the flow were no different than were provided in the past; On the digital side, inputs included Verilog, technology LEF, reference design LEF, digital timing libraries and constraints, RC files, I/O assignment files, and other normal inputs for the SOC Encounter suite. On the AMS side, the input was an analog Process Design Kit (PDK) which included an OpenAccess 2.2 library (including CDFs, simulation model files, etc.).

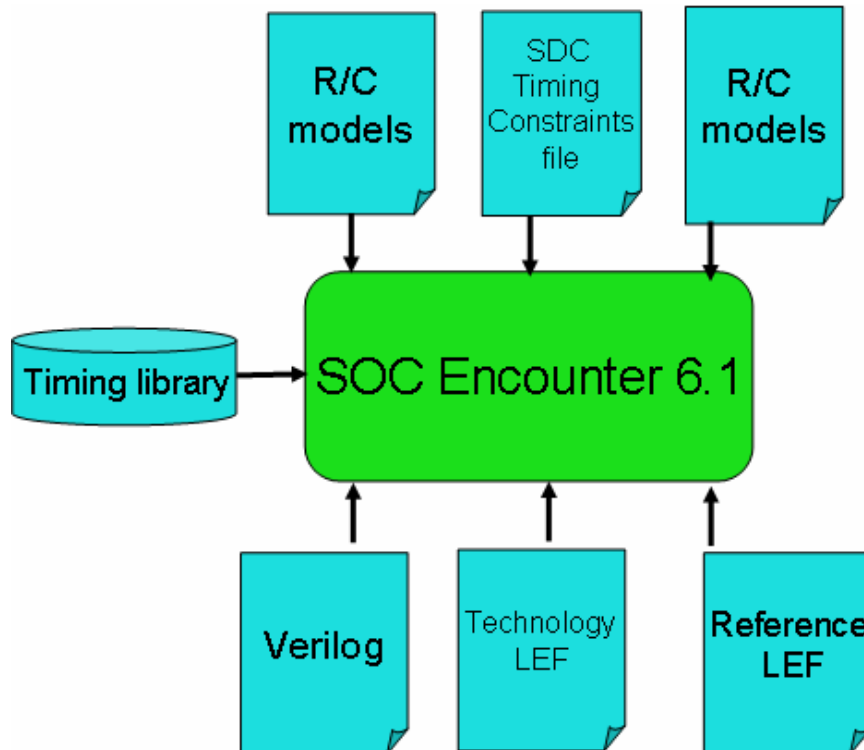


Fig. 1: SOC Encounter Inputs

The goal was to simplify the data exchange in SOC Encounter by basically replacing the Verilog, Technology LEF, and Reference LEF with OpenAccess 2.2 databases. The result was no longer reading in Verilog and LEF, but rather reading in directly an OpenAccess database. This could also include routing data which would be normally stored in DEF. This is shown in the following figure:

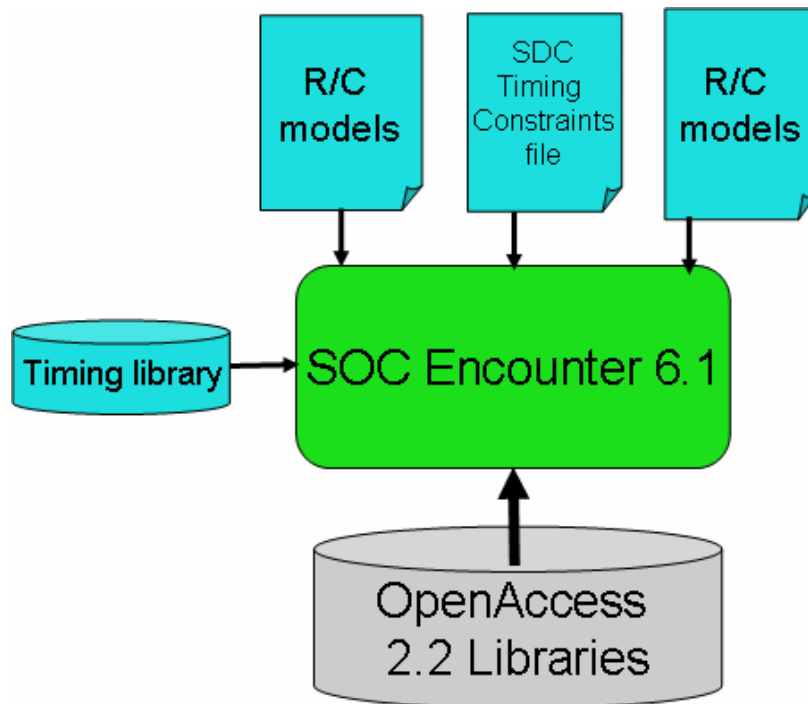


Fig.2: New Inputs to SOC Encounter

In addition to simplifying the design flow by consolidating the Verilog, technology LEF, and reference database LEF into the OpenAccess database, this data is now interoperable with other OA2.2-based applications.

An ITDB Primer

A short introduction to ITDB is provided to illuminate some of the topics in this discussion. Incremental Technology Database (ITDB) is very easily setup using a single short reference in the controls section, called refTechLibs. Here is a short excerpt of an ASCII Virtuoso technology file prior to being loaded:

```

;*****
; CONTROLS
;*****
controls(
  techParams(
    ;( parameter          value          )
    ;( -----          -----          )
  ) ;techParams

  viewTypeUnits(
    ;( viewType          userUnit          dbuperuu          )
    ;( -----          -----          -----          )
    ( maskLayout          "_def_"          1000          ) )
  ) ;viewTypeUnits

  mfgGridResolution(
    ( 0.100000 )
  ) ;mfgGridResolution

  refTechLibs(
  ; techLibName
  ; -----
  ; "anaTechLib digTechLib"
  ) ;refTechLibs

```

Fig. 3: A sample technology file with ITDB statement

Setting up ITDB is fairly simple from the user's standpoint, belying the complexity of the software applications that make use of it. In fact, all the user needs to do in order to create an ITDB database is simply load a new techfile with the above refTechLibs statement. The arguments are one or more technology library names, the technology data of which will be loaded along with any local technology data. It's as if those other technology files were copied and pasted into this techfile at run-time, although there is more involved from an organizational standpoint, and there are rules that govern which data can and cannot be duplicated.

There are several ways to load a techfile, but depending on which particular implementation you are going to use, one particular feature might offer an advantage over another. Following is the form that is accessed from IC6.1.1:

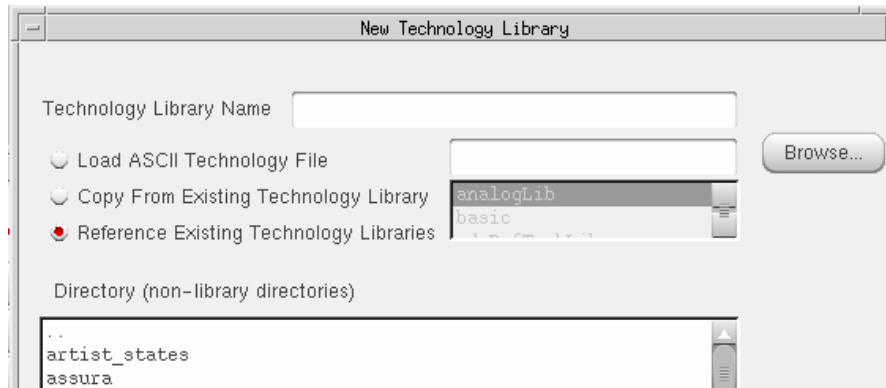


Fig. 4: New Technology Library Form in Virtuoso 6.1

Note that there are three options for creating a new technology library. The second option, “Copy from Existing Technology Library” is not important. The only two ways to add the correct refTechLibs statement are to load an ASCII techfile (which ostensibly has the refTechLibs statement in it) or the third (new in IC6.1) option: Reference Existing Technology Libraries. With this option, the user specifies one or more library names in the form, and a new technology file is created which includes a refTechLibs statement, along with a few empty sections (including layerDefs and constraintGroups). A user can then dump that technology file at a later time, and make additions or changes as necessary.

Technology Setup

The first step in any ITDB library scheme is planning. Once the desired technology structure is decided, it is a straightforward task to determine the best way to achieve that technology structure. Since the concept of ITDB is new to most users and CAD groups, a good suggestion is to use the structure proposed in this document as a jumping-off point for a particular ITDB interoperability implementation. The following goals were identified as key priorities in forming the ITDB structure:

- Consolidated design rules for most applications in the analog technology file
- Site definitions stored in a centralized location, either the main PDK or in a digital technology library, perhaps a standard cell library
- LEF technology rules (which become a special constraint group called LEFdefaultRouteSpec) stored either in the main PDK or in a digital technology library
- Basic set of vias stored in the basic PDK, in the digital technology library, or both
- Ability to store custom vias created by SOC encounter in the design library

- Minimizing the number of technology libraries to the smallest number possible

After reviewing the various technology information (as described below), it was decided that one of two potential library structures would be implemented.

Implementation #1: Four libraries

The first library structure included the following features:

- Basic analog/mixed signal technology, including layers, basic via set, foundry constraint group, and tool setup constraint groups. Library name: cmos(#)
- Digital technology from technology LEF stored in digital technology library (new). Contains LEFdefaultRouteSpec constraint group, vias, and some digital foundry constraints
- Standard cell library(s) with local siteDefs, which depend on the selection of standard cells in the library
- Design library(s) with local technology for unique vias

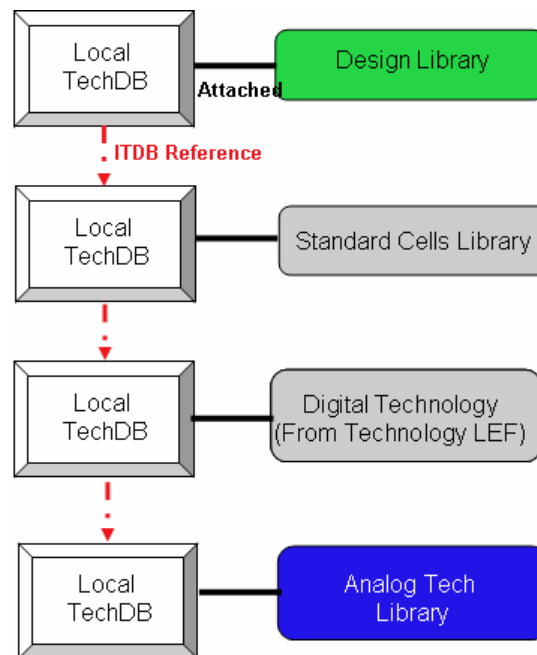


Fig 5: First ITDB Hierarchical Library Scheme

This model allows for a great deal of flexibility, because of the partitioning of data. A few key points become evident when studying this model in the actual rollout phase:

- Additional design libraries (of which there could easily be several in the eventual completed design hierarchy) would either be attached to the standard cells technology library (through the normal attachment

mechanism) or could have its own technology library (as shown above), and simply reference the standard cells library through an ITDB reference

- o Also, perhaps as many as 10+ standard cells libraries might be available for a given designer to use, which means that these libraries would have to be available to all referencing design libraries. More will be said about this in a moment.

As stated previously, one of the goals is to reduce the number of technology files that must be maintained, and as there could be many design libraries and possibly more than 10 standard cell libraries, it quickly became evident that this structure could result in a huge effort required by the CAD organization in organizing and maintaining these libraries. The following illustration shows this complication. Bear in mind that each local technology file must be either generated or hand-edited to create the ITDB references in each local techDB:

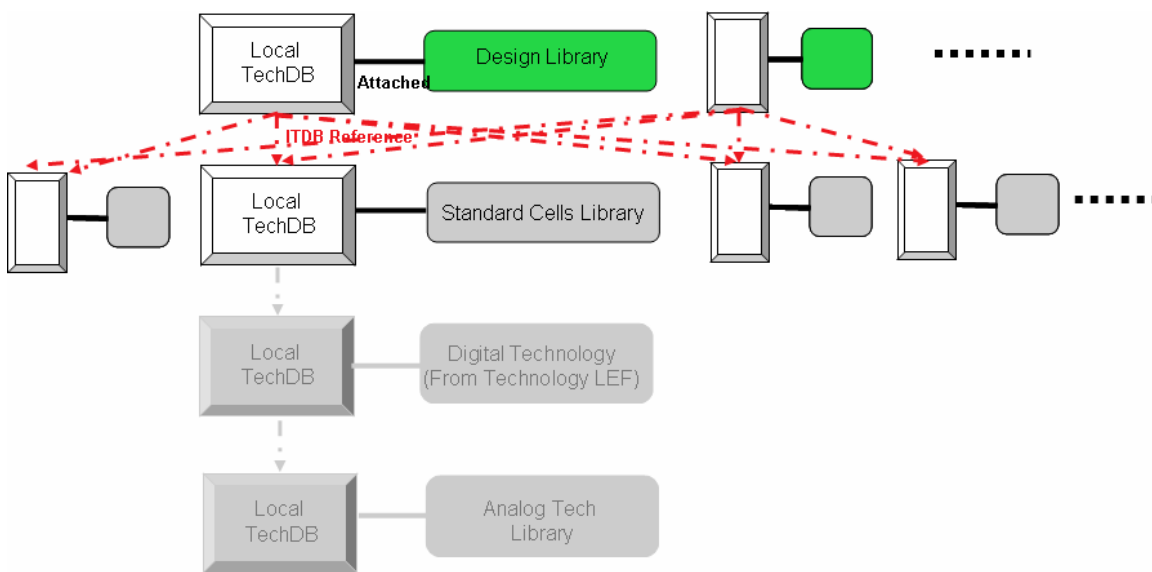


Fig. 6: Complications in the ITDB Scheme

Unfortunately, where there are m standard cell libraries, and n design libraries with local technology, there are now $n + m$ libraries to be maintained, and m^n connections to account for in the various technology libraries. Furthermore, for each additional standard cell library added, m techfiles must now be updated to contain this new reference.

Implementation #2: Three Libraries

The second-choice structure, which relieved some of the concerns that were inherent in the first implementation, was to remove the requirement of

having the standard cell libraries be part of this ITDB structure in order to reduce the workload in both setting up and maintaining a given design hierarchy; Would it be possible to move the technology information from the standard cell library techDBs into either the digital or analog technology databases? This question deals with site definitions (siteDefs), which are typically based on the size of the standard cell that will occupy that site or series of sites in a standard cell row. The underlying assumption was that the ideal location for site definitions is in the libraries where the standard cells are stored. This has the following advantages:

- Only site definitions that work with available standard cells will be available to the designer
- Less chance of erroneously using the wrong siteDefs when creating standard cell rows
- Maintenance of siteDefs is more logical as they are stored in a logical location – updates to standard cells would be more easily tracked and fixed in the corresponding siteDefs in the techDB

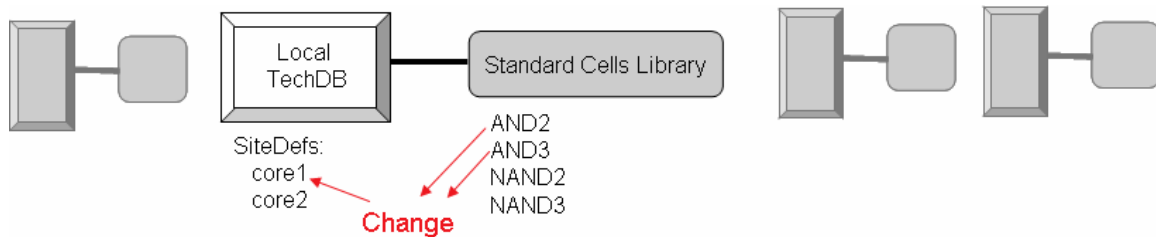


Fig. 7: Changing Standard Cell Heights

Despite these advantages, the challenges listed above (with maintaining such a potentially large number of technology files prompted the revisiting of this plan. The result was a superset of all possible site definitions being placed in the digital technology techDB, with standard cell libraries being simply referenced for standard cell instances. Following is a diagram of the resulting library hierarchy:

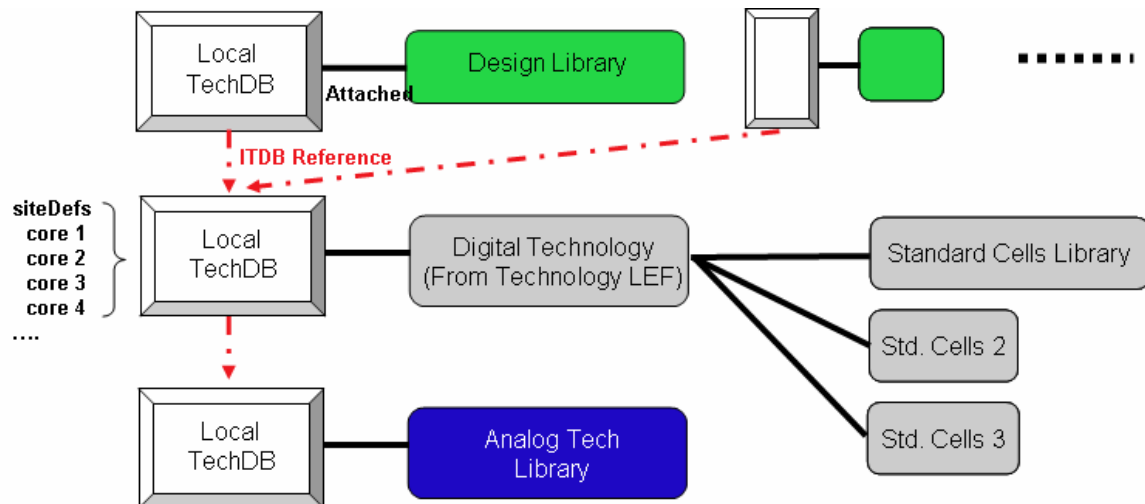


Fig. 8: A Simplified Model with Site Defs in Digital Technology Library

This was the model selected, due to simplicity and ease of maintenance, which was implemented for this digital design flow at National Semiconductor. Once the desired high-level architecture was determined, the construction of that structure was planned, with the results provided in the next section.

Technology Creation Flow

One of the challenges in constructing the technology hierarchy is that we have so many options for how to construct the final product. The key to success in this endeavor is to get all data into a format which could be used for comparison, and eventually for merging. To accomplish this, the technology LEF was read into SOC 6.1 and then saved to OA2.2, thus creating an OpenAccess 2.2 technology library. This was done out of necessity; There are currently limitations in the LEFin process that prevent this operation directly, and while they are being addressed at the time of this document, it was simple enough to save the design directly from SOC 6.1, with the same results.

In order to monitor the saving process, the library was first created using the Virtuoso 6.1 library manager, with local techDB and referencing the analog technology library. Then the design was saved into that new library from SOC 6.1. Using Virtuoso Technology File Manager to then dump an ASCII technology file, an ASCII / DFII version of the technology LEF was obtained.

The short term goal at this point was to ensure that there was minimal (or no) overlap between the analog technology techDB and the digital technology techDB. There are two reasons for this; First, in many cases (including layer names, via names, constraint group names, etc., overlap (really, collisions) are forbidden by the OA2.2 API. Second, in general it is more efficient and organized to minimize duplication and collision, so that if updates are required, they can be easily located in a single, logical place. Thus, the next step was to dump the analog technology techDB into an ASCII file, and observe the overlaps and differences.

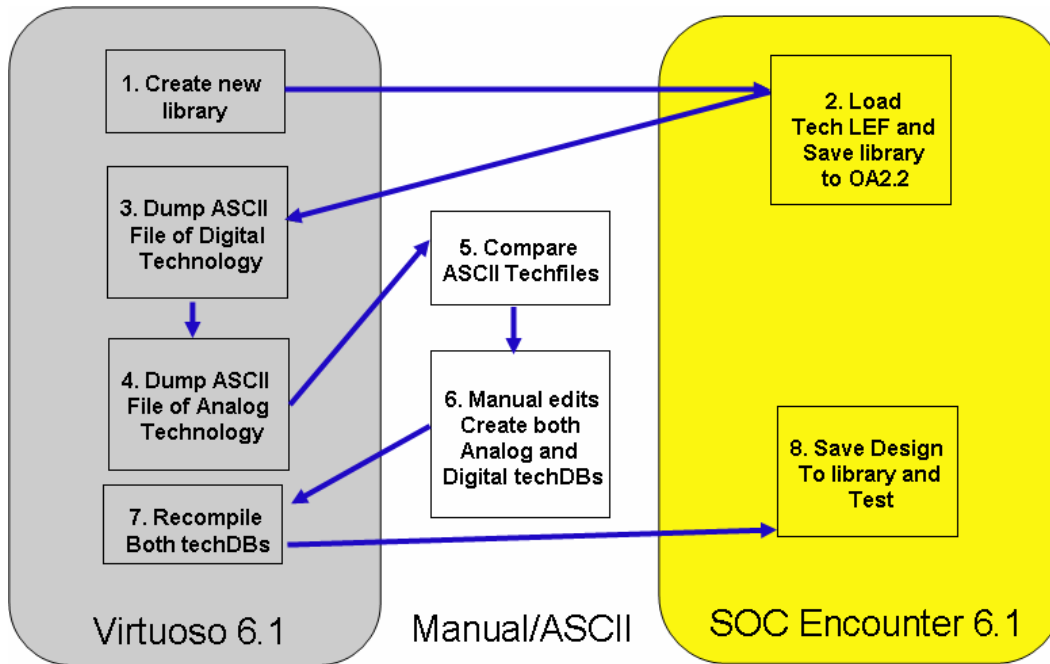


Fig. 9: Technology Creation Flow

In reality, only once the individual files were available (in steps 5 and 6 in Fig. 9 above) was it possible to start planning the actual modifications to the respective techfiles. The reason for this is that the overlap of rules and layers is not known until one observes the two technology files side by side. While it's not necessarily intuitive that there should be overlap, in fact, it's almost always the case. For example, for a given technology, the technology LEF that is passed back and forth between designers often contains a subset of the layers provided in the analog PDK. What might be surprising to some is that, often, there are one or two layers in the technology LEF (typically reserved for individual tools in the digital flow) that are not in the analog PDK.

The next section describes the separation of technology data from the original dumped analog PDK techfile and the newly created (then dumped) digital technology library. Those two files served as the source for all technology data (including siteDefs, but excluding vias that were generated during the saving of the digital design into the OA2.2 design database).

Partitioning the Technology Data

As was mentioned previously, once both the digital technology (saved from SOC Encounter) and the analog technology (from the analog PDK) were available in dumped ASCII format, a comparison was performed. This was a visual comparison, performed by Cadence and National personnel, and once the common items were identified, it was determined which sections should remain in the analog technology, which should be moved to the digital technology, and which should be moved to another location (either a standard cell library or into the design library).

One of the key decisions that was made early on is that the majority of the technology data would be stored in the analog techlib. There were several reasons why this was done:

- Typically, there are more layers required for analog/mixed signal design applications than in digital technologies
- Since the layers are defined in the analog techlib, it makes sense to keep the layer functions in the same location
- Most analog tools are customized from the analog technology file, including Virtuoso XL and other products
- The analog technology file supports all types of vias, including those used in digital applications, while LEF only supports certain types of vias

The following list describes the process that was used. Note that this was a trial-and-error process to a large extent.

1. **Remove the layerDefs section from the digital technology file.** This was done because, as was mentioned earlier, after careful examination it was determined that each and every layer in the digital technology file was present in the analog technology file as well, and thus, no layers were needed. When the techfile did not load successfully, the header declaration for the layerDefs section was put back in, and it loaded. In other words, your technology file must contain the following in order to load properly:

```
layerDefs( )
```

By including that statement, and leaving it empty, no layers will be created in the digital technology, but the techfile will load.

2. **Reconcile differences in the layerRules sections.** There were several issues in this area. The first was a layer function in the digital technology file that disagreed with the analog technology file. Specifically, a layer was defined as “metal” in the analog techfile, and “poly” in the digital. This was fixed in the appropriate file. The second issue was a metal layer that had the incorrect routing direction, and this too was fixed. Finally, the current densities in the digital technology file were much more detailed, and

contained table data for most routing layers. The analog was much simpler. The current densities were left as-is in the technology file.

3. **Comment out redundant via definitions in digital technology.** This included an entire set of standard vias (which were completely redundant and identical to those in the analog techfile) and a single custom via, whose name collided with another in the analog techfile.
4. **Reconcile the LEFdefaultRouteSpec constraint group.** When technology LEF is converted into OpenAccess technology data, both LEFin and saving to OpenAccess directly from SOC6.1 will create a special constraint group called LEFdefaultRouteSpec. Upon inspection, both the analog and digital technology files contained this constraint group, but the digital version contained much more detail, more constraints, and seemed a superset of the analog version. The single exception was the interconnect section, which in the analog techfile contained all of the layers in the valid layers section that the digital did, plus poly and a metal layer. The validVias statement contained all of the digital vias. For National, the entire LEFdefaultRouteSpec was removed from the analog techfile (commented out) and recompiled, with no negative consequences.
5. **Move the siteDefs from the digital techfile to the standard cells library (Maybe).** As was discussed earlier, the first attempt at an ITDB scheme included moving the site definitions into the library where the standard cells that utilized them existed. This was eventually reversed, and the siteDefs section was moved back into the digital techfile.
6. **Remove the viaGen statements from digital techfile.** The viaGens were duplicated in the techfiles. Those in the analog remained, those in the digital were deleted.
7. **Removed layerFunctions from the digital techfile.** After all of the previous changes were completed, the digital techfile was saved and compiled (using the Virtuoso 6.1 technology file manager). Errors appeared indicating that there were layerFunctions defined for layers which were not locally defined. The entire layerFunctions section was removed (they were already present in the analog techfile). However, the current densities were not present in the analog techfile, and were left in the digital techfile.
8. **Vias stored in the design library.** One of the primary reasons for using ITDB is to store vias created by a design-specific application in the library with the design. OpenAccess 2.2 requires that vias be stored as technology information, which for digital designers is standard operation. However, in analog design individual designers seldom have write access to the technology file. The problem is that digital design applications

(including SOC Encounter, among others) create new vias on-the-fly, necessitating a creative solution. In this scenario, ITDB is an important solution, as the vias can be stored in a separate techlib from the analog technology. After saving a test design, several new vias were saved to the local technology database in the design library.

9. **Digital technology stored in digital techfile.** The “digital” technology data includes the default routing rules, from LEF, which are stored in a special constraint group called “LEFdefaultRouteSpec”, which defines metal widths and spacings, acceptable vias and via dimensions, and other default routing rules. Non-default rules are stored in separate constraint groups, the names of which are consistent with the non-default rule name in the technology LEF.

In addition to the above steps, there were some differences that were more complicated, and while they have not been completely resolved, they are under test and appear to be working correctly. These included:

1. **Techlayer properties:** There were many techLayer properties in both the analog and digital techfiles, many of which disagreed. These were reviewed by the CAD team, and for the most part, the digital techLayer props were retained.
2. **Routing directions:** The analog techfile was changed to match the digital routing directions. However, the question arises as to why there should be any disagreement, when routing directions and generally adhered to even in analog applications. This was discussed with the CAD team, and is being reviewed.
3. **Current Densities:** Again, there were two versions of the current densities, which were not the same. The version in the analog techfile was commented out, and the digital version was moved into the analog techfile, although this is still under review by the National CAD team.
4. **Via Specs:** There were some changes required for the viaSpecs section. ViaSpecs specifies which vias are available for routing in a given constraint group, and it was noticed that vias 1 through 4 were not present, but desired in the LEFdefaultRouteSpec constraint group. These were added to the digital techfile to ensure that these layers would be usable. Without them, SOC 6.1 produced errors during read-in of a design that contained these vias. The vias were standard vias from the analog technology, not custom vias from the digital technology.
5. **Site Defs:** Again, site definitions are moving around, but for now are located in the digital technology file.

Side note: It's important to remember that what will be used by all applications, in the end, is a single, logical union of all techfiles. What this really means: If a certain item appears only once in the union (i.e., it's not located in more than one techfile), then where you put it is more a matter of bookkeeping than it is a matter of proper function. No matter where you put it, it will correctly.

Lessons Learned

A number of lessons were learned through the process of constructing this ITDB hierarchy. Here just some of the lessons from which others who are constructing a similar hierarchy will benefit:

- Certain sections must be included in a techfile, even if they are completely empty. These include: controls, layer definitions (layerDefs), and constraintGroups (must include foundry, which can be empty).
- Technology objects with unique names or numbers cannot be duplicated. Examples include constraint groups, vias, and layers. If a layer, via, or constraint group is defined in more than one technology, the techfile will not load successfully. The only exception to this is the foundry constraint group, which is merged during the loading process into a single, effective foundry constraint group.
- layerFunctions can only be defined for layers in that techfile – not in referenced techfiles. Said differently, a layer function must be defined in the techfile in which that layer is defined.

Conclusion

The current technology setup appears to be working successfully, although testing is ongoing. There are currently limitations in the round-trip flow (unrelated to the technology setup) that are being addressed that will enable the completed testing of this flow by National. However, based on the available features and software performance, the current technology structure is meeting the requirements of the design flow. All basic technology is stored in the analog technology library. Digital technology (including LEFdefaultRouteSpec, non-default rules, and site definitions) are stored in the digital techlib. And vias that are design-specific, as well as any design-specific constraint groups (such as design-specific non-default rules) are stored in the design library. The result is a maintainable, scalable, efficient, and extremely flexible technology library structure.

Acknowledgements

Many thanks to go to Dennis Lau and Sini Mukundan of National Semiconductor, who supported and took personal ownership of this project. Thanks also to Marie Kunesh and Nishank Gupta at Cadence who provided guidance and editing expertise along the way.