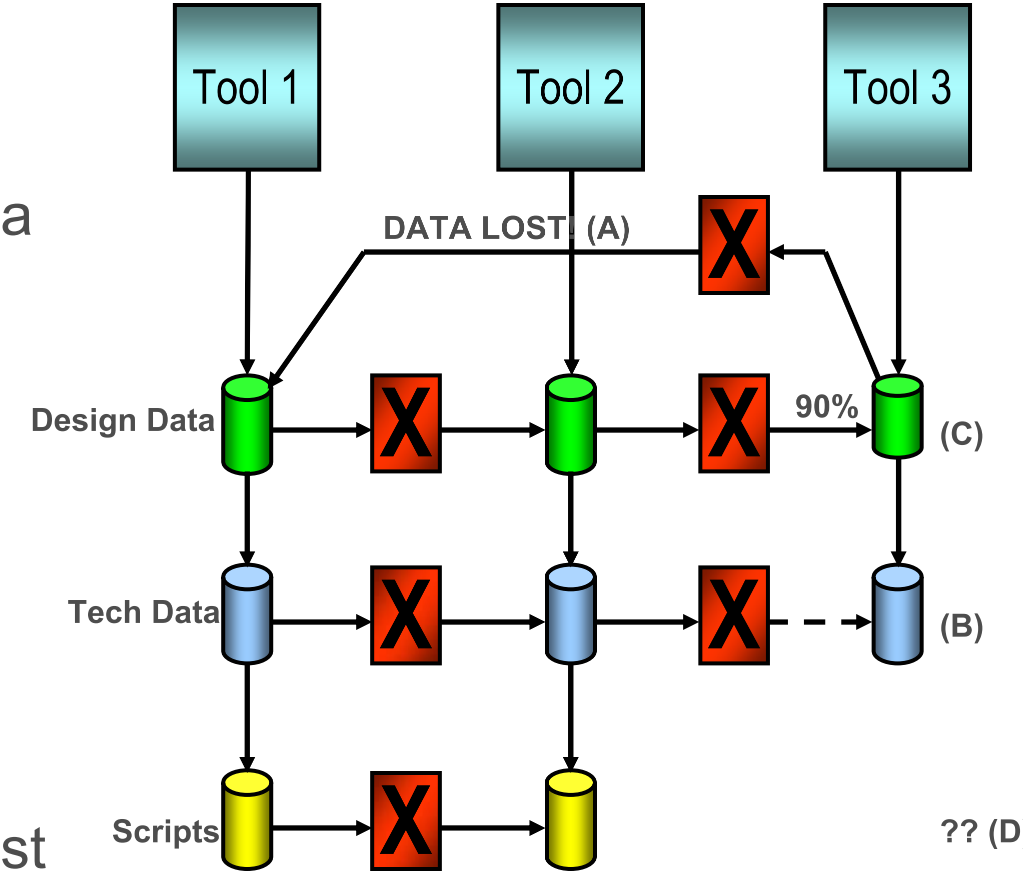
cadence

Facilitating EDA Flow Interoperability with the OpenAccess Design Database

EDP 2003
Mark Bales
14 April 2003

User Flow Requirements

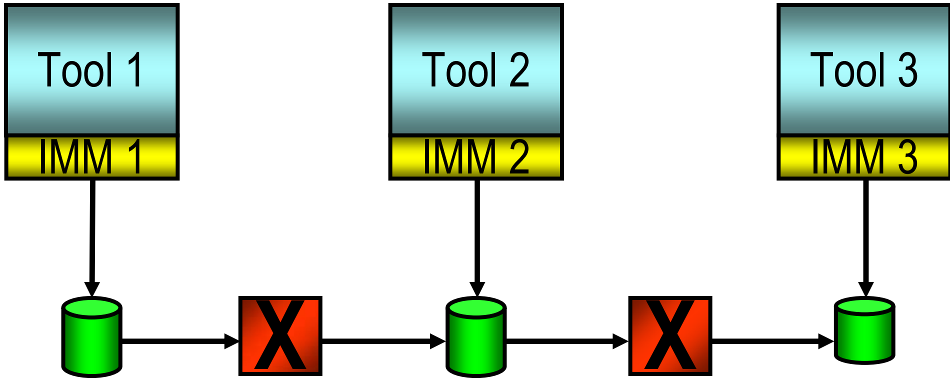
- Design data may be lost when you need to take data back to a previous flow step (A)
- Incomplete/inconsistent representation of technology data can require re-entry of data in multiple tools (B)
- Inconsistent representation of constraints may require re-entry in multiple tools (C)
- Scripts may not even be translatable from one tool to the next, meaning design intent is lost (D)



Application Developer Requirements

File-level Interoperability

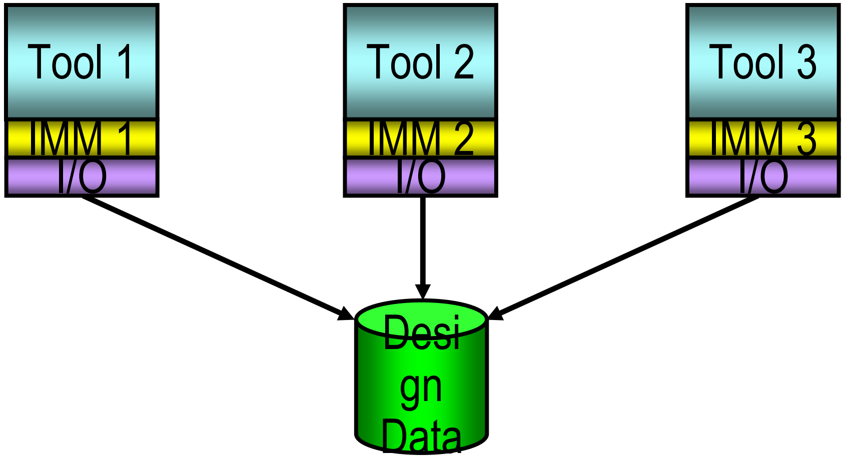
- Advantages:
  - Easier to effect independent tool release
  - Human-readable data storage
- Disadvantages:
  - Data usually much larger than a design database
  - Tools can become a “lossy filter”
- Suitability
  - Situations where output is different form from input
  - Situations where format is mature



Application Developer Requirements

Database as Interchange Format

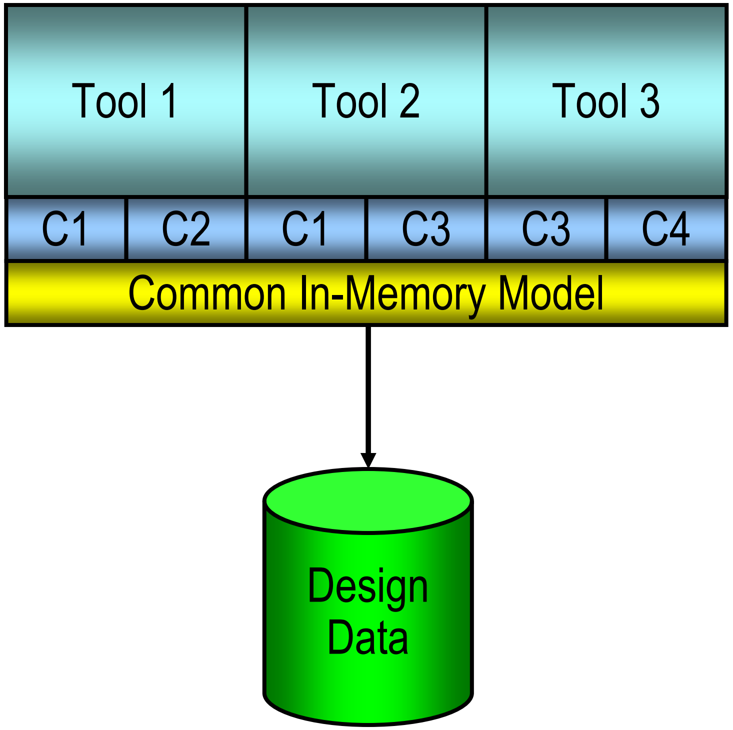
- Advantages
  - Smaller size of design db
  - More consistent semantic interpretation
- Disadvantages
  - A “thick” I/O layer causes performance problems
  - Not easy to share components among tools, or to build large systems
- Suitability
  - Tools where I/O time is small fraction of overall run time, and/or where there are few shared components



Application Developer Requirements

Common In-Memory Model

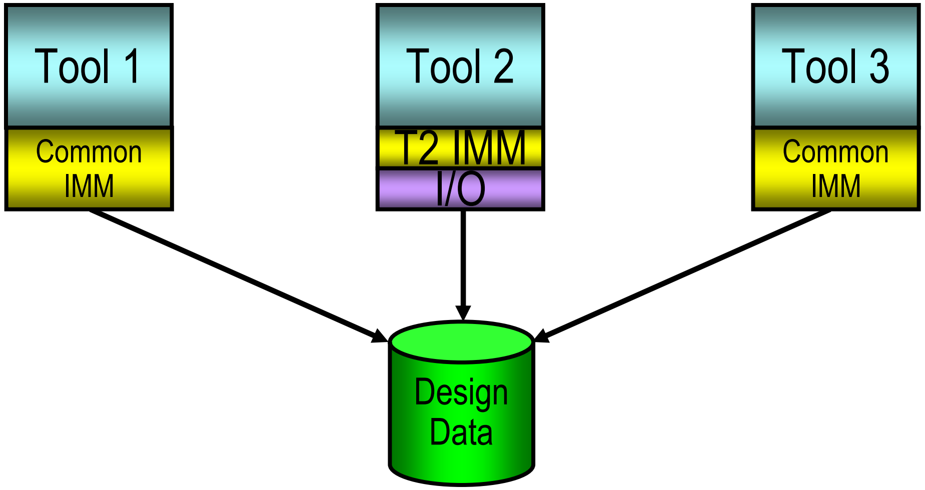
- Advantages
  - Shared components now possible
  - Possible to build large and/or standalone systems
  - Greatest chance of properly shared semantics
- Disadvantages
  - Evolution of data models slower than when tools/system is decoupled
  - No general-purpose db can be as performant as a special-purpose db
- Suitability
  - Systems with many reusable components
  - Systems with need for frequent tool data exchange



Practical System Architecture

In real world, a mix is best

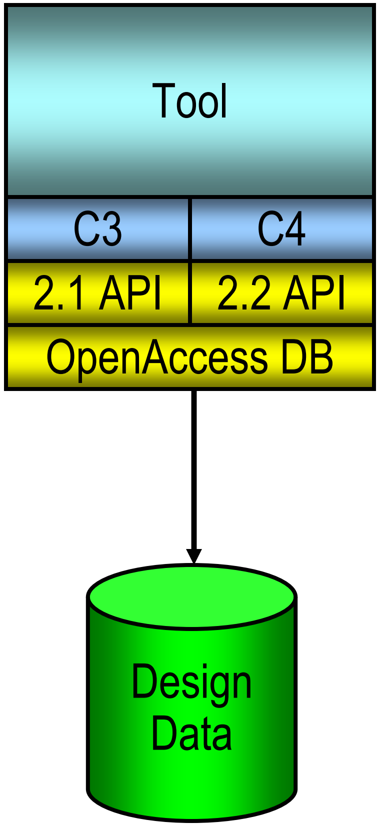
- Use common in-memory model:
  - For subsystems with many components
  - Break tool suites at appropriate places
- Use I/O layer:
  - When I/O is a small fraction of the overall time
  - When there is a highly specialized set of data structures in the tool



This mix is perfectly acceptable

OpenAccess

- Supports File, I/O Layer, and In-memory interfaces
- Has Full Extensibility
  - Prevents stagnation; provides “escape hatch”
- Has shared control with standardized versions
  - Prevents a single company from controlling evolution
  - Provides for a better end result
  - Provides for migration from one version to the next
- Rapidly evolving standard with new capabilities each release:
  - Technology advances
  - Logical/Physical mapping with occurrence model
  - 65nm support; X support; UDM support coming

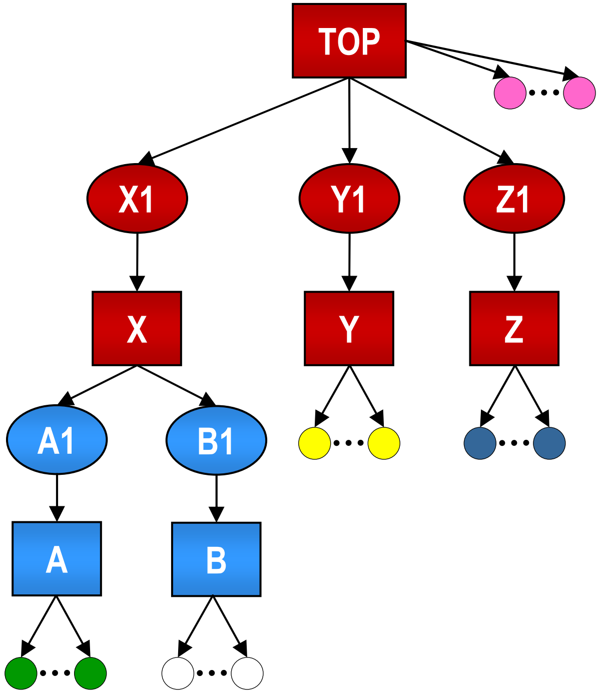


Why OpenAccess?

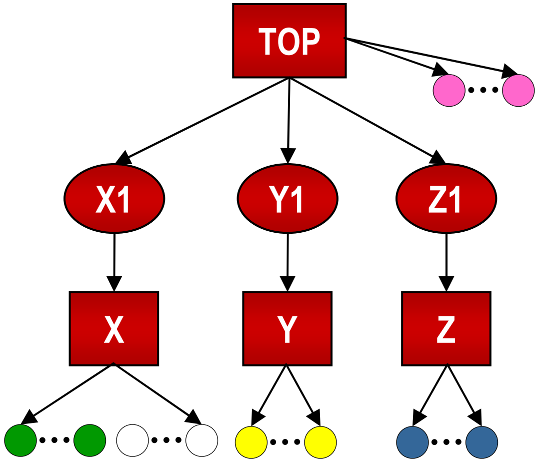
- For the record, any good design database can be made to work
- OpenAccess is:
  - A state-of-the-art, C++, new implementation of very mature ideas
  - Shared control, but freely available open source
  - Full of capability (over 3000 methods and growing)
  - A place for industry-wide standardization and sharing
  - Allows for proprietary and prototype extensions
- OpenAccess is not:
  - Limited to a single set of tools
  - Stagnant
  - A typical “committee” standard
- Find out more at http://www.openeda.org

OA 2.1 Work: Embedded Module Hierarchy

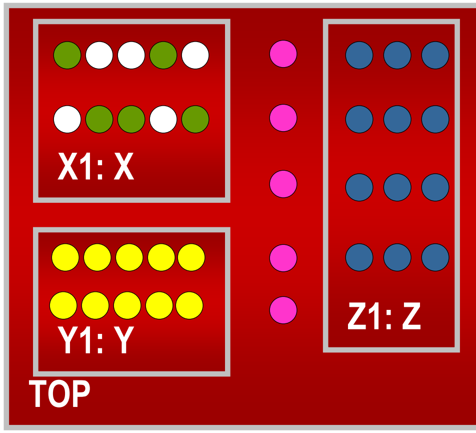
Module / Occurrence Domain



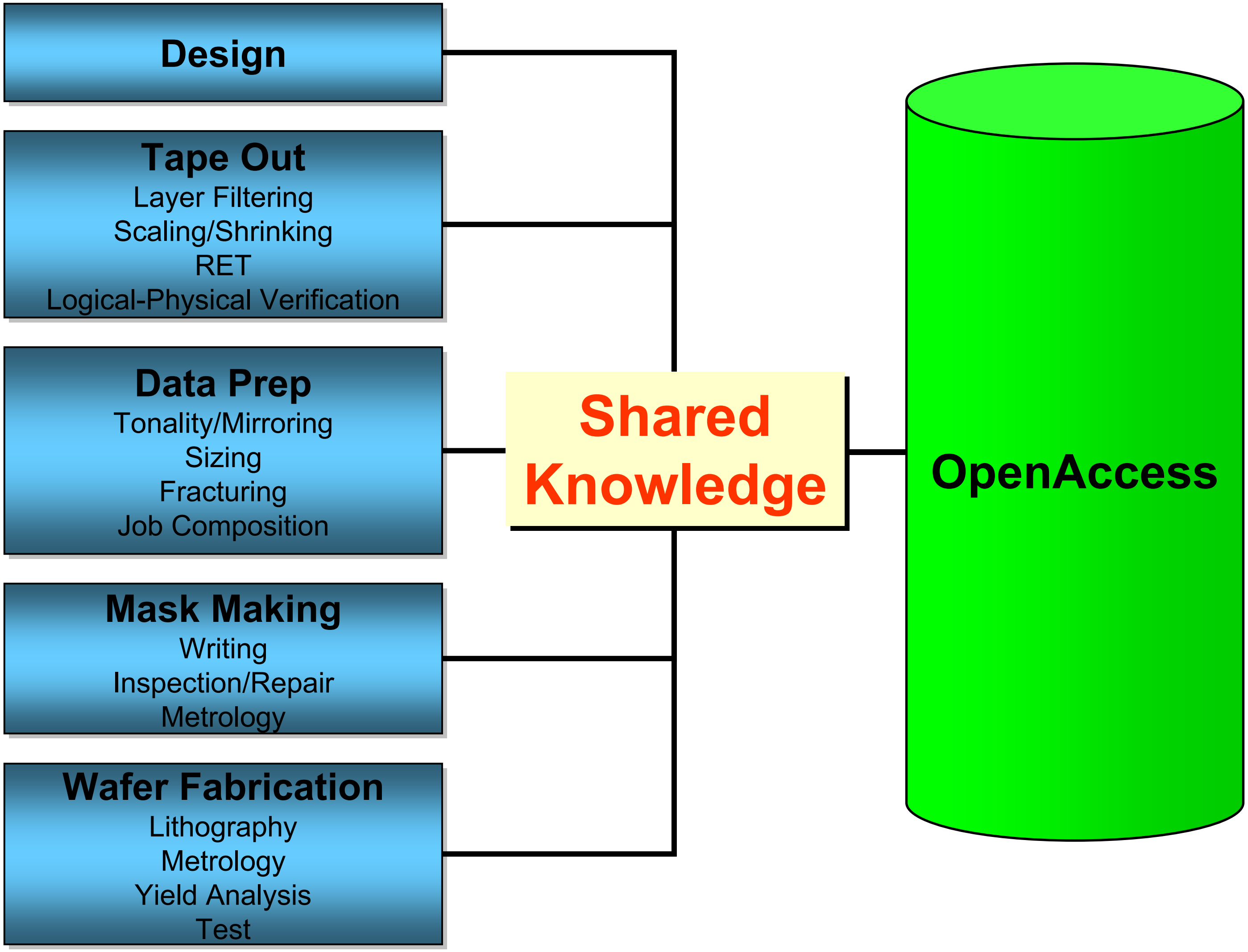
Block Domain



Layout Perspective



Future Work: UDM/Manufacturing Support


Design
Tape Out
Layer Filtering
Scaling/Shrinking
RET
Logical-Physical Verification
Data Prep
Tonality/Mirroring
Sizing
Fracturing
Job Composition
Mask Making
Writing
Inspection/Repair
Metrology
Wafer Fabrication
Lithography
Metrology
Yield Analysis
Test
Shared Knowledge
OpenAccess