Learning OpenAccess

Problem Areas Programmers Need to Understand

Kevin Nesmith
Chief Architect

June 25, 2014

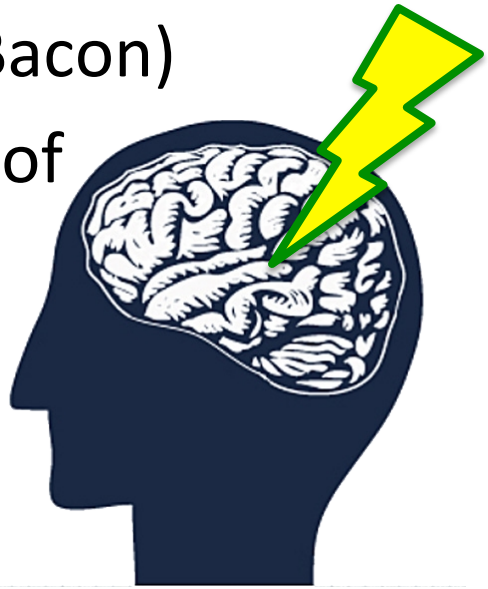
Overview

- Why teach “Learning OpenAccess?”
- Why OpenAccess?
- Documentation and training
- Problems getting started
- Utilities
- lib.defs (cds.lib)
- Domains
- Observers
- Namespaces
- Translators
- Hierarchy
- Some helpful tools
  - oaScript
  - oaDebugging Suite
- Who’s involved in this OpenAccess effort

Why teach “Learning OpenAccess” at IEEE?

“ipsa scientia potestas est”

- “Knowledge itself is power” (Sir Francis Bacon)
- Knowing more about the inner workings of OpenAccess will make you more
  - Productive
  - Valuable



- Improve your job related problem solving skills
- Reviewing what you already know keeps the information fresh in your brain
- Even if you only remember 1% from today, its more than what you knew yesterday

Background of OpenAccess

- mid 1990's, SEMATECH created Chip Hierarchical Design System: Technical Data (CHDStd)
- 1999 SEMATECH asked Si2 to take ownership of the CHDStd program to find a way to make is successful
- This eventually lead to a new project called OpenAccess
- To address concerns of CHDStd, a replacement for the CHDStd API was needed
- Si2 put out a call for a technology contribution
- Cadence answered the call

WHY OPENACCESS?

OpenAccess as a Concept

- Eliminate translation steps in the EDA flow
- Prevent loss of data in transfer between tools
- Standardize representations semantics, avoiding conflicts or ambiguities in data interpretation
- Centralize support for name mapping, data integrity, and other routine database chores
- Provide dynamic accessibility from a common repository of all design data
- Enable high-performance loops among tools (placement, timing, extraction, etc.)
- Realize plug'n play of multiple vendor tools and proprietary applications into a flow

OpenAccess Interface Standard

- An Information Model defined by a collection of entity relationship diagrams (UML).

  This model describes a conceptual perspective of the objects and their relationships, but also includes some specification level detail, such as accessibility across object relationships.

- A Data Model defined by C++ header files, which specify class and function interface details.

- The API Specification, which presents the header information in a more readable format and includes additional constraints.

  With the use of a third party tool Doxygen, the details to the implementation code are tied to the documentation details, thereby improving the accuracy and consistency of the API documentation.

OpenAccess Reference Implementation

- Transfers technology to the public that includes a high-end, commercially successful design database implementation
- Demonstrates strong future commitment to OpenAccess from a major EDA vendor, proven by over a decade of continuous support and development
- Offers an extraordinarily low-cost quick start option for
  - Experience companies with pre-existing databases could benefit from a next-generation implementation
  - Newer or smaller companies looking to leverage the significant engineering resources that the reference implementation contains
- Provides a common development and deployment platform for a wide variety of EDA products, already in commercial use by major EDA and semiconductor vendors across the globe

OpenAccess
OpenEvolution Process

- Centralized information related to the API Standard, Reference Implementation, and auxiliary components, including bug tracking, enhancement requests, information exchange, and education materials.
- Coordinate development of specification and code, including support of working groups, other collaborative activities, and community contributions.
- Distribution of the software, documentation, and related information via the website.
- Protect the rights of participants in OpenAccess through joint use agreements to protect the rights of everyone involved and avoid legal side-effects of cooperative efforts.

EDUCATIONAL MATERIALS

The Si2 Website

One stop for all OpenAccess resources

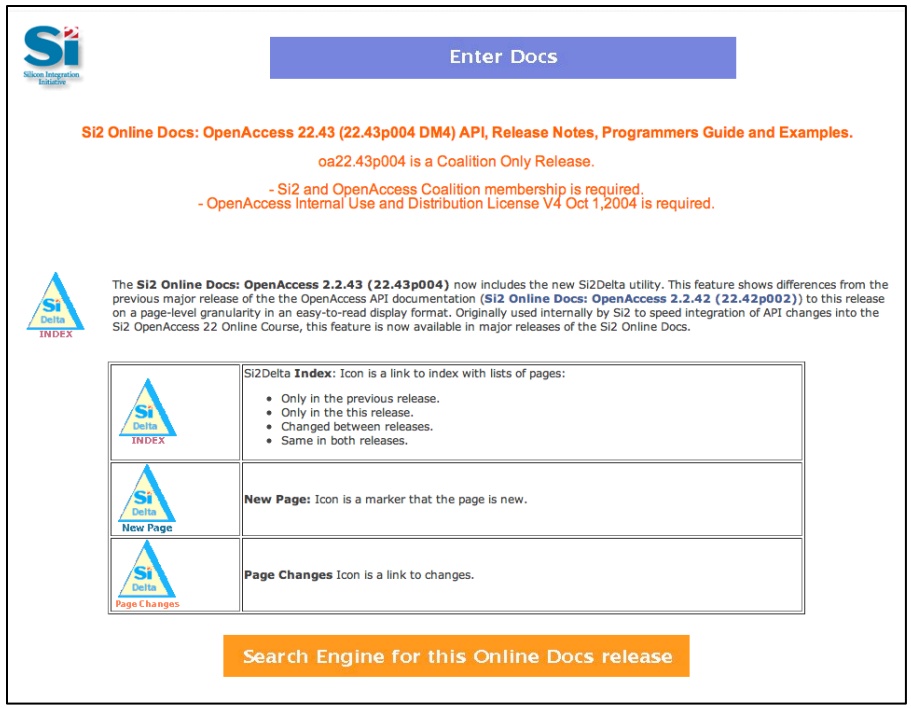
http://www.si2.org/?page=1181

- OA Releases
- OA License and Download Policies
- Documentation and Training
- Labs and Examples
- Community Contributions
- Official OpenAccess Support
- OA Community Wiki

OpenAccess Documentation

HTML based documentation

- Download or use online versions
- Online search features
- Si2 deltas to show changes and additions to previous release



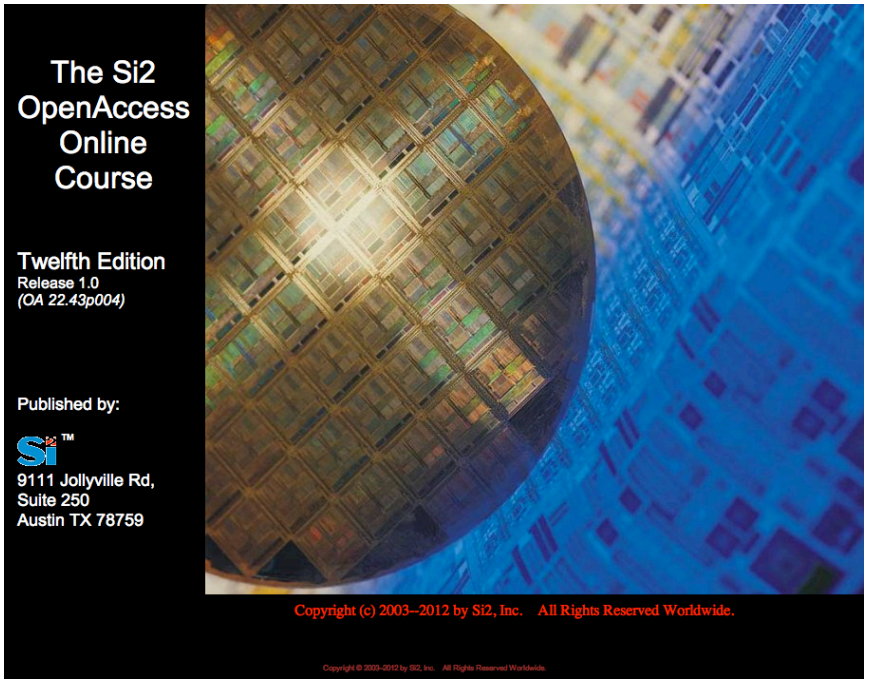
OpenAccess C++ API Documentation Page #1

- Getting Started
- OpenAccess (API, Guide, Coding Style, & Diagrams)
- Migrating to Newer OA Versions
- Extension Languages
- Translators
- Plug-In Interfaces

So much information in one location!

Tutorials & Labs

- Si2 OpenAccess API tutorial PDF
- Web based API tutorial, including a HowTo webinar
- Labs with source code



Tools and Applications

Many tools and applications donated or supplied by OAC member companies.

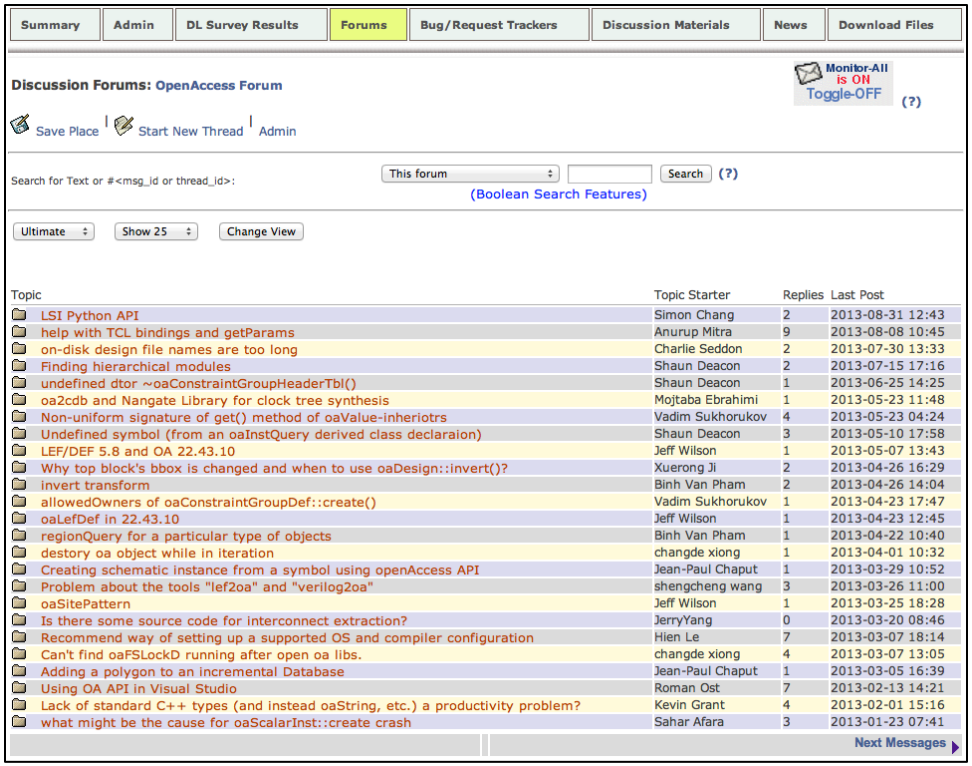
I bet this is already on your machine

OpenAccess Downloads - Contributions

- Synopsys oaCompare
- Cadence oaScan
- Cadence OA Multi-Patterning Technology (Colors)
- LSI Python for OpenAccess
- Ciranova PCell Caching
- Synopsys oaViewer
- Si2 oaDebugging Suite
- Analog Symbol Library
- Si2Delta
- Nangate FreePDK45 Generic Open Cell Library
- oaScript Extension Language Bindings
- Contributed Code and Snippets Library
- Cadence cdsLib plugin
- OpenAccess Wiki (oaScript & Si2oaD docs +)
- Open Modeling Calculation Interface (OMCI)
- List of All OpenAccess Download Projects
- Contributed OpenAccess Application Notes

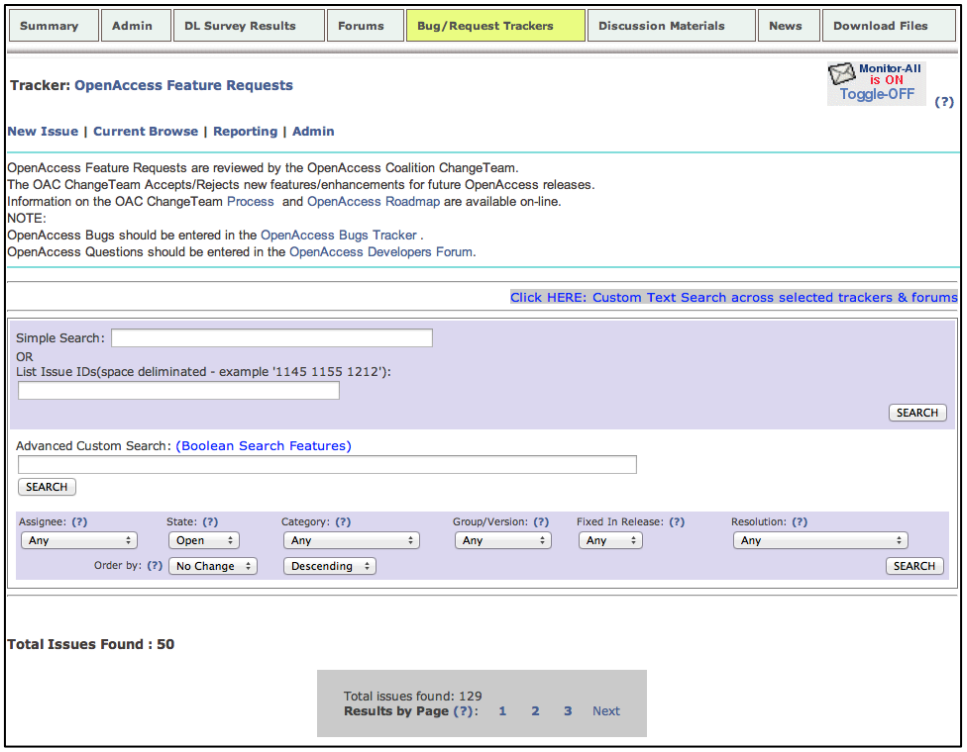
Forums

Forums are available for informal discussions related to OpenAccess. Here OpenAccess users can freely exchange information, ask for help, and make suggestions. At some point a discussion might culminate with a formal issue logged in one of the trackers.



Bug/Request Trackers

Bugs in the documentation, the API, the Reference Implementation, or any related components, should be logged in the bug tracker. These are reviewed regularly by the OA Integrator (usually Johannes Grad of Cadence), who prioritizes the issues based on relative importance and ease of implementation, makes appropriate changes to the code, and schedules the fixes for inclusion in future releases.



PROBLEMS GETTING STARTED

Defining Your Own Standard

In the OpenAccess documentation…

"OpenAccess Coding Style Guide and Standards"

- Great place to refer to when defining a consistent way to program
- Build from this to define your own, or in-house rules
- Can even be adapted for scripters
- Put these rules into your editor of choice

Compiling for the 1st Time

- Not for the faint of heart
- Follow the directions from the release notes
- Remember, **`OA_UNSUPPORTED_PLAT`** is your friend
- This will save your hair…

  "**`export COMPILER_PATH=/usr`**"
- Don't try to build the "extra" stuff until you are comfortable building OA. Comment out APP_PACKAGES from the GNUMakefile in the oa directory.
- Windows is trickier. Just work with one configuration option first. Once working, copy it to all other configurations.

LD_LIBRARY_PATH & PATH

- On Linux, the LD_LIBRARY_PATH points to **<oaDir>/lib/linux<version><bits>/{opt,dbg}**
  - Problematic when machines already have an OA installation.
  - When in doubt, put your version of OA first in the list.
    export LD_LIBRARY_PATH=<myOAversion>:$LD_LIBRARY_PATH
    This will force your OA to load first.
- On Windows, similar issues, but since we are working with DLLs, your environment variable is PATH.
  PATH should point to
  <oaDir>\bin\<win version>\{opt, dbg}
  - If problems persist on machines with pre-existing installations, try putting your new <Path to OA> in your environment first, from the control panel or command line…
    set PATH=<myOAversion>;%PATH%

OPENACCESS UTILITIES

OpenAccess Utilities

OpenAccess provides several utilities to make some tasks more straight forward. These utilities are located in the **<oaDir>/bin** directory.

- **sysname** – returns the <platform_name> for your computer and operating system
- **oaGetLibPath** – returns the path to the shared libraries depending on which platform you use
- **oaGetVersion** – returns the version numbers of the OpenAccess libraries
- **oazip** – a tool to use the compression capabilities built into OpenAccess…

OpenAccess Compression

OpenAccess supports storing the designs compressed. There are times when it is necessary to uncompress the designs. The oazip utility provides the ability to compress or decompress these designs.

oazip supports:

- Compress/decompress design databases as needed.
- Check the OpenAccess design libraries and report the ones that don't match the compression control attribute of the library.
- Query the compression level on a design library.
- Update the OpenAccess design databases in a library that didn't match the compression control attribute of the library.

YOUR LIBRARY DEFINITIONS

lib.defs (cds.lib)

- The **library definition file** is an ASCII text file defining how to resolve logical oaLib name references to an actual physical directory on disk.
- OpenAccess reads a library definition file when a call to `oaLibDefList::openLibs()` is made by the application.
- Information from this file is used to determine the file system location of the persistent library data.

```
DEFINE myLibName /path/dirName
```

lib.defs (cds.lib)

The default libdef PlugIn searches for the lib.defs file in the following order:

1. Current directory
2. The current user's "home" directory
3. In the data/libraries subdirectory of the OA installation,
   example: /opt/oa22.43p028/data/libraries

lib.defs (cds.lib)

File Syntax

- # maybe used anywhere to identify a comment.
- Only one statement per line.
- All white space before and after a statement are ignored.
- Any line not recognized as a statement or comment will generate a LoadWarning event, but *no exception will be generated*.

Any subsequent calls to **`oaLib::find(someLib)`** will return **`NULL`** even though the parse error may have been unrelated to that Lib's definition.

OPENACCESS DOMAINS

Domains

- **Module** – an optionally folded representation of the logical structure only. Module data contains netlist connectivity information (and extension objects) and is usually the result of synthesis of an HDL description (such as VHDL or Verilog).

- **Block** – an optionally folded representation of the physical implementation. Block data contains connectivity, plus it likely has physical implementation specifics such as shapes, wires, and parasitics.

- **Occurrence** – a fully unfolded union of the Module and Block objects. Though unfolded, all hierarchical boundaries remain intact. Includes direct support for traversal up and down the hierarchy as well as to each corresponding object in the Module and Block Domains.

Integrated Domain
The Positives

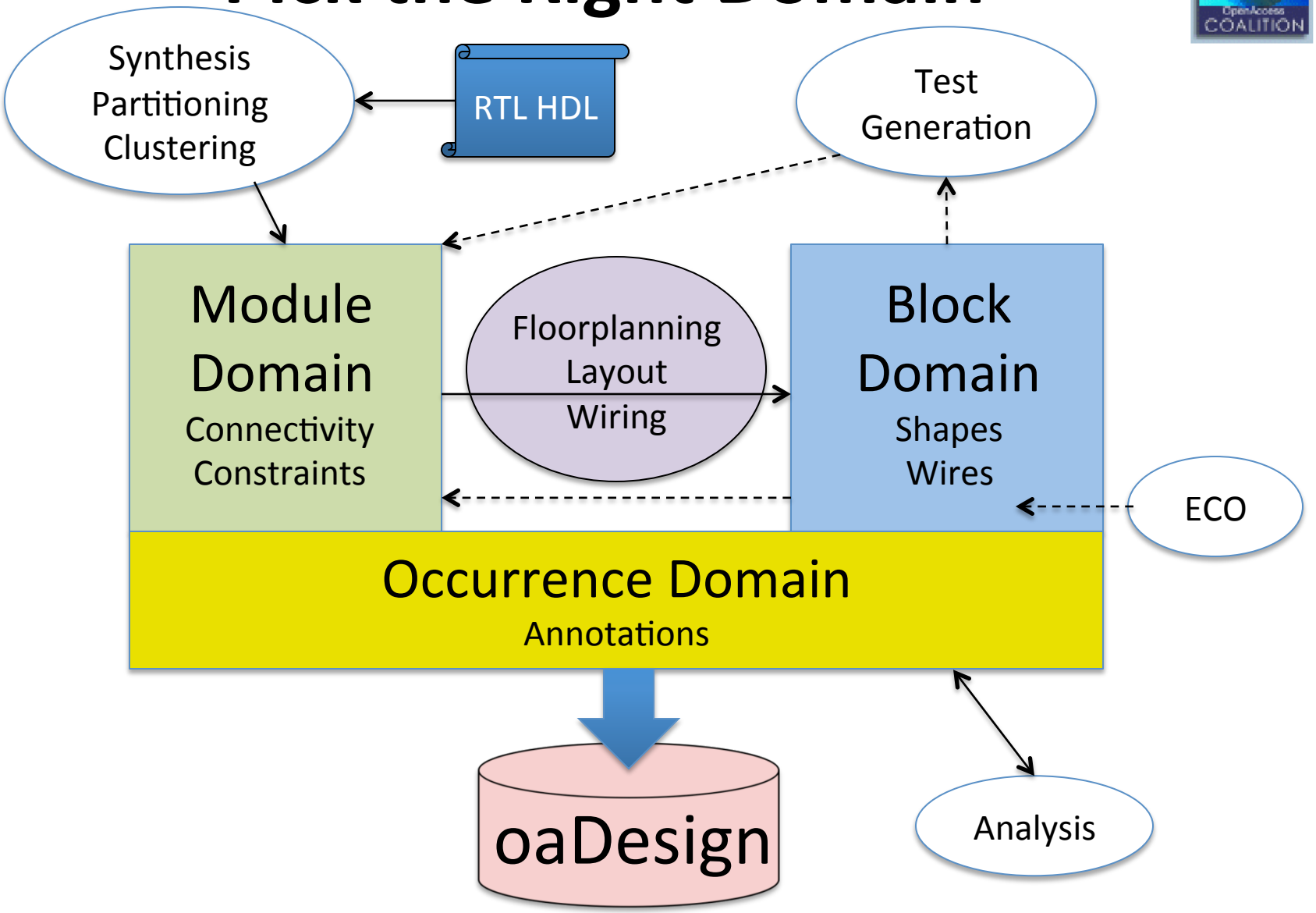
The three Domain representations are fully integrated in important ways.

- **Traversal** across them is **straightforward**.

- The database also maintains **integrity** such that edits performed in one of the Domains are guaranteed to **propagate** their effects appropriately to the others.

- This integration can provide significant **advantages**, enabling floorplanning, prototyping, physical synthesis, and physical analysis to:
  - Reduce the complexity of clock tree and scan chain reordering.
  - Enable ECO application at a finer granularity of hierarchy.
  - Annotate at the occurrence level.
  - Enforce consistency of timing constraints from the original HDL to the simulation test bench.

Pick the Right Domain



Domain Differences

Domain	Objects Available	Objects Not Available	Limitations	Typical Uses
Module	• Logical Data • Extensions: Groups, Props, AppDefs, AppObjects	• Physical Data: Shapes, Routes • Parasitics: Nodes, Devices, Reduced Models	Connectivity may be edited, automatically propagating changes to the Block and Occurrence Domains.	• Synthesis of HDL into an OA model. • Initial partitioning or clustering before placement or wiring. • Implementing independent design descriptions suitable for incorporation in multiple technologies, flows.
Block	• Logical data • Placement • Floorplanning • Routes • Shapes • Parasitics • Extensions		Connectivity may be edited, automatically propagating changes to the Module and Occurrence Domains.	• Floorplanning, placement, wiring. • ECO, auto-propagated to the Module Domain.
Occurrence	• Logical data • Shapes • Parasitics • Extensions		Each Occurrence is a full-fledged object that can be individually annotated, for example, with extensions and parasitic data. Only annotations are allowed in the Occurrence Domain. Connectivity or physical editing is not possible, except via the uniquify() method.	Analysis of data that might be different depending where on the chip the data is instantiated (e.g., because of unique proximities to things like power routes or temperature islands) such as • timing analysis • parasitic extraction

OPENACCESS OBSERVERS

OpenAccess Observers

The Observer is an object-oriented design pattern that defines a dependency between one object (the *subject)* and any number of interested objects (the *observers)* such that a change to the subject causes notification to its observers.

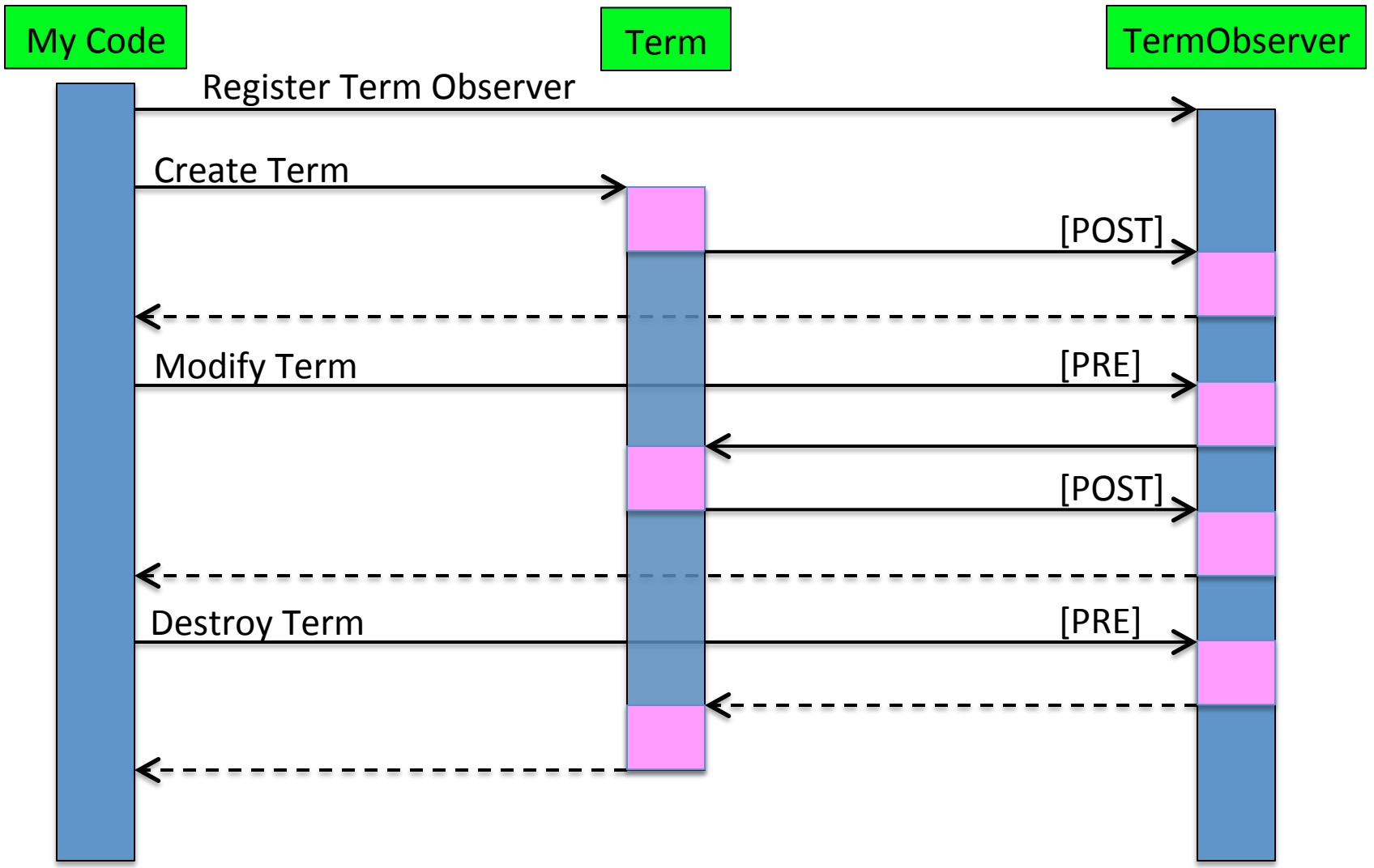
OpenAccess **Observers** provide a way to implement such asynchronous flows of control, based on actions that occur in the RTM, using the *callback* technique.

OpenAccess Observers

Observers enable an application to monitor actions that are otherwise beyond its control. Observers provide the hooks that the application can use to get control when database events of interest occur, including

- Ordinary events
  - Explicit actions initiated by the app like
    - Object creation
    - Object destruction
    - Attribute modifications
  - Implicit consequences initiated by the API implementation, such as
    - Creation of automatic objects
    - Cascading destruction of dependent objects in owner relationships
- Unexpected events…
  - lib.defs syntax errors (discussed earlier)
  - conflicts caused by out-of-context edits of components of a Tech graph

OA Observer Order of Operation



lib.defs Observer

- Set up an **`oaObserver<oaLibDefList>`** to see events and warnings.
- This Observer can intercept
  - onPreOpenLibs()
  - onPostOpenLibs()
  - onLoadWarnings(),
    - which can be used to identify a variety of errors and warnings encountered by the API when trying to parse the file.
- **An application must set up this Observer or it will not be notified of any problems in a lib.defs file.**

OPENACCESS NAMESPACES AND NAME-MAPPING

Name-Mapping

- A DBMS that can perform the appropriate transformations across different name contexts provides a significant benefit for tools. Such transformations within an OA context are called **name-mapping.** The OA API integrates name-mapping into the classes that need it.

- The application tells the API the context of each string as it is used to create a Name object – whether it came from a Verilog/SPEF file format or UNIX/Windows environment, for example. This tells the API how to decode the semantics from the source character representation for that name. That Name object can always be transformed appropriately when an application requests it be represented as a string in a different naming context.

NameSpaces

- The NameSpace encapsulates a set of rules for creating and interpreting legal names within an application and is represented by one of the subclasses of oaNameSpace: LEF/DEF/SPEF/VHDL, Verilog, UNIX, Win, etc. Whenever a mapped Name is created from a string, or a string representation of the name is required, a NameSpace must be used to provide the context for converting special syntax.

- The application controls the use of NameSpaces, for example:
  - Some tools may use the same NameSpace everywhere to create and retrieve names of objects.
  - Other tools will allow the user to set the NameSpace to be used.
  - A tool might even enable the user to change NameSpaces depending on the task being performed.

Application Responsibilities

While the OpenAccess database maps names automatically, applications still must understand name mapping issues and rules enough to recognize how names can change when using data from other applications – and to use the OA name mapping engine properly. When names must be entered at multiple places in a design flow and must correlate, applications must be prepared to form the corresponding name in multiple NameSpaces.

For example:

- When working with a library name that is used in VHDL, the VHDL NameSpace should be used.
- If netlists that contain Verilog designs are to be used, the Verilog NameSpace should be used for those names.
- A library called **MYLIB** in VHDL is subsequently used in Verilog as **mylib**. This library must be defined in the library definition file with a statement like:

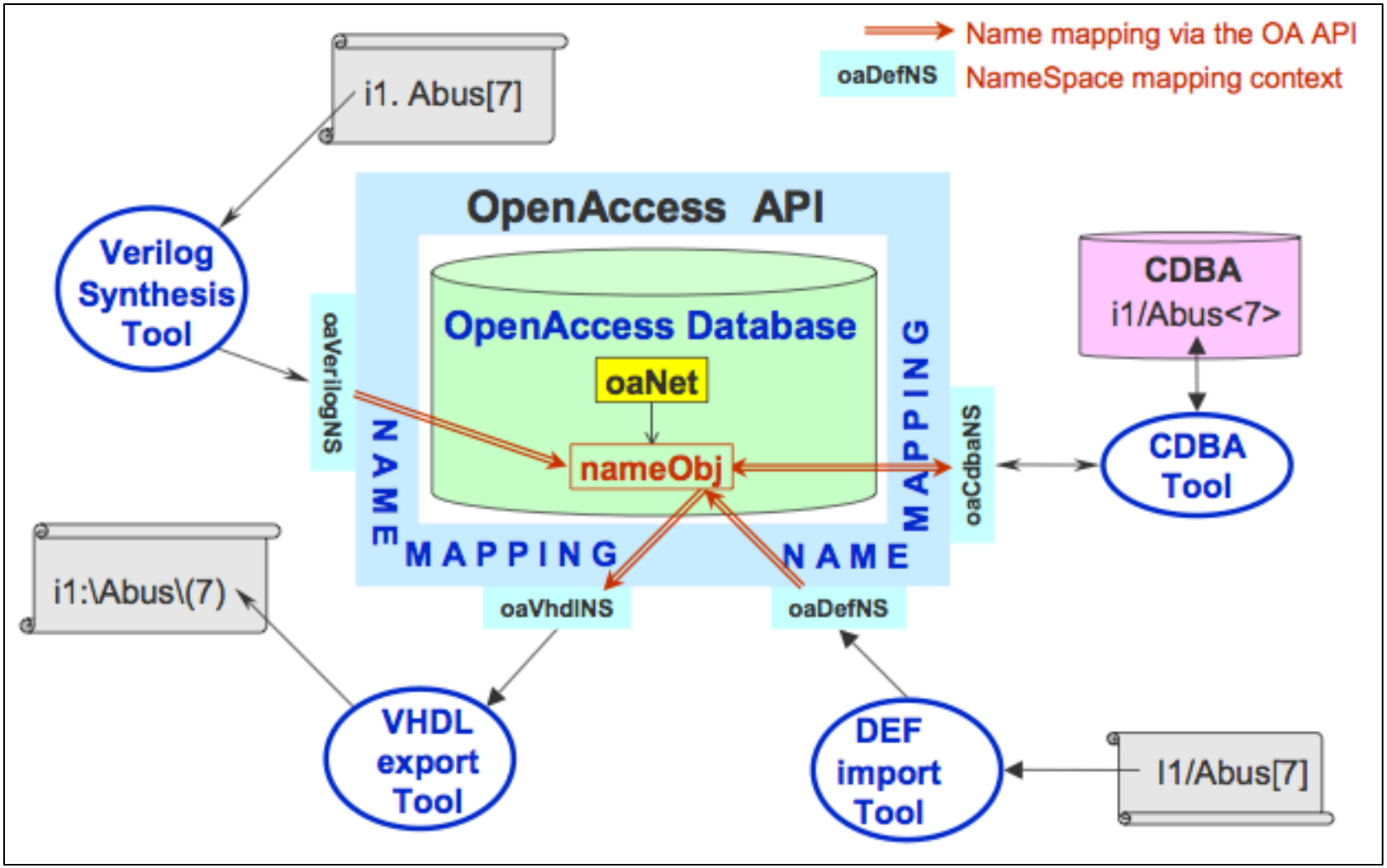
DEFINE mylib /usr/libs/mylib (notice the lack of caps)

NameSpaces Differences

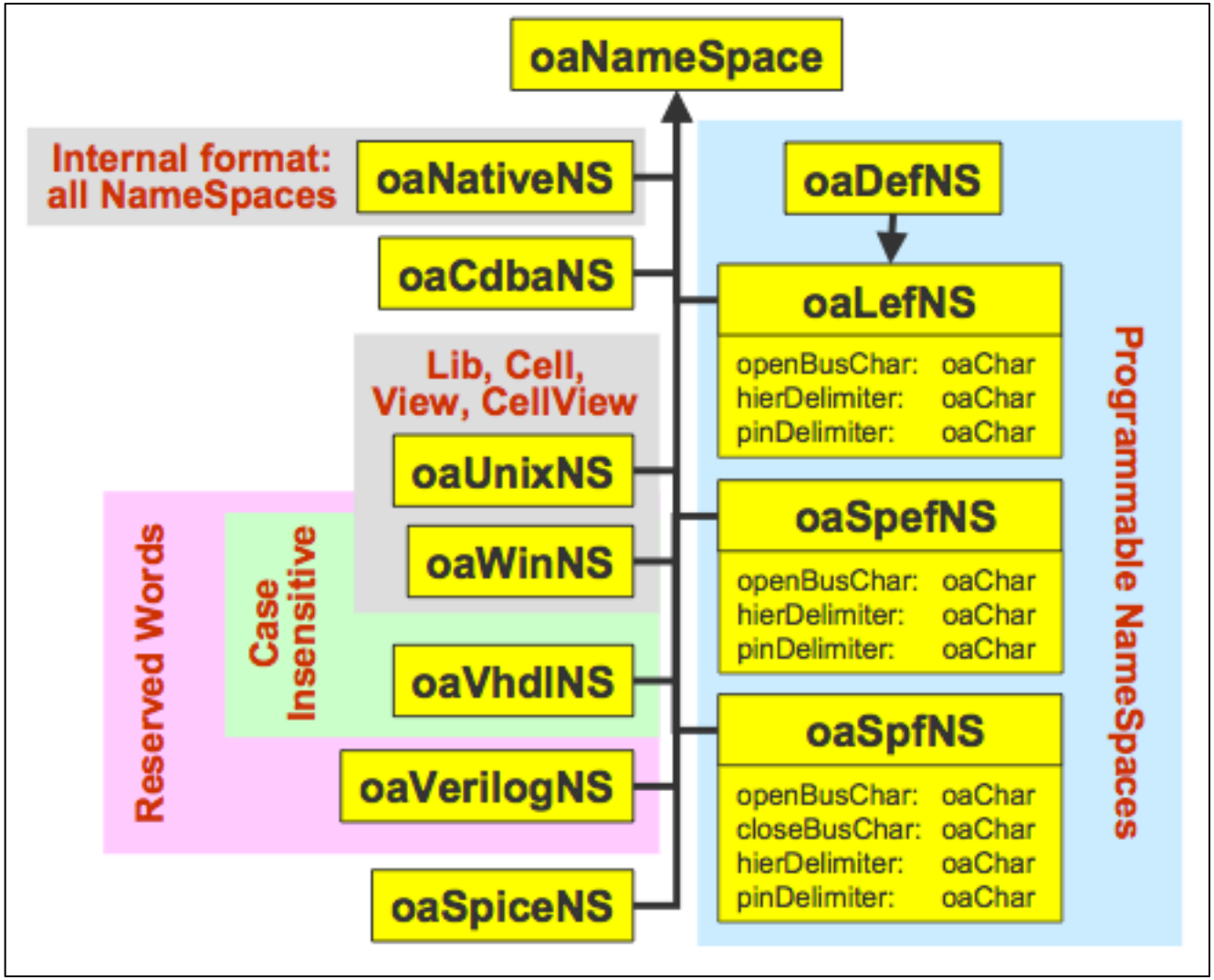
Semantics	Comparisons of NameSpace Rules
Keywords	The string and is a keyword in Verilog and VHDL, while process is a keyword in VHDL but not in Verilog. Many NameSpaces, such as Cdba and UNIX, have no keywords.
Case Sensitivity	In VHDL, the name aaa refers to the same object as AAA. In Cdba or UNIX, these names are different.
Syntax and Characters	Many NameSpaces have an alternative way to include characters in names that would otherwise be illegal. In VHDL, \a+b*\ is a legal name because the back- slash escapes the characters that are otherwise illegal. A normal Verilog name can contain a dollar sign, but a VHDL name cannot unless it is escaped.
Bit indexes	Some NameSpaces have a way to indicate a name that is a particular bit of a vector. The syntax for these names and the characters used to set off the bit index from the rest of the name are different in different NameSpaces. The syntax for indicating a range of names in a vector also varies among NameSpaces. For example, <4>, [4], and (4) are all ways that certain NameSpaces use to indicate bit 4 of a vector. In some NameSpaces, the bit index character can be set by the caller.
Hierarchy	When using hierarchical designs, different NameSpaces use different hierarchy separators to indicate the hierarchical levels of a name.* The slash and period are common hierarchy characters. In some NameSpaces, the hierarchy character can be set by the caller.

* These NameSpaces do not support hierarchy: oaLefNS, oaUnixNS, and oaWinNS.

NameSpaces Complexities



NameSpaces Overview



OPENACCESS TRANSLATORS

OpenAccess Translators

OpenAccess translators use OpenAccess libraries to map and convert data from one format into another.

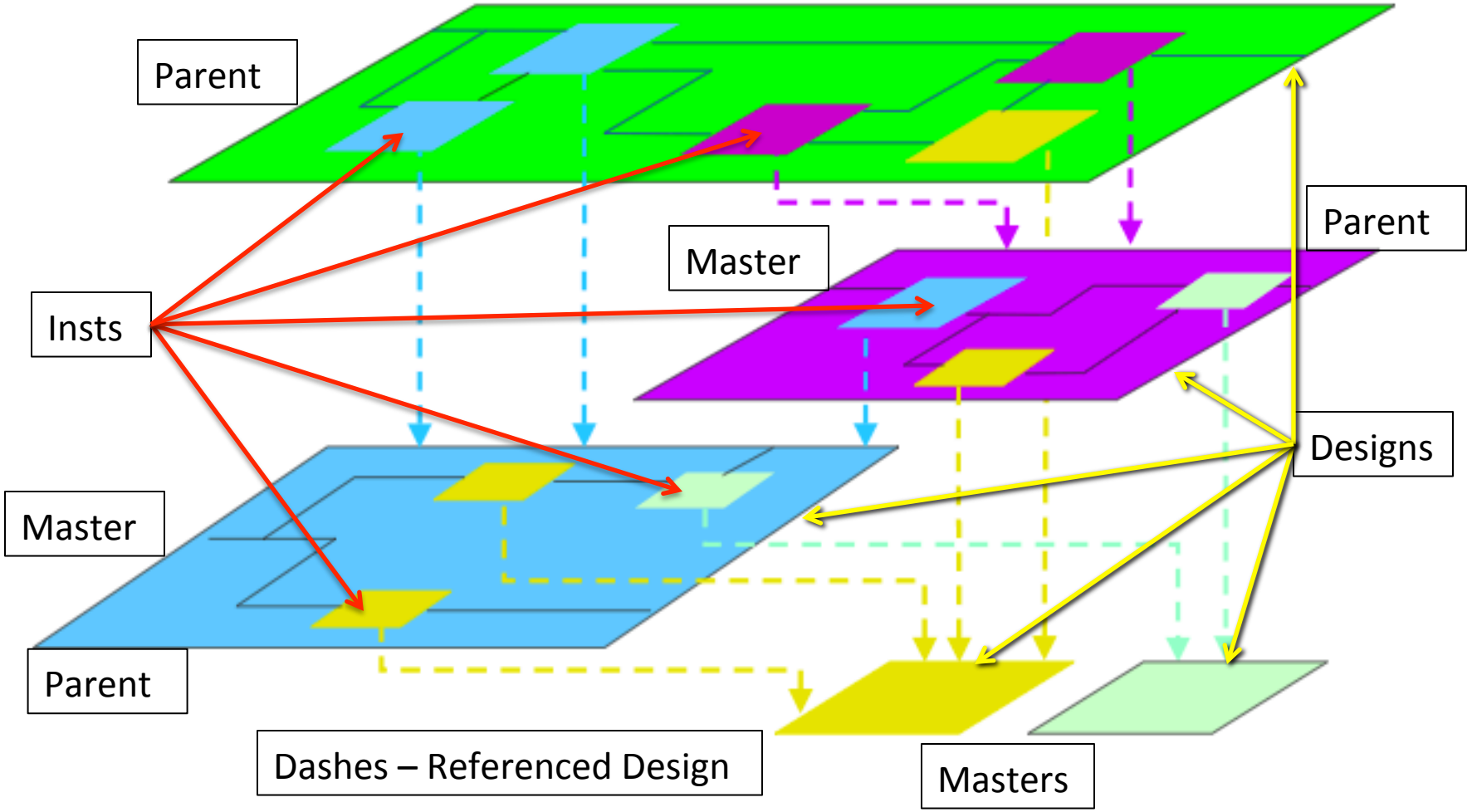
- **Verilog**: the hardware description language (HDL) used to model electronic systems.
- **LEF**: contains technology information, design rules, macro cell definitions, and information about design objects such as vias, sites, and layers.
- **DEF**: contains the design-specific information about a circuit.
- **SPEF**: contains detailed parasitic information for nets that are connected to terms in a block. It extracts reduced parasitic information for nets that are connected to instTerms only.
- **STRM (GDSII, stream)**: a database file format which is the de facto industry standard for data exchange of integrated circuit or IC layout artwork. It is a binary file format representing planar geometric shapes, text labels, and other information about the layout in hierarchical form.

OPENACCESS HIERARCHY

Inst versus Design

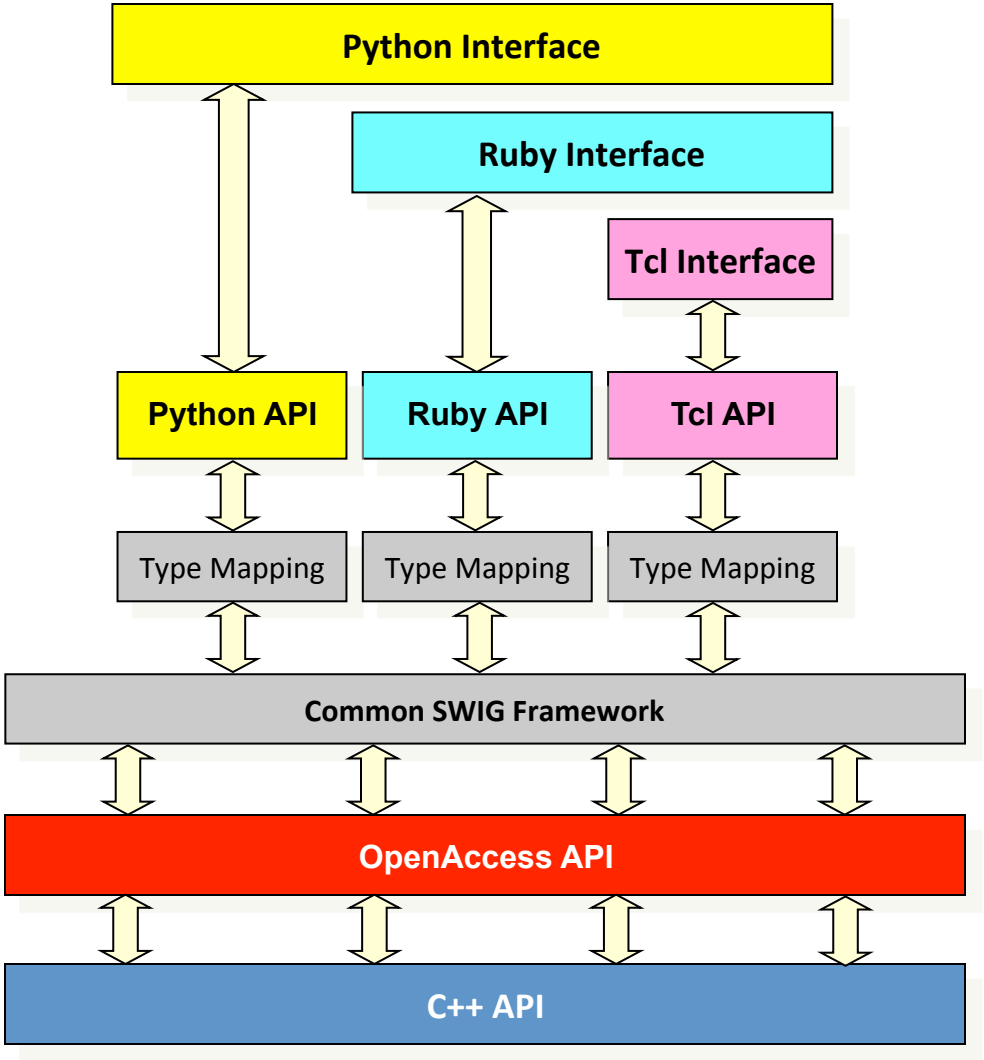
- A design holds connectivity information such as nets, terms, etc., and layout information such as shapes.
- An inst is an object in a design that refers to another design.
- Designs containing insts, which refer to other designs, is how OpenAccess represents hierarchy.
- Insts do not have any connectivity or shape information in them.
- The design that contains the inst is called the parent.
- The design the inst refers to is called the master.

Insts vs. Designs
Master vs. Parent

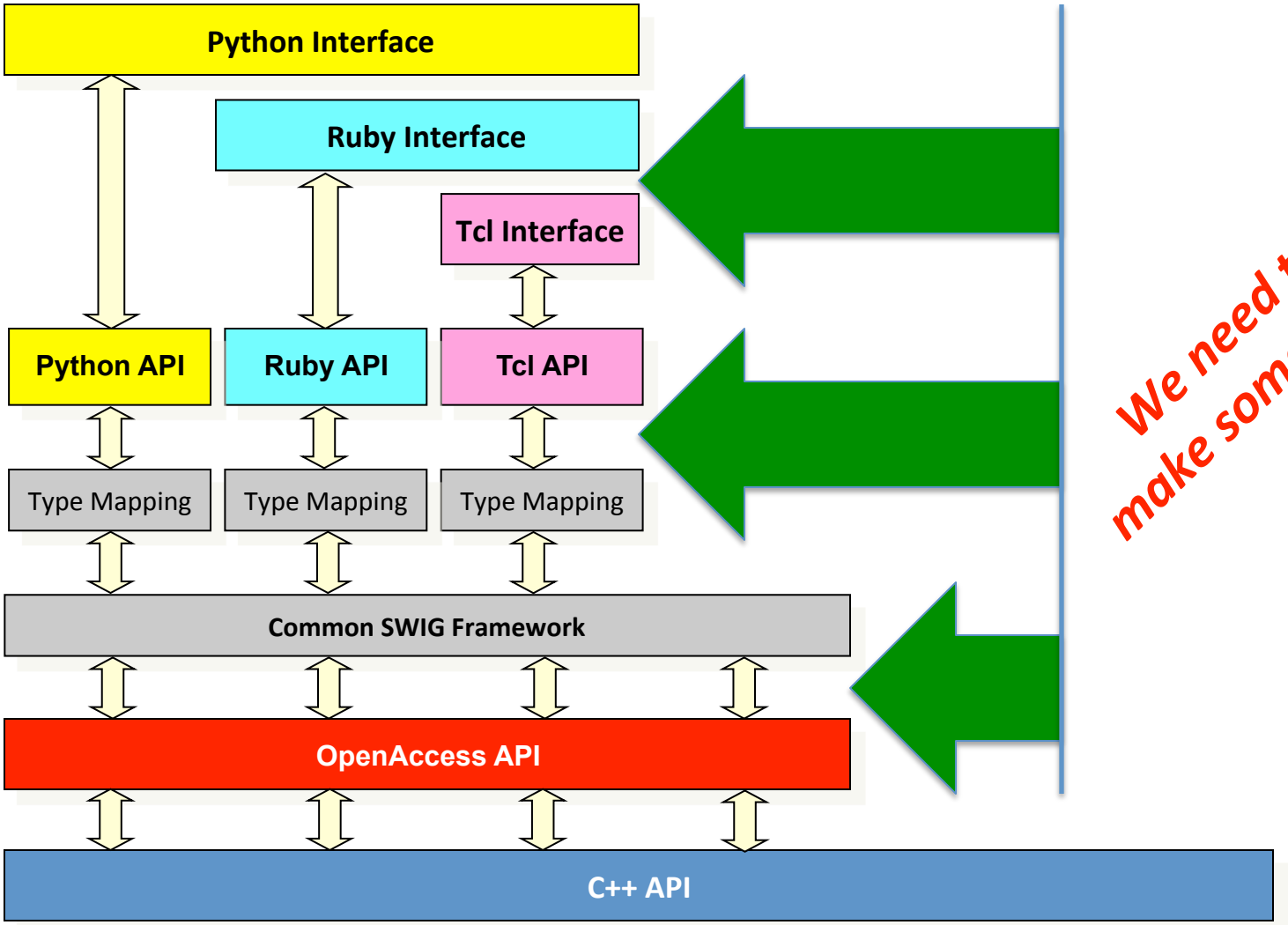


oaSCRIPT

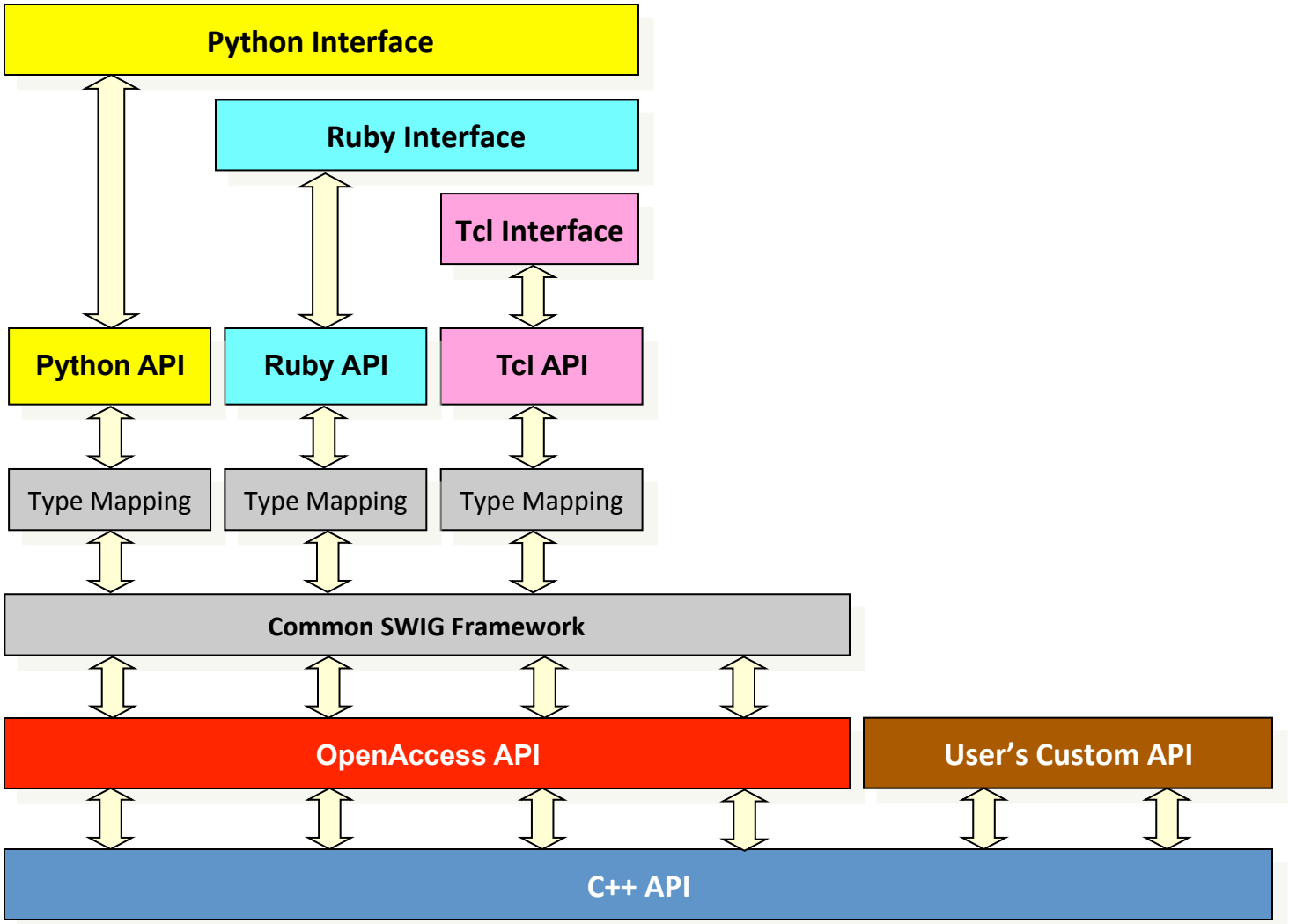
The oaScript Architecture



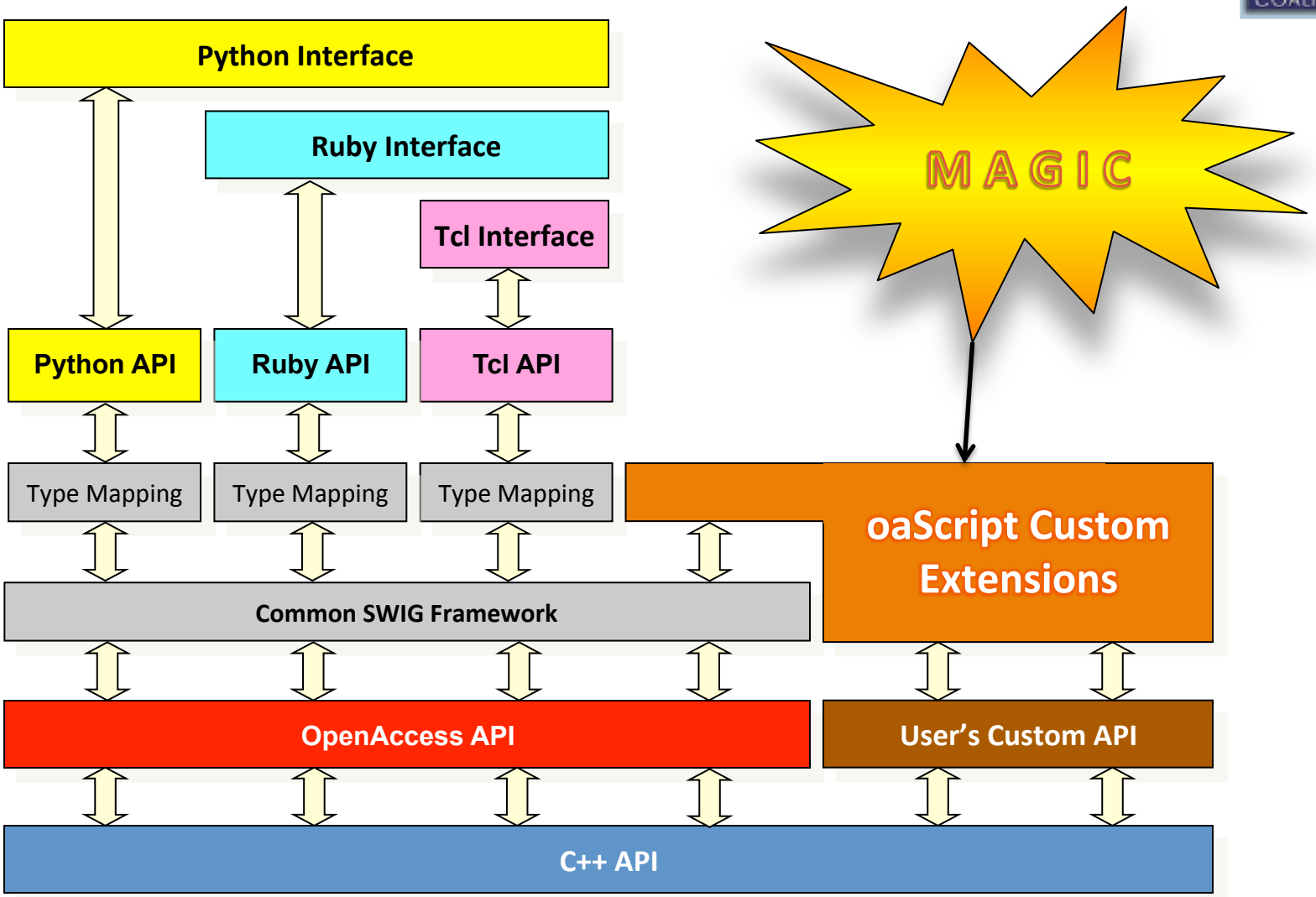
The oaScript Architecture



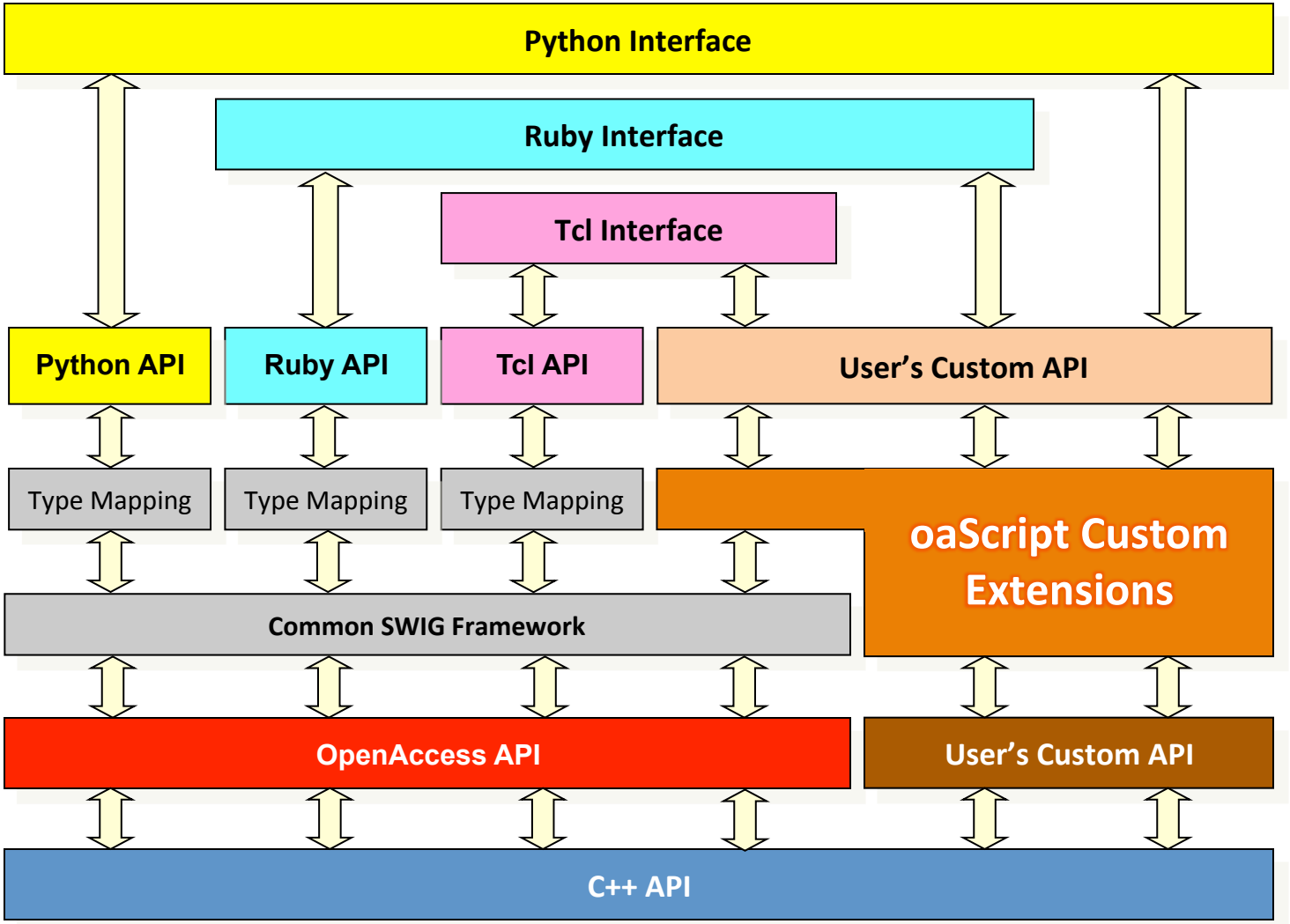
The oaScript Architecture



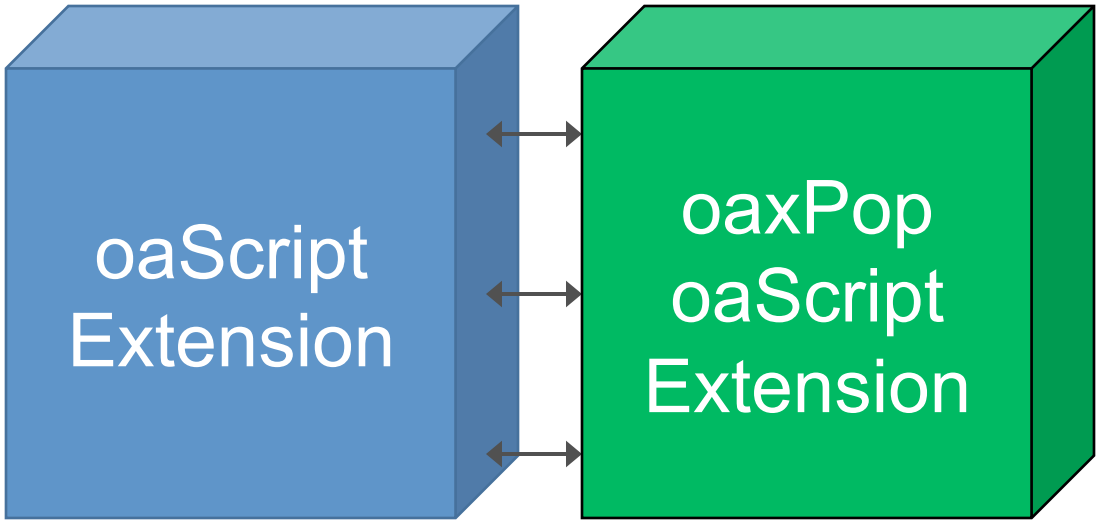
The oaScript Architecture



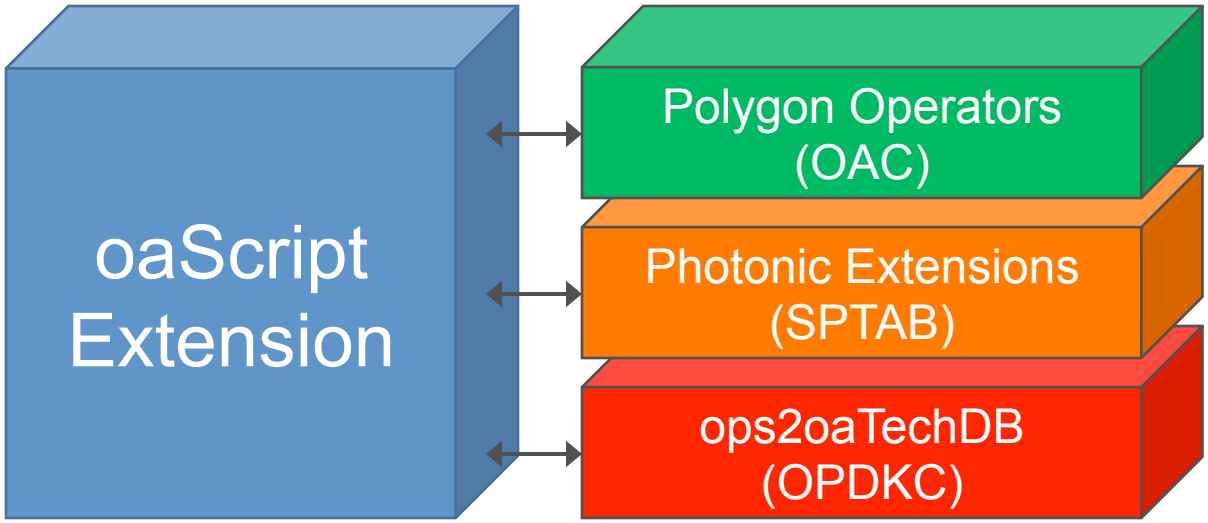
The oaScript Architecture



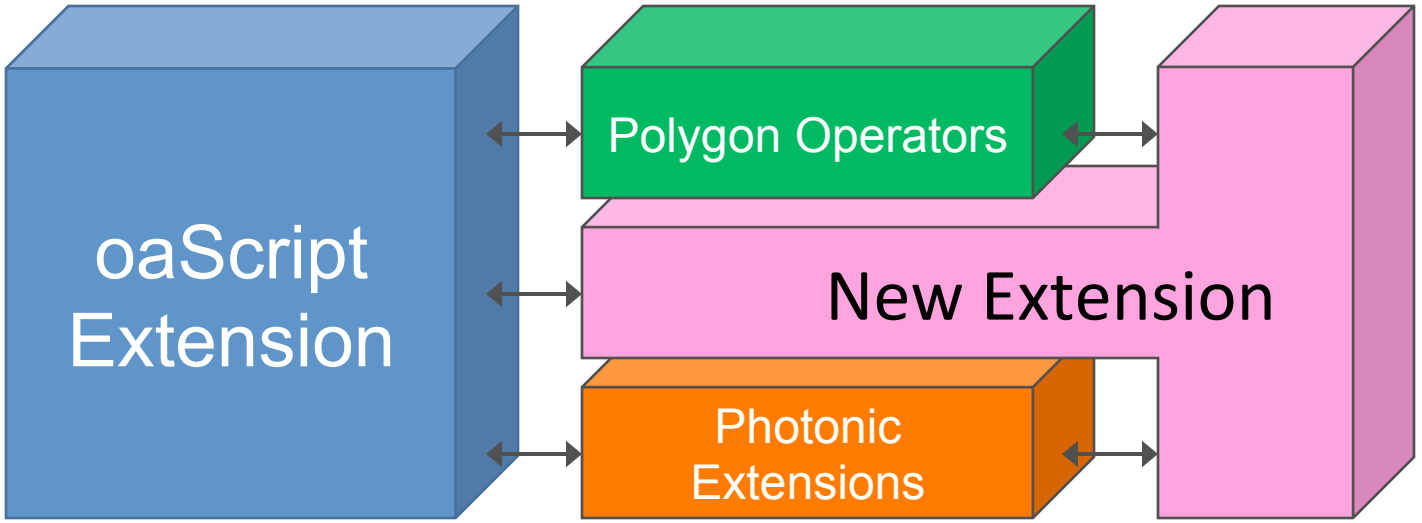
The Polygon Operators Extension



Current Projects Using the Extension Capability



Extend the Extensions



Time to think outside the box
Like and L or T even

The oaScript Support

- Scripting languages supported (Linux)
  - Python 2.5-2.7
  - (NEW) Python 3.3
  - Ruby 1.8-1.9
  - (NEW) Ruby 2.0-2.1
  - Tcl 8.4-8.6
  - Perl (no support)

The oaScript Support

- Scripting languages supported (Linux)
  - Python 2.5-2.7
  - (NEW) Python 3.3
  - Ruby 1.8-1.9
  - (NEW) Ruby 2.0-2.1
  - Tcl 8.4-8.6
  - Perl (no support)
- Scripting languages supported (Windows)
  - (NEW) Python 2.7

Get Involved

- Join the OpenAccess Extension Steering Group

or

- Join one of the ESG sub working groups

https://www.si2.org/?page=88

oaDEBUGGING SUITE

What is the oaDebugging Suite

A collection of tools to help users look into the OpenAccess Database.

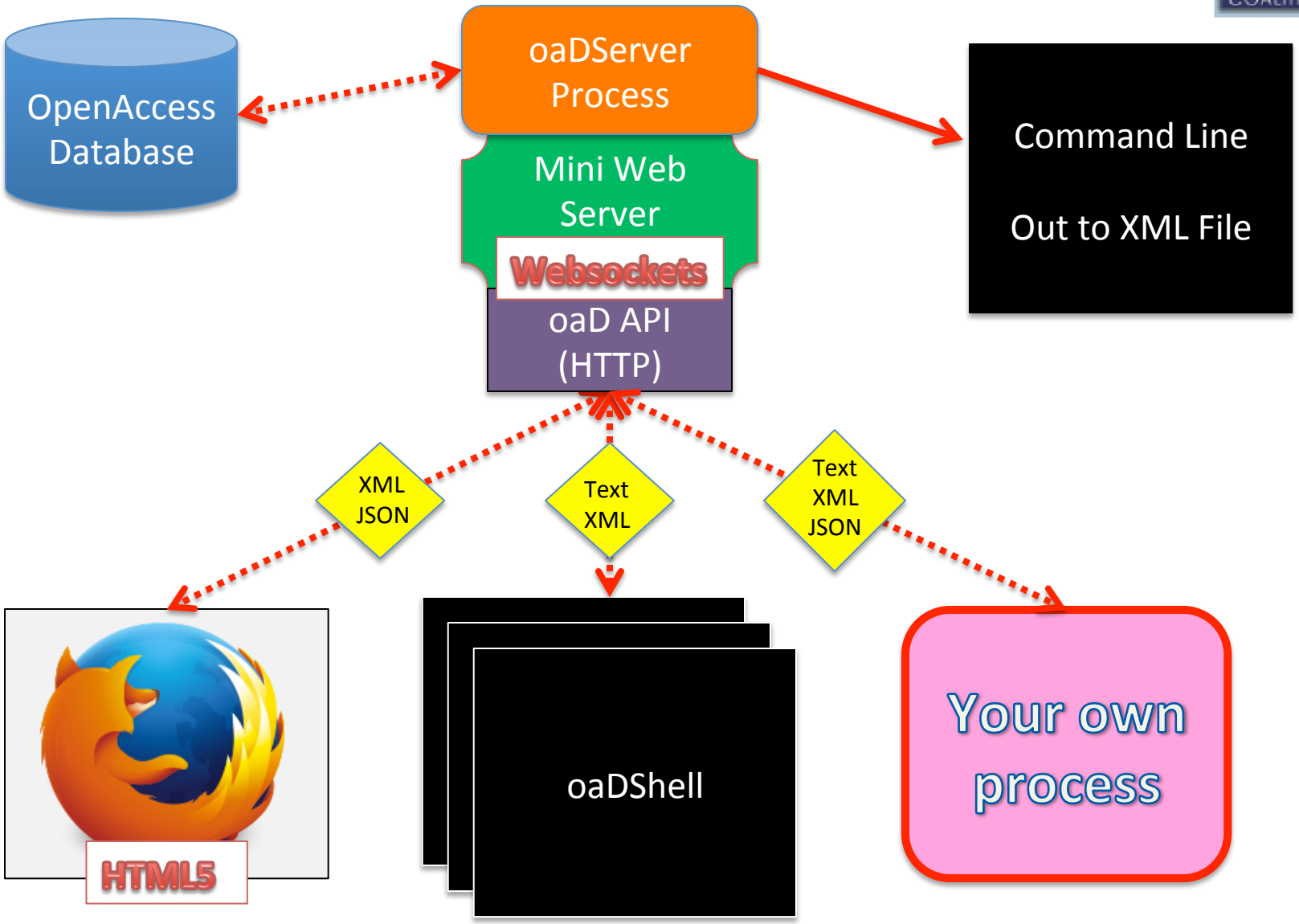
Includes:

- oaDebug
- oaDiff
- oaForensics
- oaDShell (in new release)

Currently supports:

- Linux
- Windows
- oa22.41p004 – oa22.43p028 (current version)

The oaDebug New Architecture



The New Codebase

Coverage of the OA API and Docs have been scripted to generate the C++ class and code to access the properties and methods.

This equates to…

90% auto generated code

10% manually written code

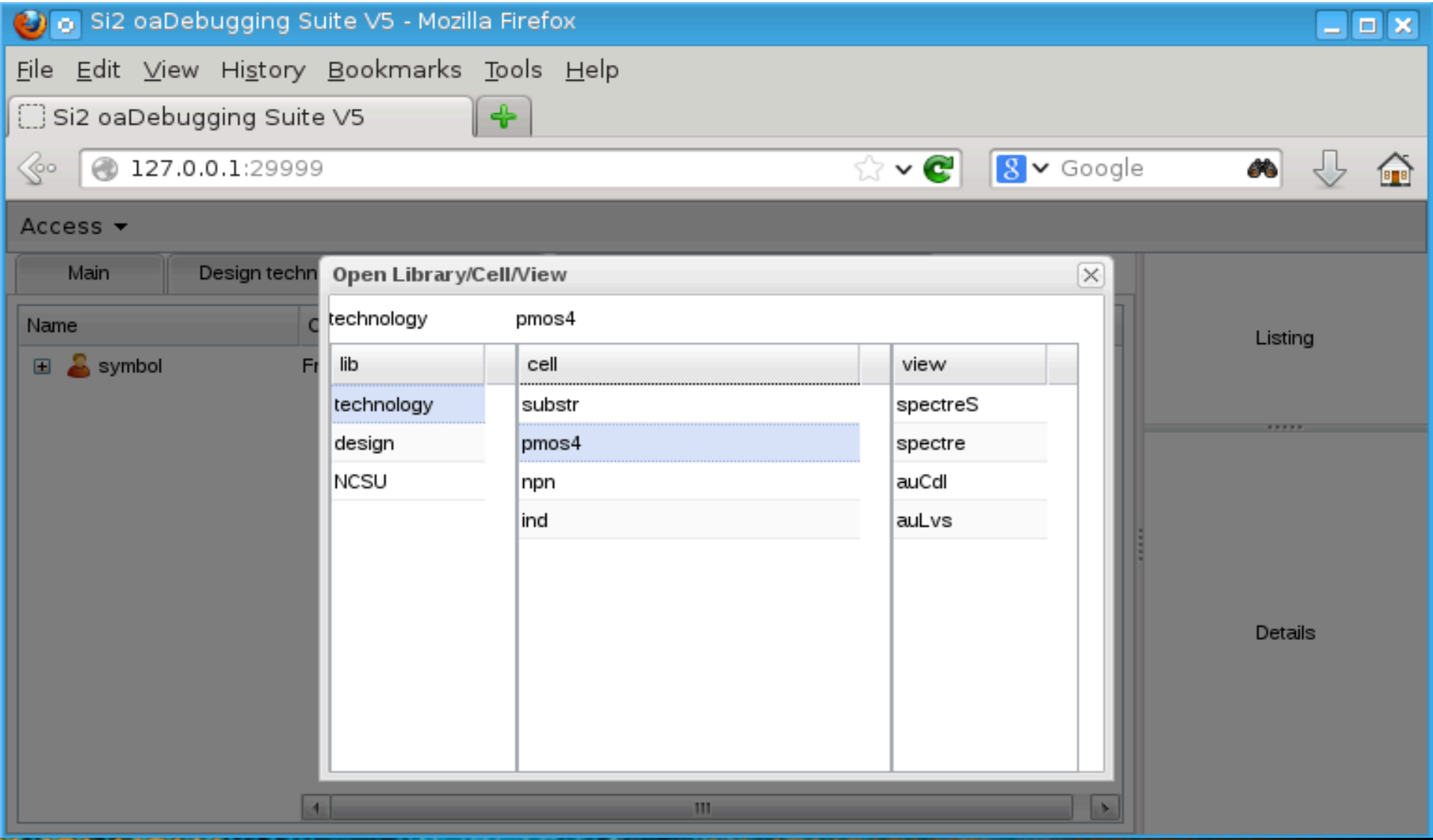
Using Boost for the web server and client, along with some other functionality

Current oaDebug Options

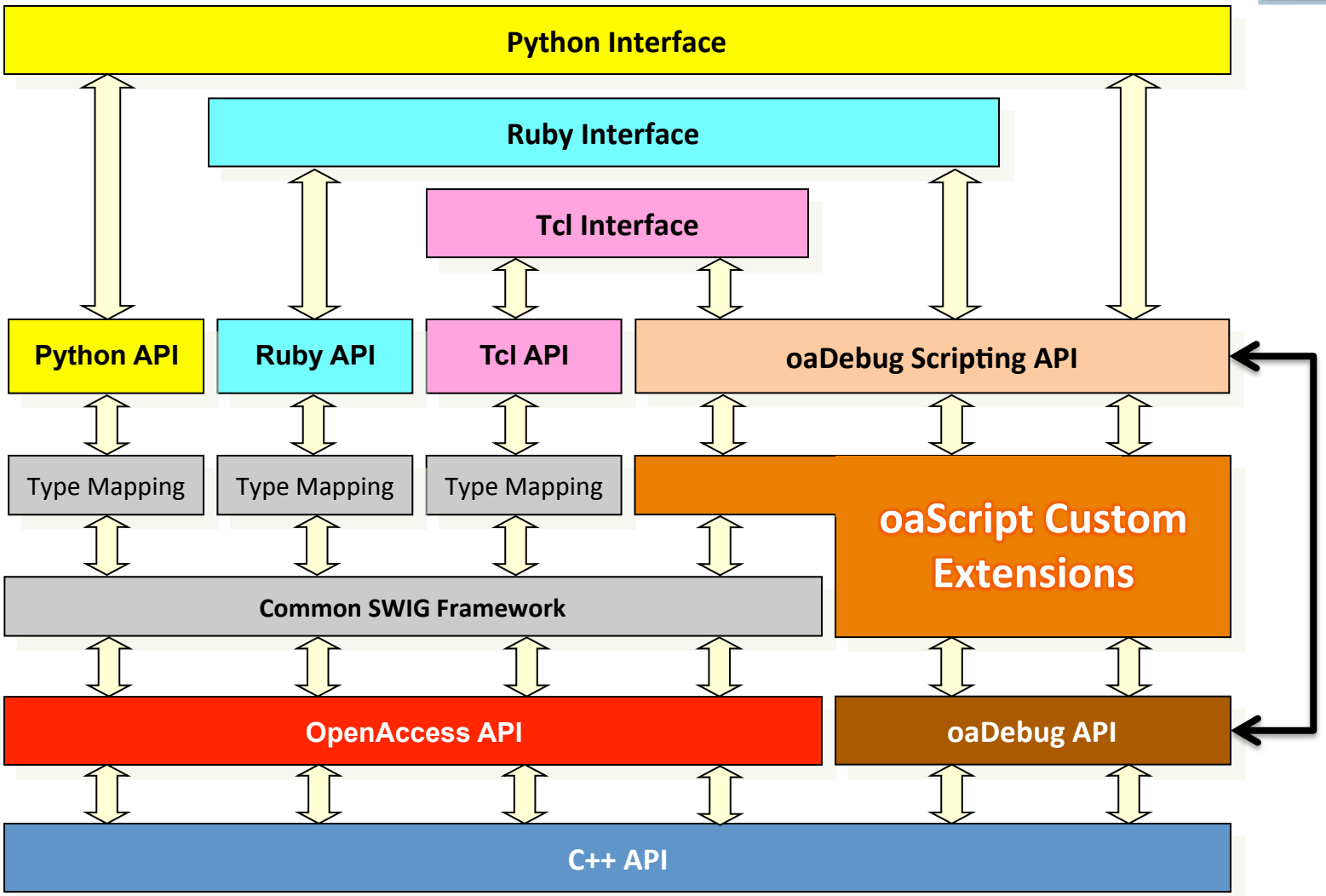
oaDebug

```
Allowed options:
-h [ --help ]                        print usage message
-q [ --quiet ]                       quiet output
-d [ --debug ]                       debug output
-o [ --output ] arg (=gui)           override output - format: gui, xml
-u [ --url ] arg (=127.0.0.1)        override the listening address
-p [ --port ] arg (=29999)           override the listening port
-b [ --browser ] arg (=firefox)      broswer (include path if necessary)
-l [ --libName ] arg                 library name
-c [ --cellName ] arg                cell name
-v [ --viewName ] arg                view name
-i [ --option ] arg                  override option-format: name=value
-r [ --read ] arg                    read fileName
-w [ --write ] arg                   write fileName
-e [ --regex ] arg                   regular expression
-t [ --tech ] arg                    tech
-s [ --session ] arg                 session
--lcv arg                            lib/cell/view name
--docroot arg (=./si2web)            override the doc. root directory
--libdefs arg                        libdefs directory
--libdeffile arg                     libdefs file
--browseropts arg                    broswer options
--parent arg                         parent
--dmd arg                            dmd
--dmdskip arg                        dmd skip
```

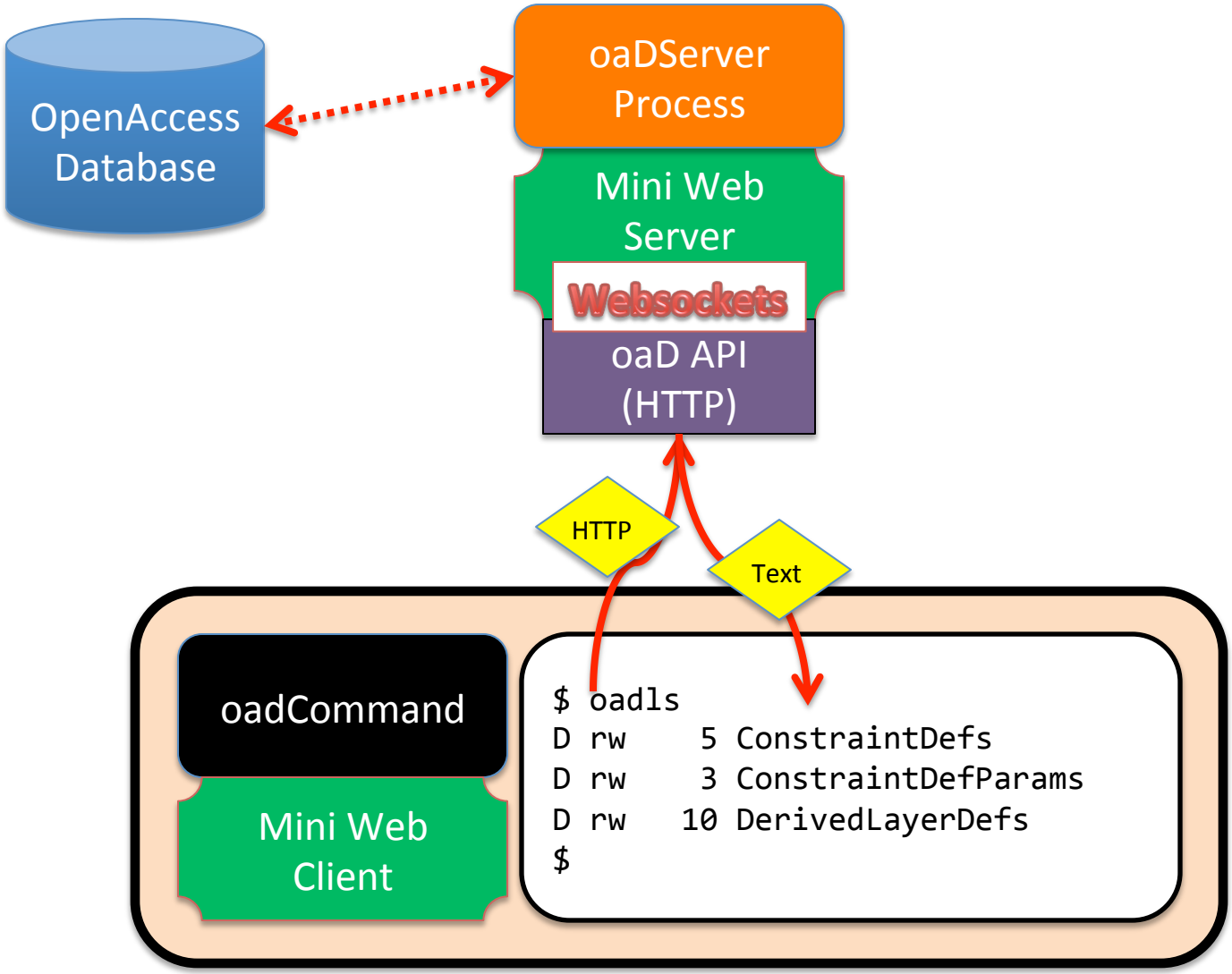
The New HTML5 Web Client



The oaDebug Extension



How oaDShell Works



oaDShell Programs

- oadstart – Starts the oaDServer process
- oadls – Lists the collections in the current location in the hierarchy
- oadcat – Outputs the properties of the OpenAccess object
- oadcd – Moves to the new location
- oadenv – Outputs the OpenAccess environment in use
- oadfind – Outputs the find results from the OA database
- oadhelp – Sends the user to the OA docs on the object
- oadpwd – Outputs the current location in the hierarchy
- oadtree – Outputs the tree hierarchy from the current location
- oadend – Ends the oaDServer process

oaDShell Programs – WHY?

The users shell has too many features that would have to be re-written.

Think about this….

```
$ for I in `oadfind . -type oaNet`;
  do
  if (( $(oadcat $I | grep "SigType: clock" ) == 1 ));
    then
      oadcat $I >> myClockNets.out
  fi
  done
```

What can you do?

- ✓ Join Si2's efforts in extending the coverage of the OpenAccess Tools by
  - ✧ Have your company join the OpenAccess Coalition
  - ✧ Have your developers join the Working Groups
    - ❖ ESG Working Group
      - ▪ oaScript Working Group
      - ▪ Polygon Operators Working Group
    - ❖ oaDebug Working Group

- ✓ Ask your EDA tool vendor to help support our efforts

- ✓ Go to http://www.si2.org for more information.

OPENACCESS COALITION MEMBERS

Current OpenAccess Member Companies

Advanced Micro Devices
Agilent Technologies
Altera
AnaGlobe Technology, Inc.
ANSYS, Inc.
Atrenta
AWR Corporation
Cadence Design Systems
D2S, Inc.
Dolphin Integration
Entasys Design
Fractal Technologies
Hewlett-Packard
Huada Empyrean Software Co. Ltd./ICScape
IBM Corporation
Intel Corporation
Invarian, Inc.
Jedat
Lattice Semiconductor Limited
MatrixOne
Mentor Graphics Corporation
Micro Magic, Inc.
Micron Technology, Inc.
Oracle
PDF Solutions, Inc.
Pulsic Limited
Qualcomm
R3 Logic, Inc.
Samsung Electronics Co., Ltd.
Silicon Frontline Technology
Silvaco Inc.
Synopsys
Taiwan Semiconductor Manufacturing Company Limited
Tanner EDA
Teklatech A/S
UMC
Zuken, Inc.

OPENACCESS REMINDERS!!!

Don’t Forget Si2 Website

One stop shop for all OpenAccess resource

http://www.si2.org/?page=1181

Questions?

THANK YOU!