Introduction to Milkyway

EDP 2003 – Monterey, CA – April 14

Laurence Brevard
Milkyway R&D
Hillsboro, OR
brevard@synopsys.com

My Background >30 years

- 1967-8 **Rice Univ. (Fortran, real-time machine language)**
- 1973 **TI (test systems, digital simulation, minicomputers)**
- 1981 **CGIS / GE Calma Austin (Tegas simulator)**
- 1984 **MCC (VLSI CAD, DR & EDA standards)**
- 1995 **Full-time Consultant**
  PC/Unix integration, EDA standards, networking – including Internet connectivity and applications
- 1997 **Motorola (unified EDA, Formal Verification)**
- 1999 **MediSpecialty.com (medical community)**
- 2001 **Avant! / Synopsys (opening Milkyway)**

Milkyway Database History

- Developed under unusual circumstances
- Used since 1998
- Physical Design and Netlist
- Used by many physical design tools:
  - Apollo, Astro
  - Enterprise, Cosmos, Hercules
  - Jupiter, Star-RCXT
- Proven in thousands of tape-outs

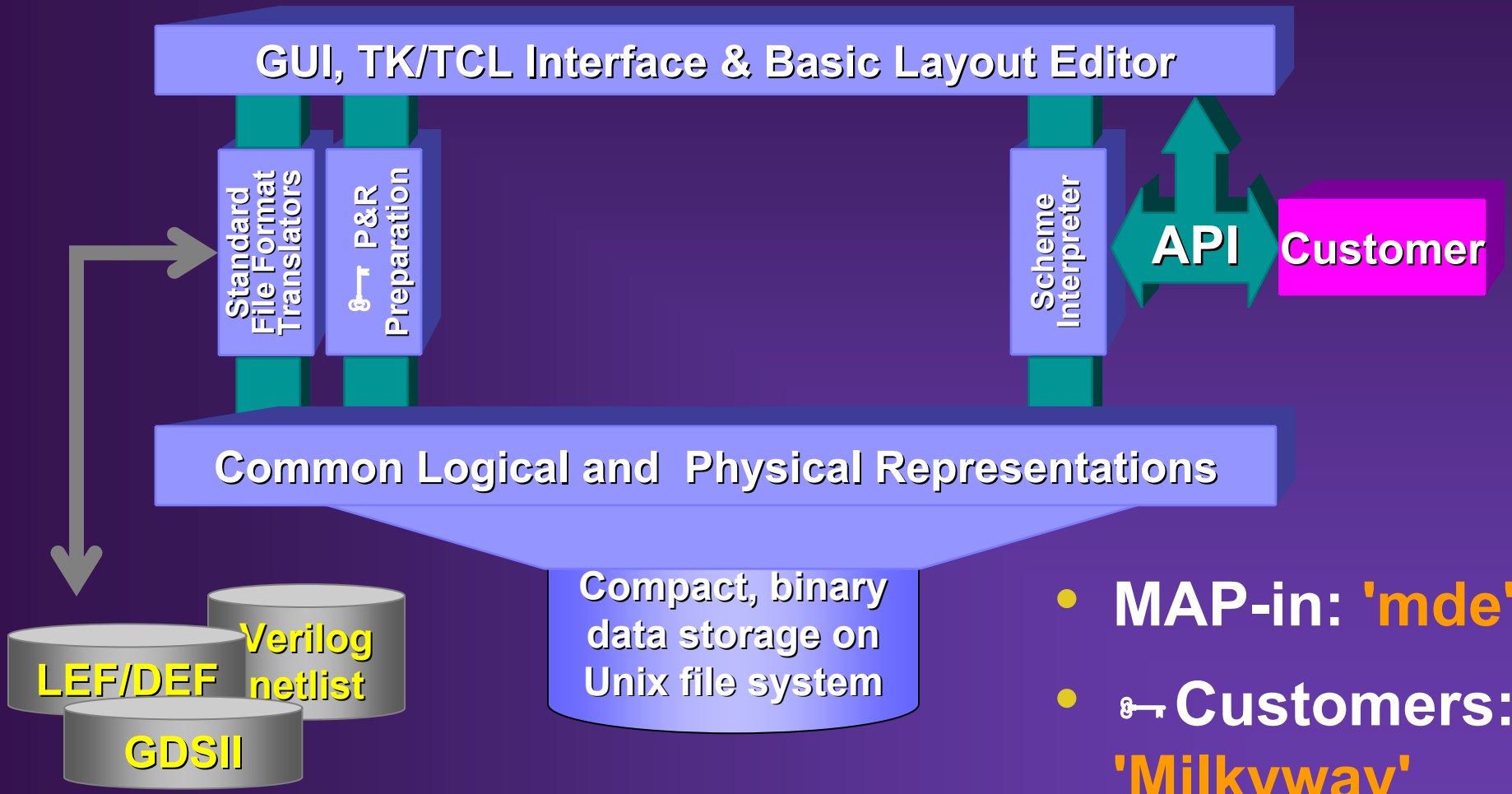
What is Milkyway?

- **Persistent Database – on disk**
  (design data storage system)
- **Objects and operations**
  - **available to tools developed by Synopsys**
  - **seen and manipulated via Scheme**
  - **seen and manipulated via C-API**
- **Environment for tool and utility operations**
- **The heart of the "Galaxy" system to be used to integrate all Synopsys front-end / design tools**

MAP-in Program for EDA vendors

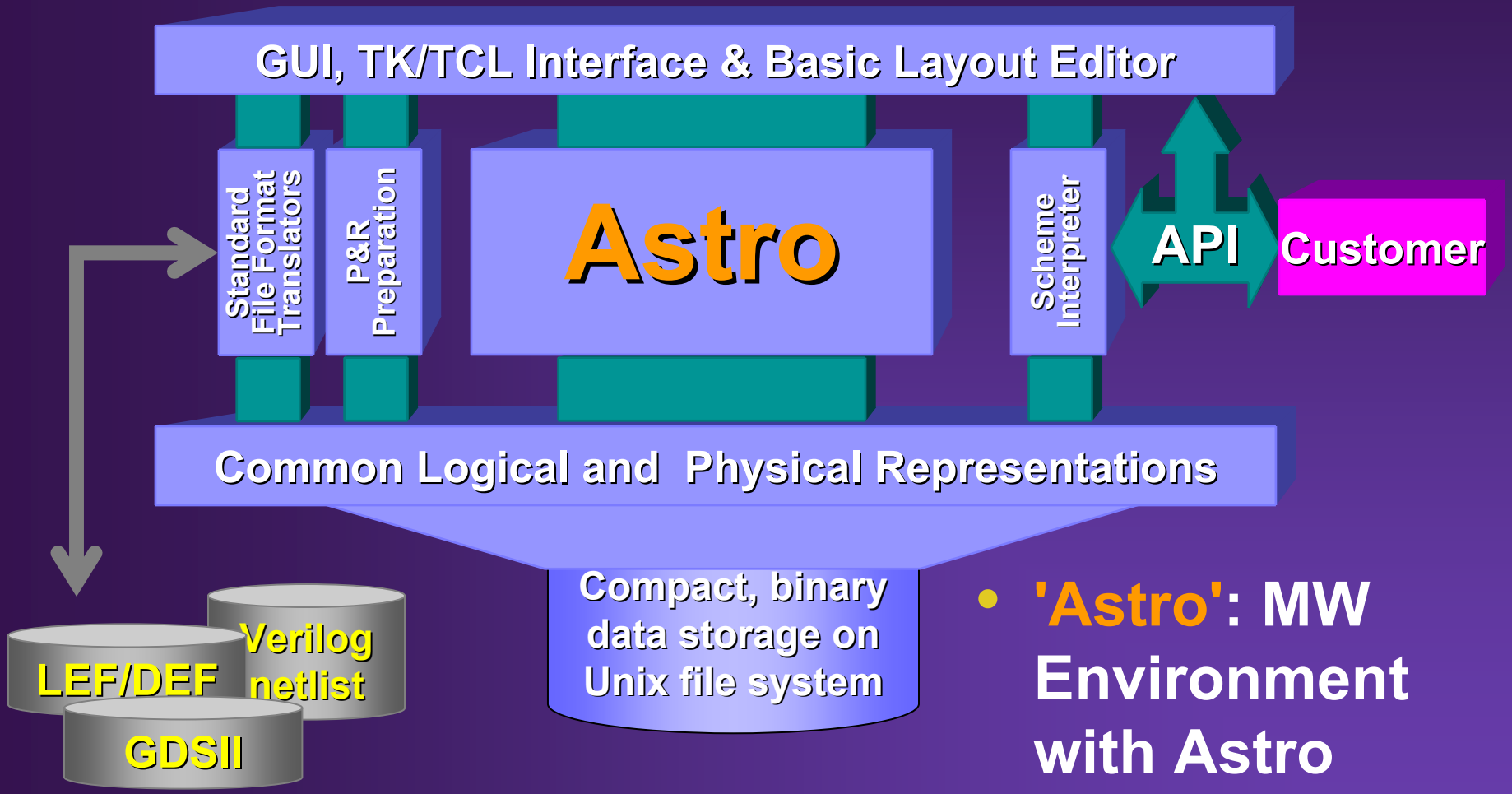
- Basic Membership Requirements
  - Commercial software tool vendor that needs access to Milkyway-based data
  - Agree to MAP-in license
- No Membership Fee
  - First copy of Milkyway Database Environment (MDE) product at no charge.
    - Additional copies of software available for modest annual fee
  - Access to SURF on available/approved basis
  - MAP-in web forum for community support
- Synopsys offers optional support contract

Milkyway Database Environment



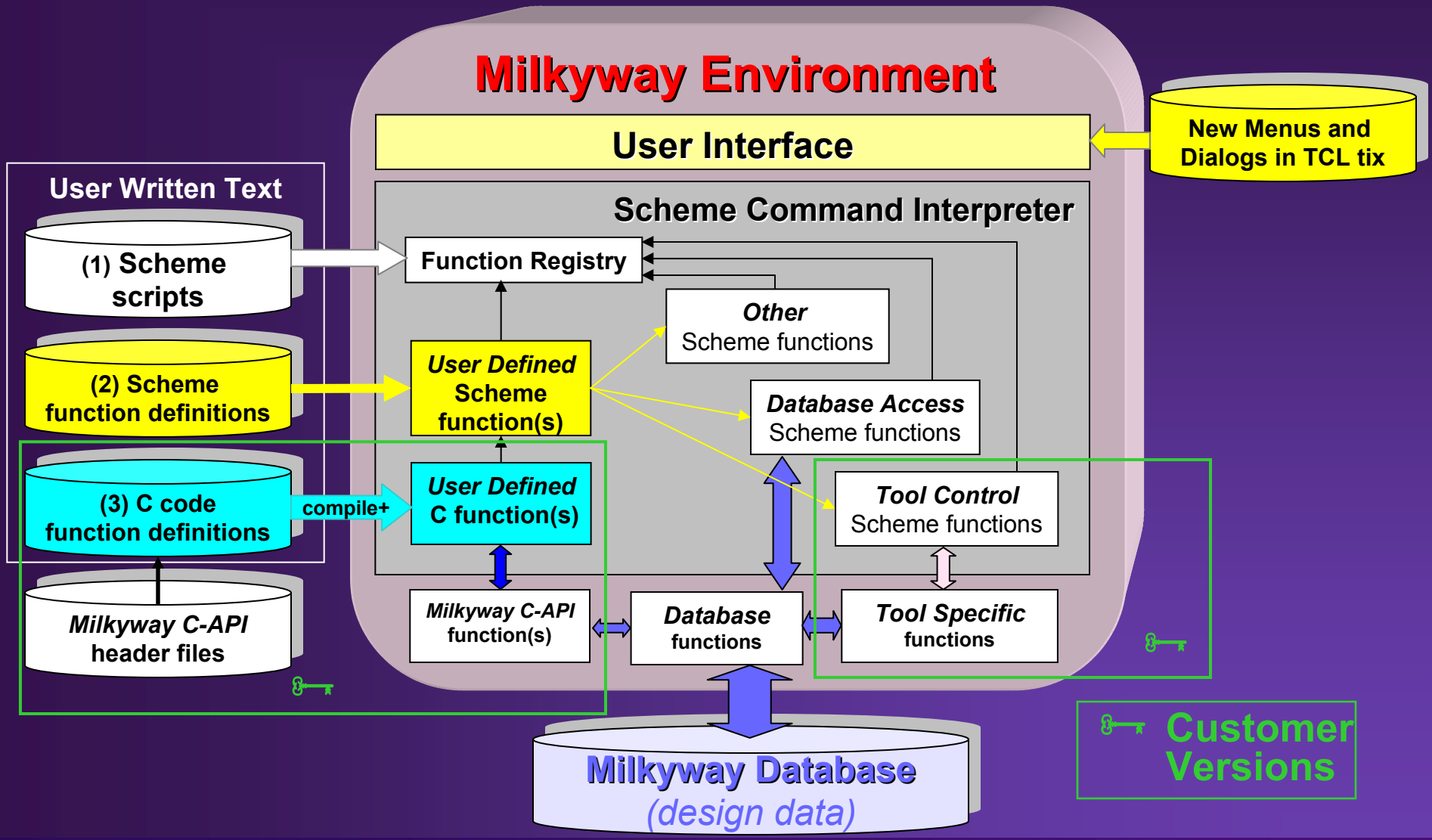
- MAP-in: 'mde'
- Customers: 'Milkyway'

Milkyway Environment – with Astro



- 'Astro': MW Environment with Astro commands

Integrated – Extension Mechanisms



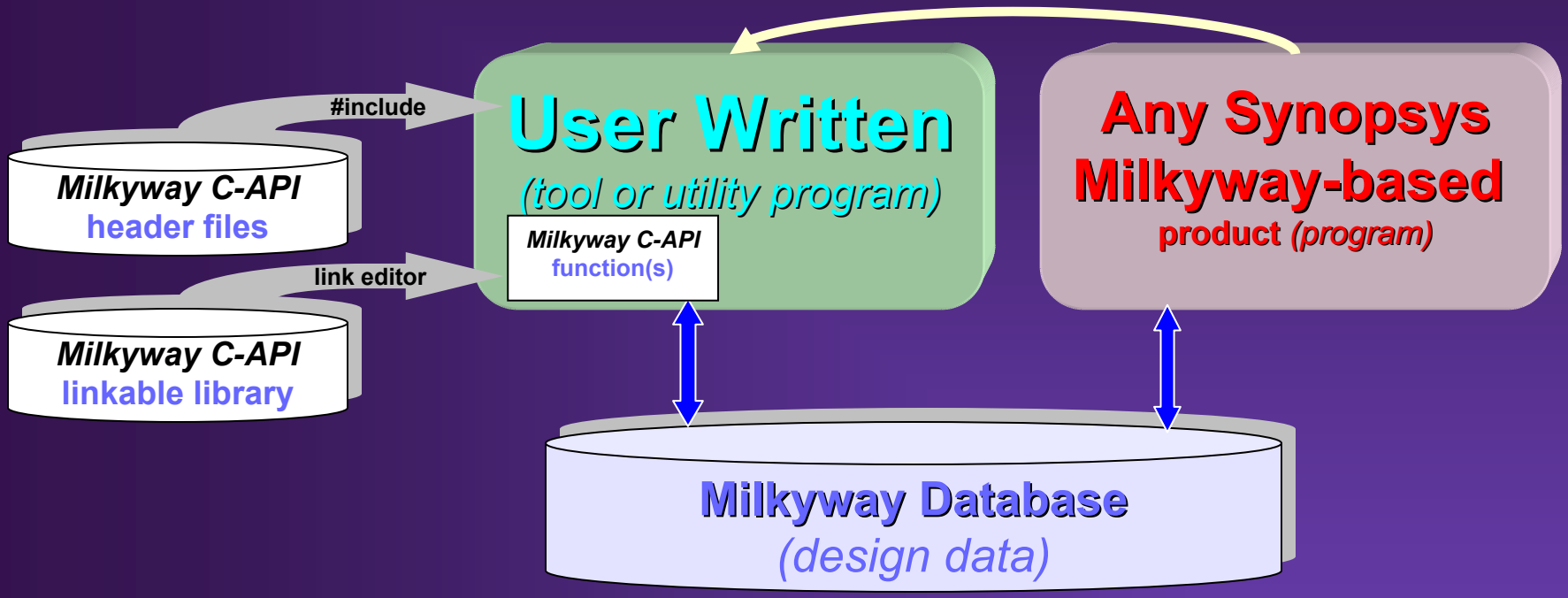
Milkyway C-API Origins / History

- Done originally to accelerate Scheme extensions
  - by replacing the inner loop of a routine with C
- Allowed "fancier" extensions than in Scheme
  - use of complex C run-time structures
  - use of C++
- But… it required user to integrate their code into the Milkyway-based program
- Appropriate for big customers extending our tools – in spite of support costs
- Not appropriate for 3rd party integration

⇒ We created a version for linking to user's stand-alone programs to allow access to Milkyway data.

C-API stand-alone Usage Model

If appropriate, the user written program can be called from the Milkyway environment using Scheme (system …) command



C-API for stand-alone applications

- Built from the EXACT same code base as included within the Milkyway environment
- Delivered as a linkable library: `libOMWX.a` instead of inside Synopsys Milkyway-based products
- You "own the main()" of the program using it.
- The `mwx_proto.h` header file defines the functions available to the stand-alone program.
- Only Milkyway data-access functions make sense in this context.

MAP-in and Customer Access

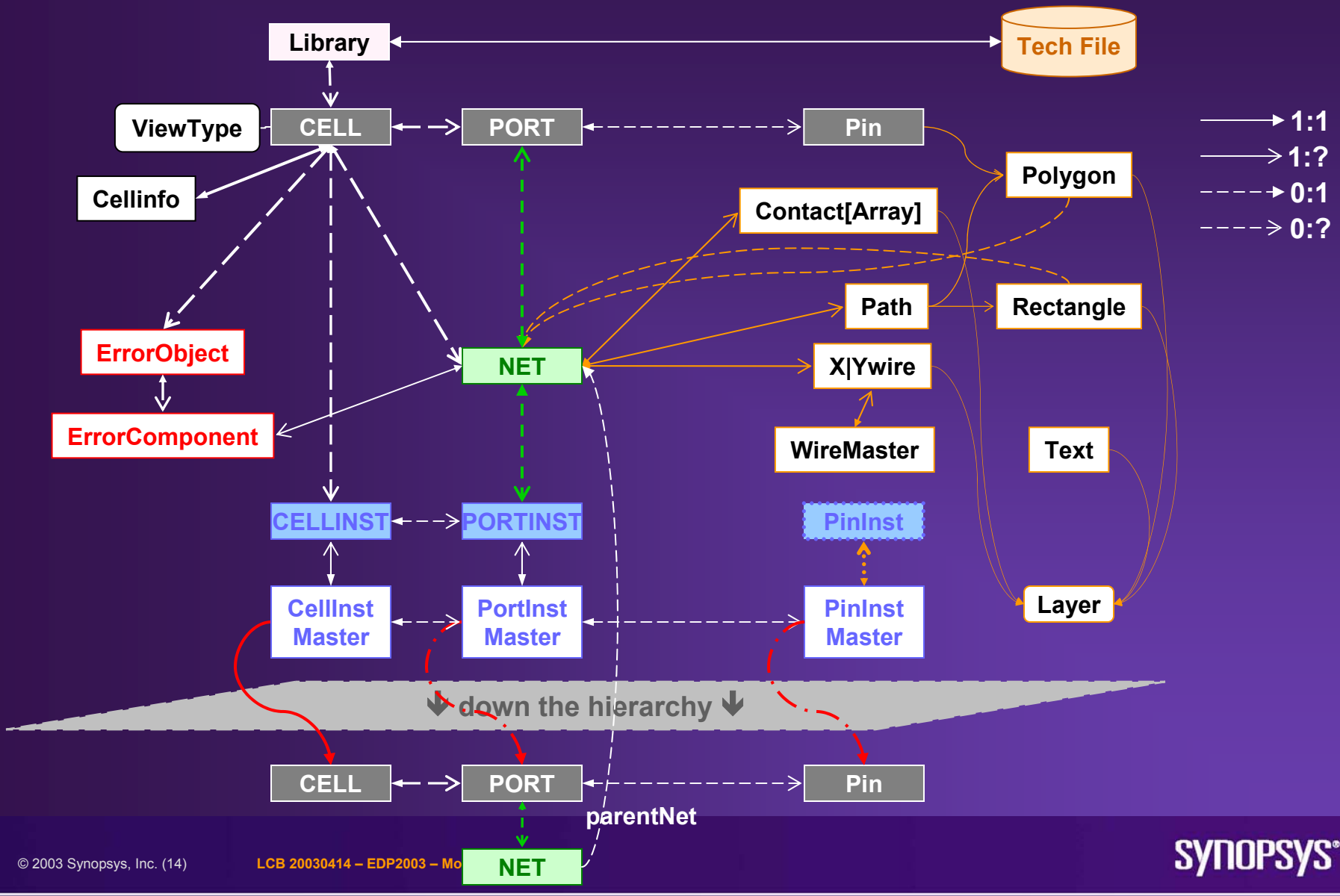
	MAP-in /mde	Customer
Scheme / TCL – tix scripting	✓	✓
Data Translators in and out	✓	✓
C-API Access to Database	✓	✓
Scheme Commands for Tool Functions		🔑
C-API Access to some Tool Functions		🔑
Integrated C-API (creation of new Scheme fcns. and data passing between Scheme and C)		🔑

Milkyway Objects Overview

- Categories of object types:

Designs and Libraries	Connectivity and Hierarchy	Physical and Place & Route
•Library •Cells •Views •Versions	•Ports •Nets •Cell Instances •Port Instances	•Pins •Contacts •Vias •Paths •Wires
Extensions	**Geometry**	**Technology**
•Property •Group •Attached File •New View	•Points •Rectangles •Polygons •Text •Layers	•Design Rules •P&R Rules •Units & Values •Capacitance

Milkyway Objects Overview *(cont)*



SYNOPSYS®

Milkyway Object IDs

- All objects are manipulated by opaque Object IDs
- No structures and structure member operators
- Two Types of Object ID
  - Non-persistent, not context sensitive *(file handle)*
    - Library ID – assigned when opened, lasts for session
    - Cell ID – does not require Library ID context
  - All others are persistent, context sensitive *(persistent index)*
    - Generally require Cell ID context to be meaningful
    - Stored on disk in the database
    - Specific value may change based on adds and deletes

Milkyway C-API Function Format

- MWXDb_ prefix
- Function return is Result Code
- Input Arguments First
  - Cell ID first argument to most functions
  - Other parameters
- Output Address(es) Last

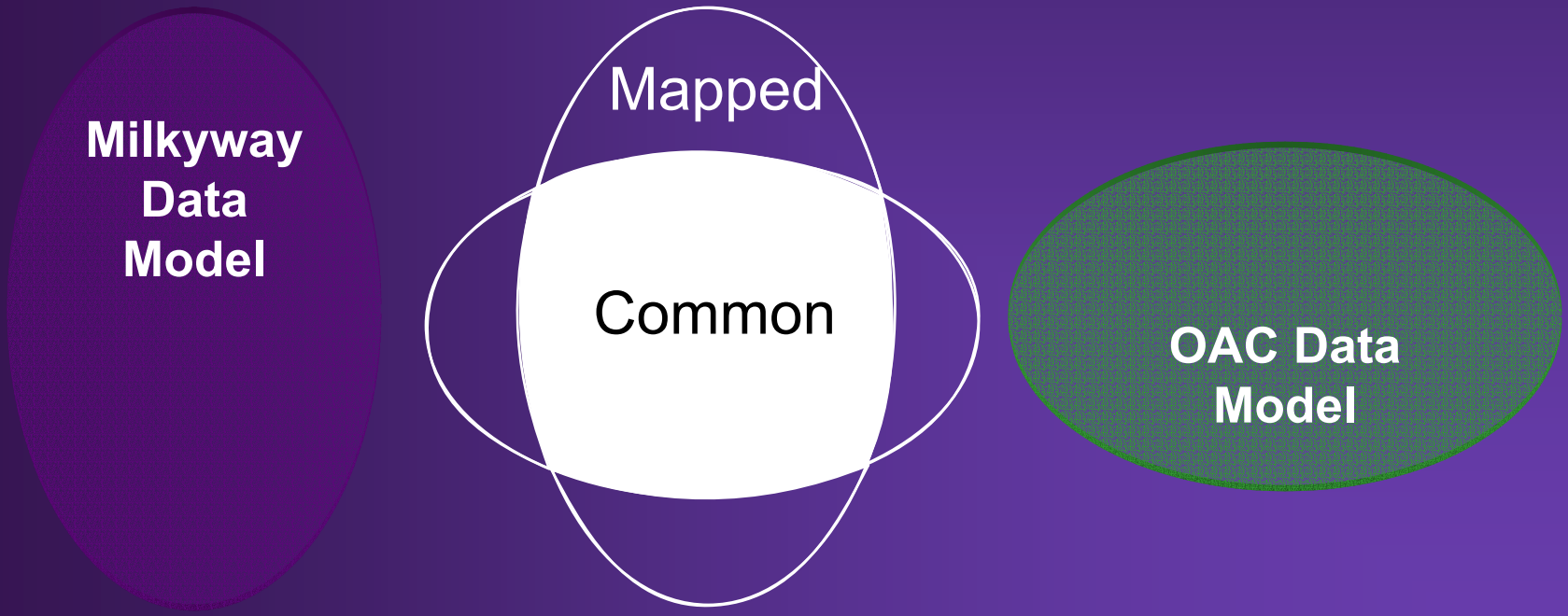
```
int32 MWXDb_Get_Net_ByName(
        MWXCellId_t      cellId,  /* input */
        char            *netName, /* input */
        MWXObjId_t      *netId);  /* output */
```

Milkyway evolution…

- Integration of pre-merger tools from Avant! and Synopsys
- Environment support for pure TCL scripting, initially for database access, later for tools in the environment
- Move to common file format translators with Milkyway as source / destination
- Continuing improvement in the infrastructure for capacity and speed
- Bridges to and from OpenAccess

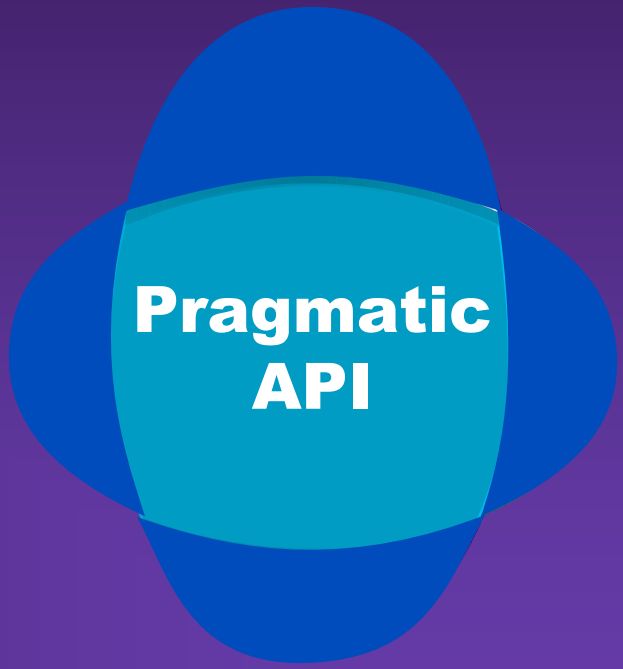
Define and Create Bridging Technology

- Initial effort to identify data model differences
- First deliverable expected to be a direct data transfer utility



Pragmatically Align APIs

- Large installed base of Milkyway-based tools requires stability of the Milkyway API
  - Synopsys
  - Proprietary, customer-developed
- Our customers strongly request that we respect and preserve their investment in Milkyway API
- Aligned APIs to be as similar as possible



Synopsys' Interoperability Commitment

- Open Galaxy platform via Milkyway Database APIs
- Preservation of customer investment in Milkyway
  - Stable APIs
  - Forward and backward data compatibility
- Working with Si2 via Golden Gate WG
  - Bridge between Milkyway and Open Access
  - Pragmatic API alignment long term

Golden Gate Activities thus far

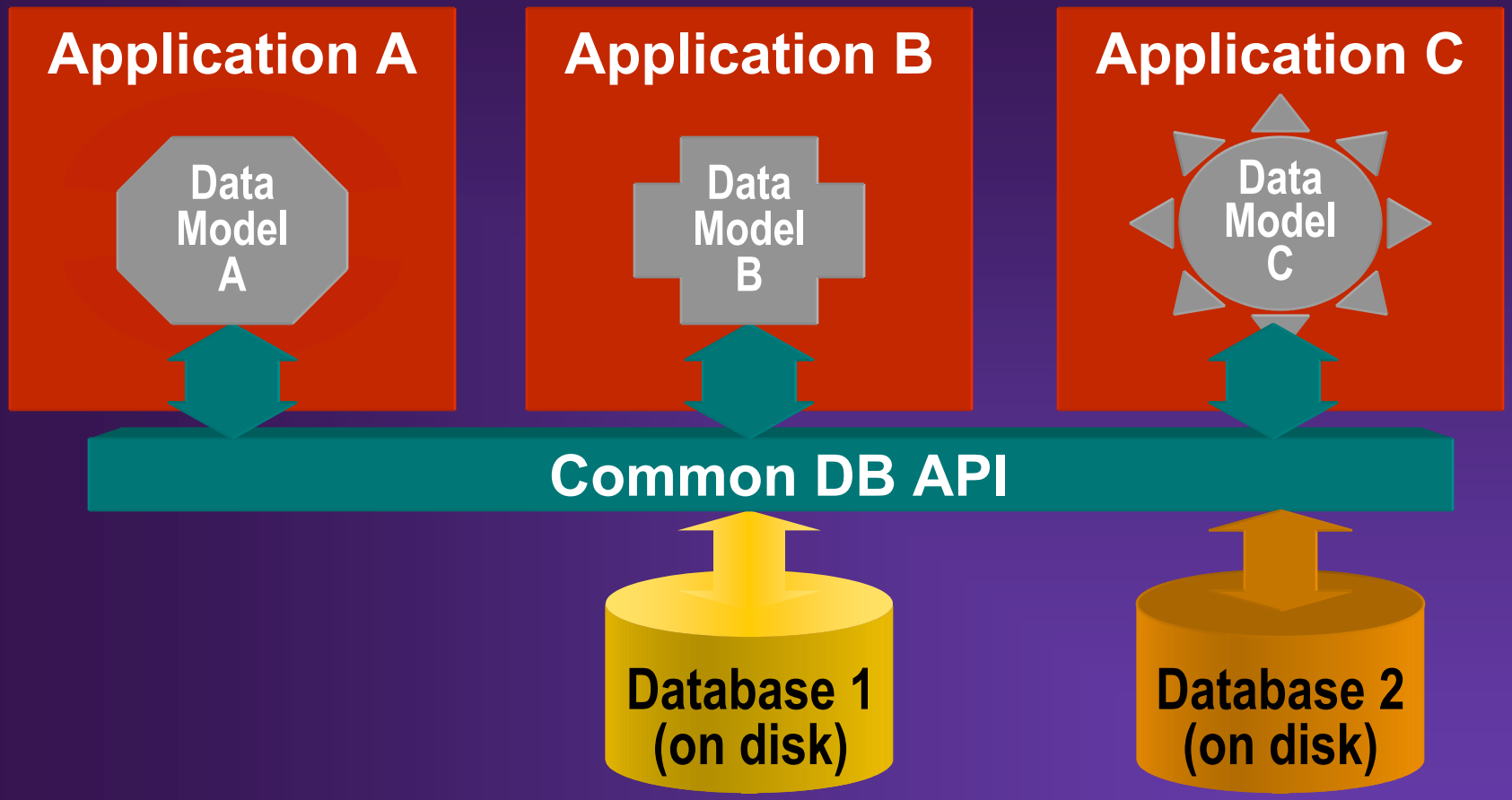
- Weekly Conference Calls
- Several one on one sessions between myself and Mark Bales – more useful (or at least more fun)
- Mapping Spreadsheets
  - First one has over 200 items – too much
  - New simplified one has fewer rows but more columns: general notes, open /close operations, access / traversal, creation, and deletion
- Semantics as defined by tool operations / expectations likely to be the hardest thing.

Some personal thoughts…

- Technical diplomacy is tough at times
- No one should try it unless they have worked for (or at least with) >1 company
- Talk about a common (OA) API with more than one implementation is *mostly* fantasy
  - How do you "hook" to more than one implementation?
  - See the following diagrams…

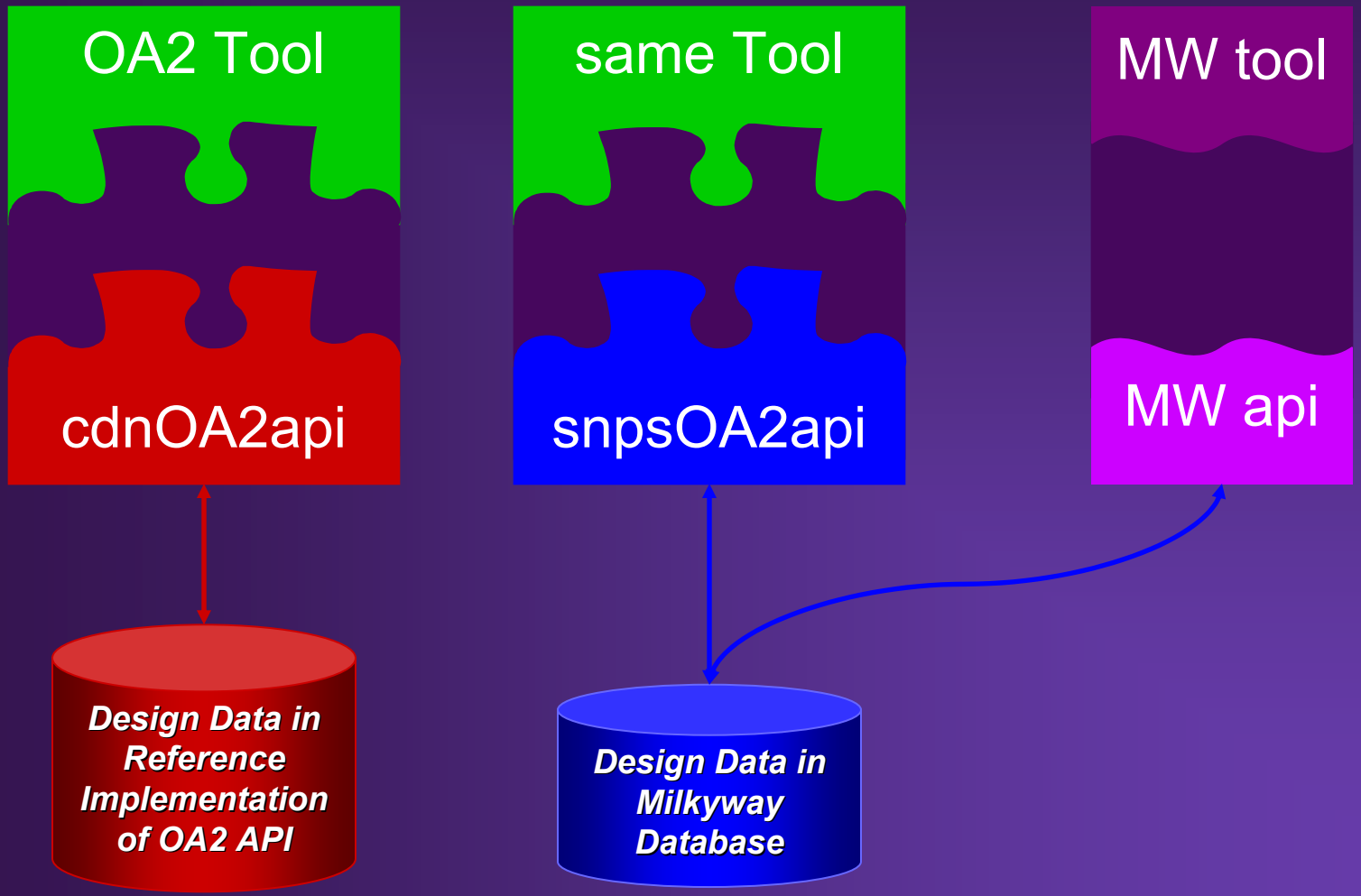
DISCLAIMER – this is me *musing*, not Synopsys promising!

I saw a diagram like this recently…

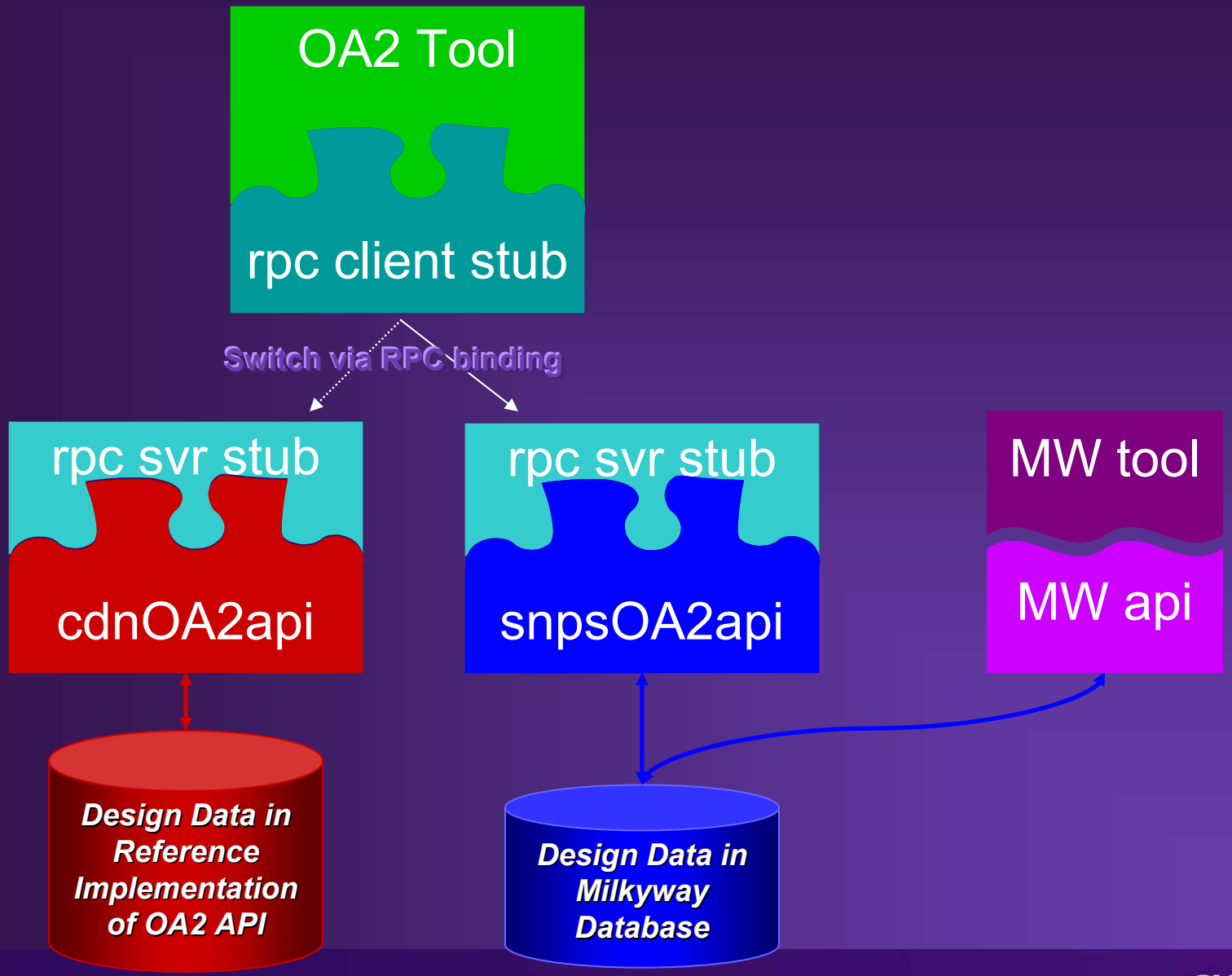


What does this mean???

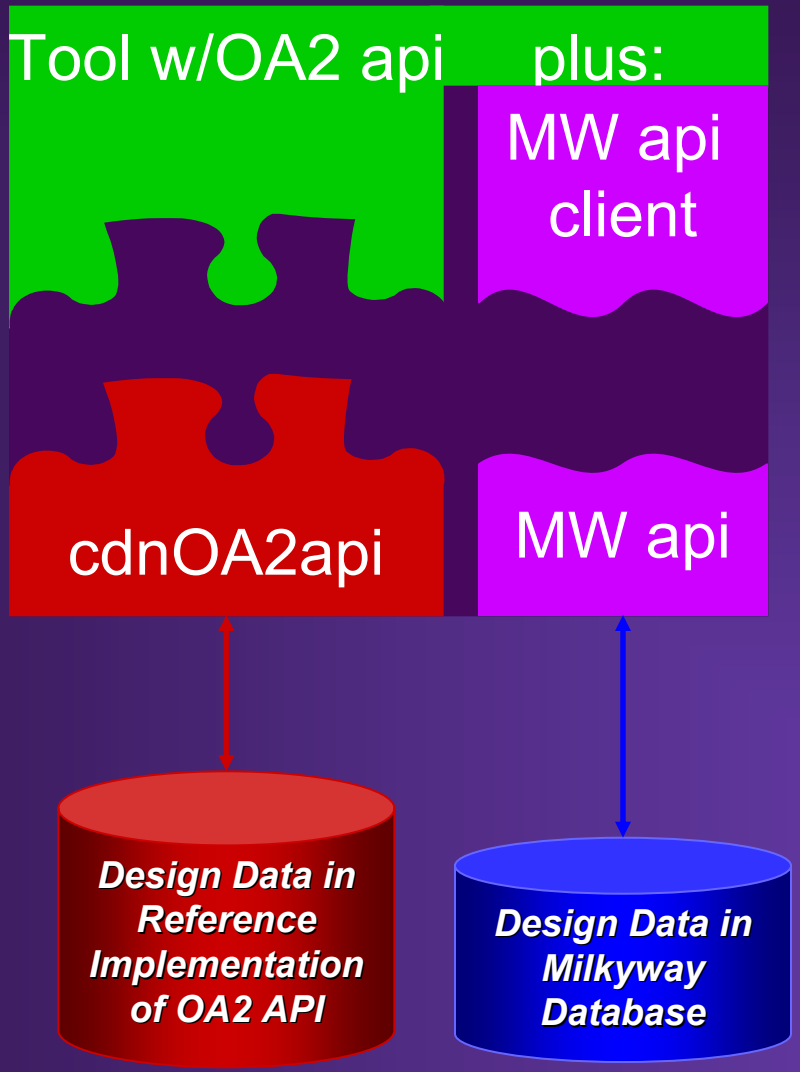
OpenAccess API on >1 Database?



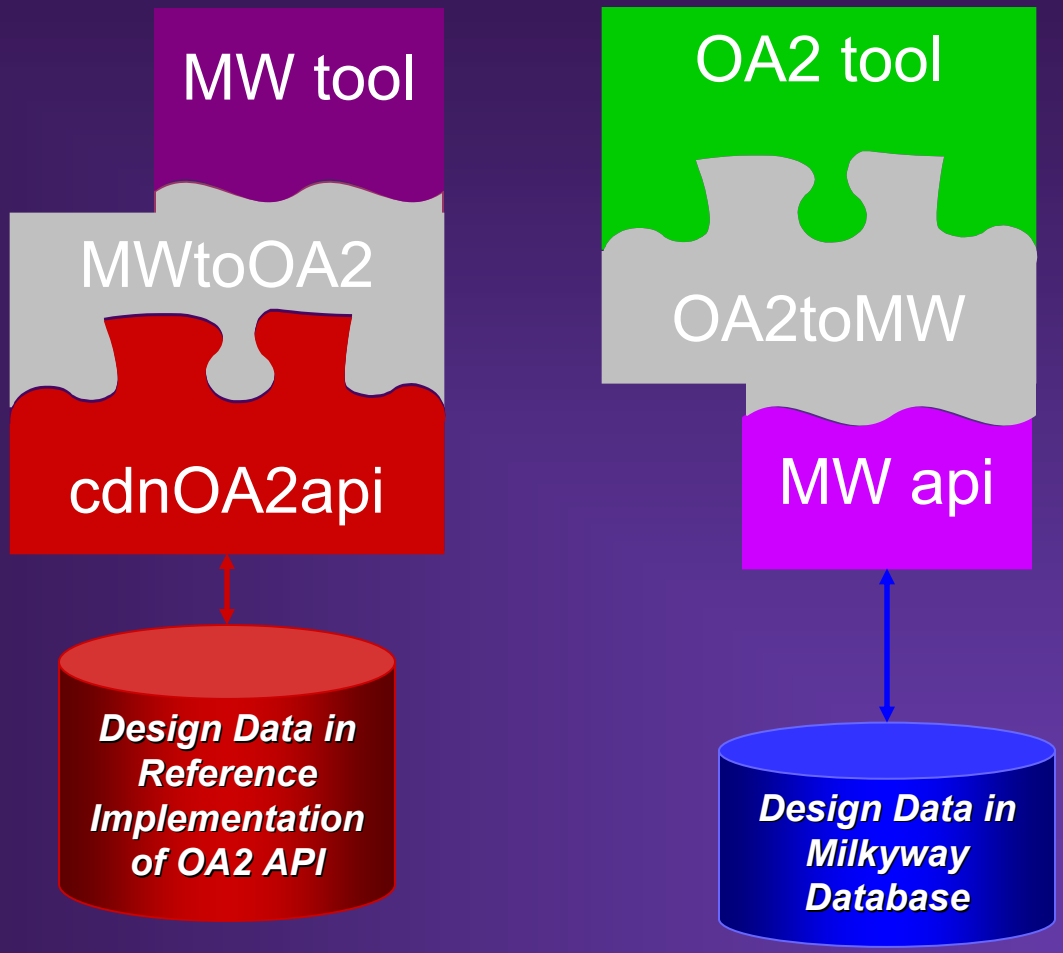
OpenAccess API on >1 Database w/ RPC



OpenAccess and Milkyway APIs



OpenAccess Milkyway Mapping APIs



My Conclusions

- Hope for interoperability springs eternal!
- Increased common understanding in the EDA community of the semantics of design data is essential.
- A publicly available world-class design database implementation with a clearly specified API can only help this process.
- An opened up Milkyway database will also help because it is (1) proven and (2) available today.
- Interoperability between the major EDA environments is essential for customer success.

Data Model, API, versus Database

- We need to have a *good enough* understanding of the information models to map or move design data between systems.
  - Allows better translators
  - Allows possibility of a mapping API
- Tools will always have their own internal run-time data structures (I do not call this a "data model").
- Unlikely to have a truly common API without…
  - A common database implementation, or…
  - A complicated implementation that understands more than one binary format on disk
- Near term APIs will be mapping APIs

The End

Thank you!