Tips and Pitfalls: Maximizing Custom Design Tool Interoperability and Choice through OpenAccess

Mark Waller, VP of Research and Development, Pulsic

The promise of the OpenAccess database format is simple but powerful: standard interoperability. This promise is very attractive to custom design teams that want to be able to use best-in-class tools in their design flows, regardless of supplier. The reality, however, is that users sometimes experience less interoperability than they had expected, and find that their choices are more limited than they had imagined. Often, this lowered level of interoperability is caused by choices users make as they migrate from proprietary database formats to OpenAccess. This paper will discuss several common migration pitfalls that can cause custom design teams to experience a lower-than-expected level of interoperability. It will also provide tips to help maximize the level of interoperability, and thus tool choice, gained for their custom designs through their migration to OpenAccess.

Migrating to OpenAccess: Optimizing the Path to Interoperability

When making a migration to a new database format, users naturally want to take a path that will change their current flow as little as possible. Accordingly, OpenAccess has been designed to accommodate a very easy migration path that minimizes changes in library elements and flow. However, the "easiest" path can actually have pitfalls that limit interoperability. A path that avoids these pitfalls will maximize interoperability and tool choice.

In general, migration pitfalls are caused by user choices that fall into two categories: maintaining or adopting proprietary data types that limit interoperability, and failing to adopt features in OpenAccess that increase interoperability.

Pitfall: Using Proprietary Pcells

A prime example of the "proprietary data" pitfall arises with Pcells, which are parameterized cells whose layout is dependent on user-defined parameters. Pcells can be defined using numerous languages, some of which result in better interoperability than others. Understandably, users are often hesitant to move to another language, preferring to continue to use a language they know well, even if it is proprietary. However, use of proprietary languages, no matter how well users know them, will limit interoperability and thus tool choice.

Tip: Avoiding the Pcell Pitfall

There are several new, open languages are available for defining Pcells that offer greater interoperability than proprietary languages. Any of these languages will enable any tool vendor to interpret Pcells correctly. The leading choices in terms of current market adoption are Python and TCL. Python is a free object-oriented language that some suppliers have adopted for Pcells. Experienced TCL users can used the native OpenAccess/TCL interface to define Pcells. This interface is not as sophisticated as some other OpenAccess interfaces, so those new to TCL may find this path difficult. But for those already familiar with TCL, this could be a good path, both in terms of adoption curve and interoperability. In addition, the new Si2 (www.si2.org) OpenPDK coalition may result in new, open languages.

Tip: Easing the Pcell Transition

- Decide which of the more interoperable languages is best for a particular design team.
- Use the transition to a new process as the trigger for the migration to the targeted interoperable language. Pcells are often redefined for every process, so there is no duplication of effort, but big gains in interoperability!

Incomplete Migration: The "Easy" Path that Limits Interoperability

It is natural for design teams to want to limit the work involved in making the migration to OpenAccess. However, sometimes the "easiest" path doesn't provide all of the possible benefits of OpenAccess. Design teams often fail to adopt all the available features when they migrate to OpenAccess, choosing instead to maintain as much of their current, often proprietary, flow as possible. In their effort to minimize the "overhead" of the migration to OpenAccess, these design teams hobble their efforts to achieve maximum interoperability.

Pitfall: Maintaining Proprietary Layer Purpose Definition

Custom design automation tools require "layer purpose" as an input. To enable tool interoperability and maximize choice, a common approach to identifying layer purpose is needed. Design teams may have existing objects paired with layer purposes in an ad-hoc manner. However, all tools may not understand these object/layer purpose pairs, so their use could limit tool choice and/or create confusion or additional work when integrating with new tools.

Tip: Avoiding the Layer Purpose Pitfall

The newest version of OpenAccess has a richer variety of object types, each of which has a pre-defined layer purpose. If design teams employ these special objects available in OpenAccess, any tool can understand the layer purpose, because it is embedded in the definition of the object.

Pitfall: Maintaining Old Technology Database Formats

The OpenAccess technology database defines devices (such as vias) and metal layers for each process, including spacing rules and other process constraints. Design constraints can be part of the technology database as well. However, while process definitions will be the same for every project targeted to a particular process, design constraints will vary for each project. If these are data are stored together in one technology file, the file must be edited and updated for every project – a time-consuming and error-prone process.

Tip: Avoiding the Technology Database Pitfall
OpenAccess has a new hierarchical technology database, which enables design teams to share common process information across projects but maintain separate design constraints data by changing only that level of the hierarchy. While the migration to the new hierarchical database might seem to create additional work, once the migration is accomplished, design teams can save work over multiple projects through the adoption of the new format.

Pitfall: Maintaining Highly Customized Vias
Historically, vias have been highly customized using proprietary languages. The newest version of OpenAccess has a very robust standard via, which can be parameterized to meet nearly every need, with support for even drawn vias. However, the option to define custom vias is still available. Unfortunately, many design teams wishing to avoid additional migration work continue to use their old custom vias, which may be defined in proprietary languages. Continuing with this approach limits interoperability and tool choice.

Tips: Avoid the Customized Via Pitfall

- Adopt the new standard via available with OpenAccess – nearly every design team will find that this new via format meets their needs.
- If custom vias are truly necessary, defining the vias using an open language such as Python or TCL can provide a more interoperable solution.
- As a last resort, use an evaluator. However, be sure to use one based on an open language. Also, be sure to read the OpenAccess manual for all the details of how to do this.

Pulsic Support for OpenAccess and Interoperability
Custom designers often design by hand, rather than using design automation technologies. However, at advanced process nodes, hand-design becomes more difficult and these habitual hand-designers may be looking to adopt custom design automation technologies. Standards such as OpenAccess can make this adoption of automation smoother as users move to advanced nodes.

Pulsic offers custom designers full support for the OpenAccess format. Users can specify preferences via a dialog interface. Pulsic is a long-standing member of the Si2 OpenAccess Coalition and has recently joined the Si2 OpenPDK Coalition.

In addition, Pulsic is also working with other industry leaders on another interoperability pitfall: non-standardized design constraints. While OpenAccess offers a standard for interoperability for design data, the representation of design constraints is anything but standard.

Full Adoption of Standards Empowers Design Teams

OpenAccess, like all standards, was founded on the premise that customer data belongs to the customer, not to any one vendor or flow and thus, it inherently supports user choice of design tools. Choice empowers design teams to create the flows that best meet their needs.

While maintaining current data types, languages and methodologies can seem to be the smoothest migration path to OpenAccess, there are instances where this clearly limits interoperability and design tool choice. If design teams can avoid the pitfalls of maintaining proprietary data types and incomplete migration, OpenAccess can provide the interoperability custom design teams seek to maximize their choice of design tools.