E.D.P. 2003, Monterey, CA.

Interoperability, Datamodels, and Databases

Michael A. Riepe

Glossary

- **Data Model**
  - The native in-core data structures used by an algorithm
- **Database**
  - Client/server disk or memory based data repository
  - (non-native)
- **Application Programming Interface (API)**
  - How software modules interact with data model or database
- **Standard File Format**
  - ASCII or Binary file representation of a data model or database

Why all the fuss about data storage?

- **There is no such thing as a best-in-class point tool, only a best-in-class flow**
- **“It’s the flow, stupid!”**
  - (My apologies to Bill Clinton…)
- **EDA = Flow + Data model + Algorithms**
  - (If you squint)

Nvidia Example: Adding more engineers to deal with complexity

Design Start	Technology node	Transistor count	Complexity	Front-end staff	Back-end staff
1993	0.5μ	0.75M	1x	1.0x	1.0x
1995	0.5μ	1.25M	1.5x	1.2x	3.0x
1996	0.35μ	4.0M	4x	1.6x	3.0x
1997	0.31μ	7.5M	7x	1.7x	4.0x
1998	0.25μ	9.0M	10x	1.5x	4.0x
1998	0.22μ	22M	20x	2.5x	5.0x
1999	0.18μ	25M	22x	1.5x	4.0x
1999	0.15μ	57M	30x	3.5x	6.0x
2000	0.15μ	60M	35x	1.5x	7.0x
2000	0.15μ	63M	40x	3.0x	7.0x
2001	0.13μ	120M	50x	5.0x	9.0x

Source: Nvida/Chris Malachovski DAC 2002

Cannot Justify Traditional Flows for Today's Designs

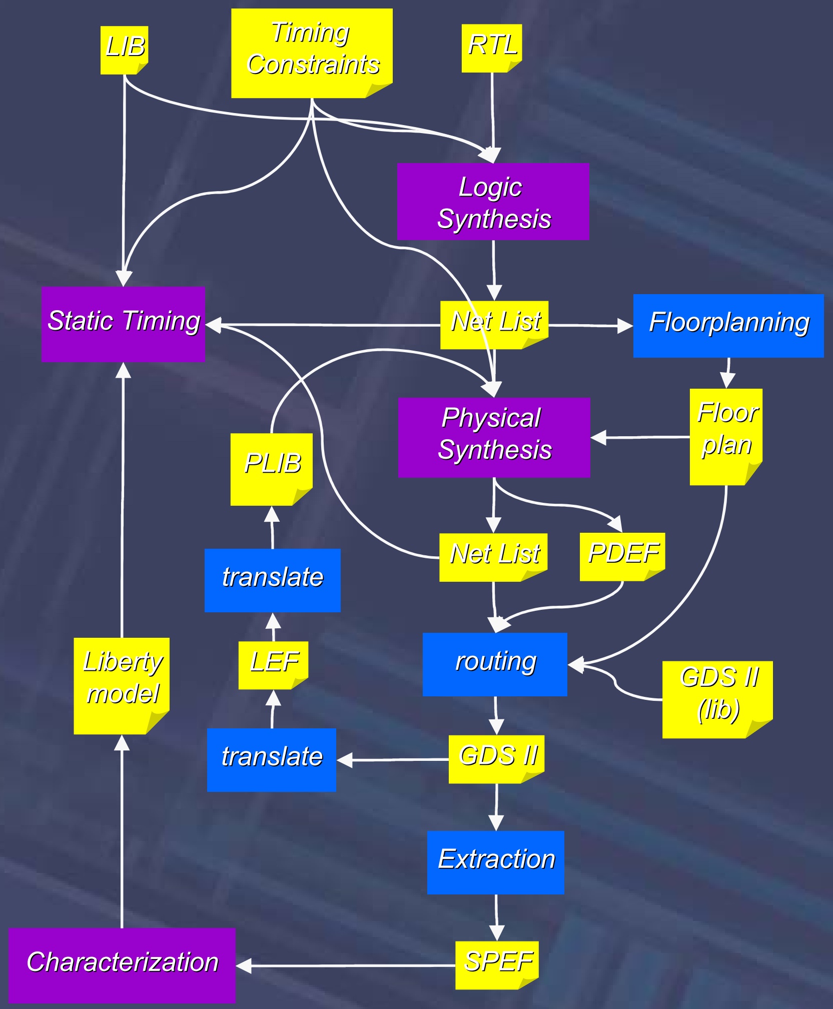
- **40 M gate designs**
  - 18mm X 18mm, 2000 I/Os, 500Mhz
  - Approximately 4M lines of RTL
- **50+ engineers**
  - Experts in synthesis, P&R, signal integrity, power analysis, design closure
- **$80MM investment**
  - Requires $160MM in 2 years to realize break even
  - Where is the killer application for this??

Traditional design flows will make Moore's law economically infeasible

Point Tools = Poor Productivity, Lower QoR

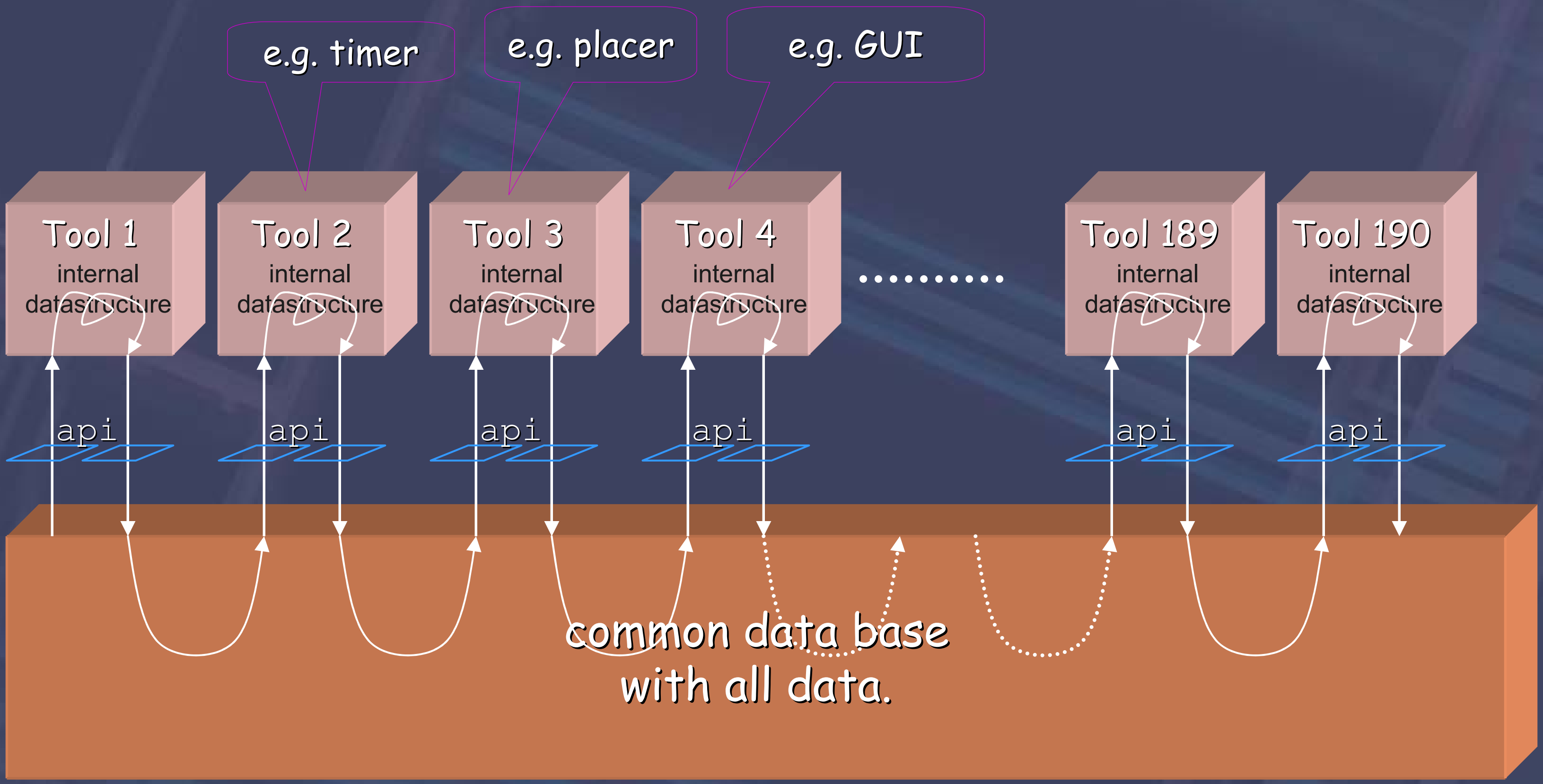
- **Point tools – architecturally inefficient**
  - Not scalable for nanometer effects: each new problem needs to be addressed by a new point tool
  - Requires wasteful iterations
- **Different timers, placers, libraries, constraint systems**
  - Correlation problems
  - Guardband to gain timing predictability → lower quality of results (QoR)
- **Huge file transfers cause waste of time**
  - Several hours of idle time for every iteration

‘Bolting’ together a flow using files



- **Design data is spread over many files**
  - Files are big and slow to read
- **Relevant information gets lost in the translation**
  - interpretation may not be consistent
- **DSM issues are *everywhere***
  - the file interface makes dealing with them harder

Tool integration through a 'Common Database'



- The database interfaces with the tools through an API
- Each tool makes its own copy of the data (data model)
- It takes memory and time to haul all this data around

MAGMA

Would you architect it like that if you could start from scratch?

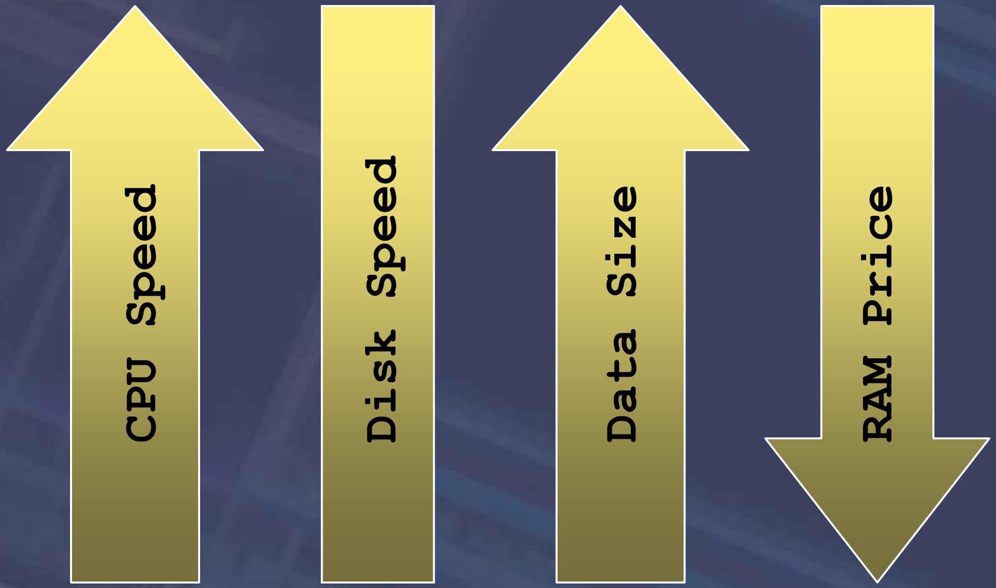
NO WAY!

instead:

- **Leverage similarity**
  - save implementation effort
  - reduce bugs
  - have consistency by construction
- **Minimize interfaces**
  - Tools spend a LOT of code reading, writing, and conditioning data.
- **Add incremental analysis tools (Timer, extraction, DRC) as part of the infrastructure. They are not point tools, they are part of the datemodel!**

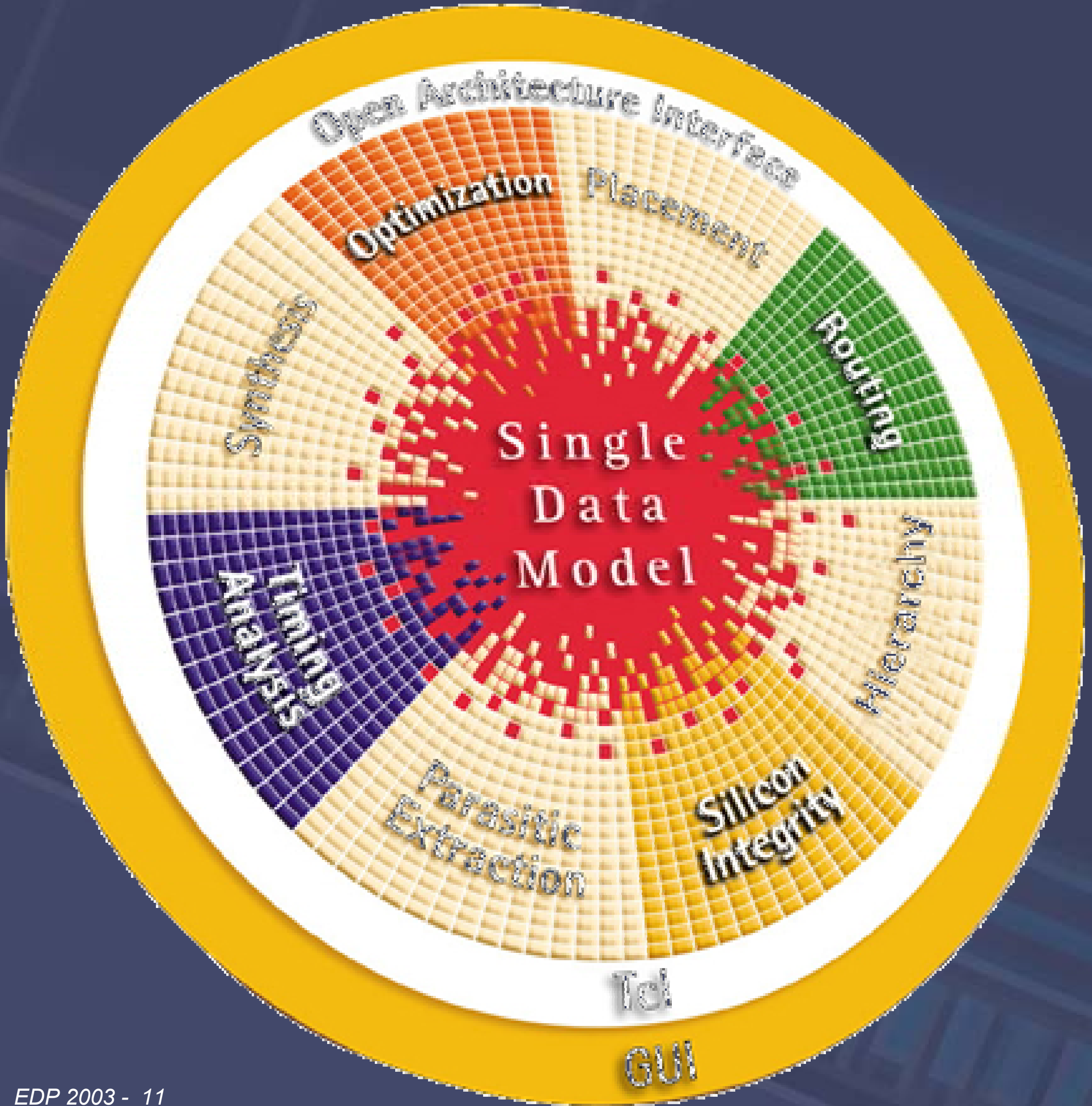
MAGMA

Why is it time for a paradigm shift?



- CPU speeds increasing *much* faster than disk speeds
- Design size growing exponentially
- Disk I/O dominates runtime!
- RAM is very cheap
- 64-bit CPUs enable unlimited data sizes

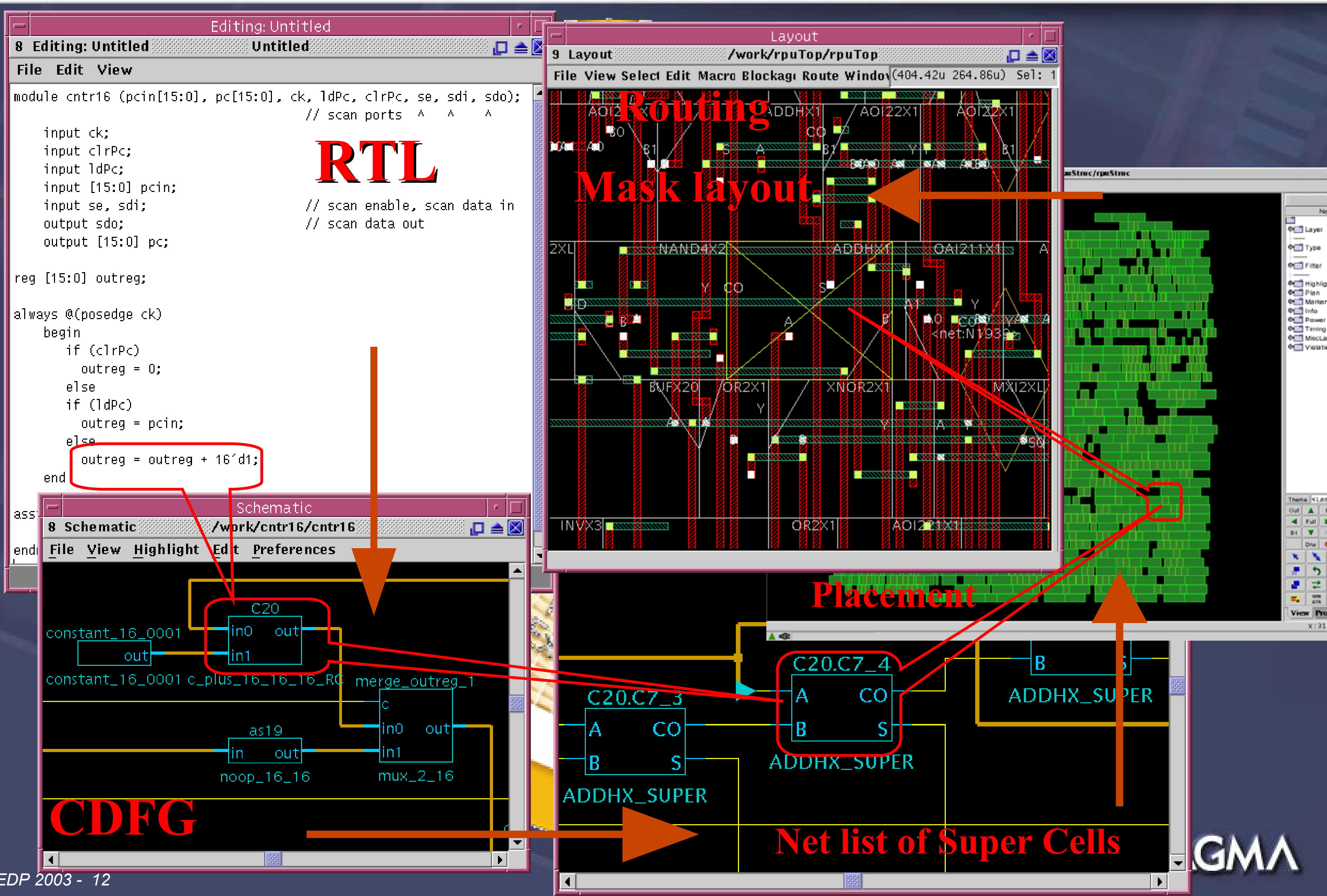
Magma’s Unified Datamodel



- **All tools share a common data structure. They run *directly* on it.**
  - Highest speed
  - Lowest memory overhead
- **All design data lives “in core”. The tools run *around* the data.**
- **Data model that contains *everything*:**
  - Complete net list, constraints, layout, library data, congestion, etc.
- **Analysis tools are part of the core**
  - timer, extractor, DRC, power, etc.

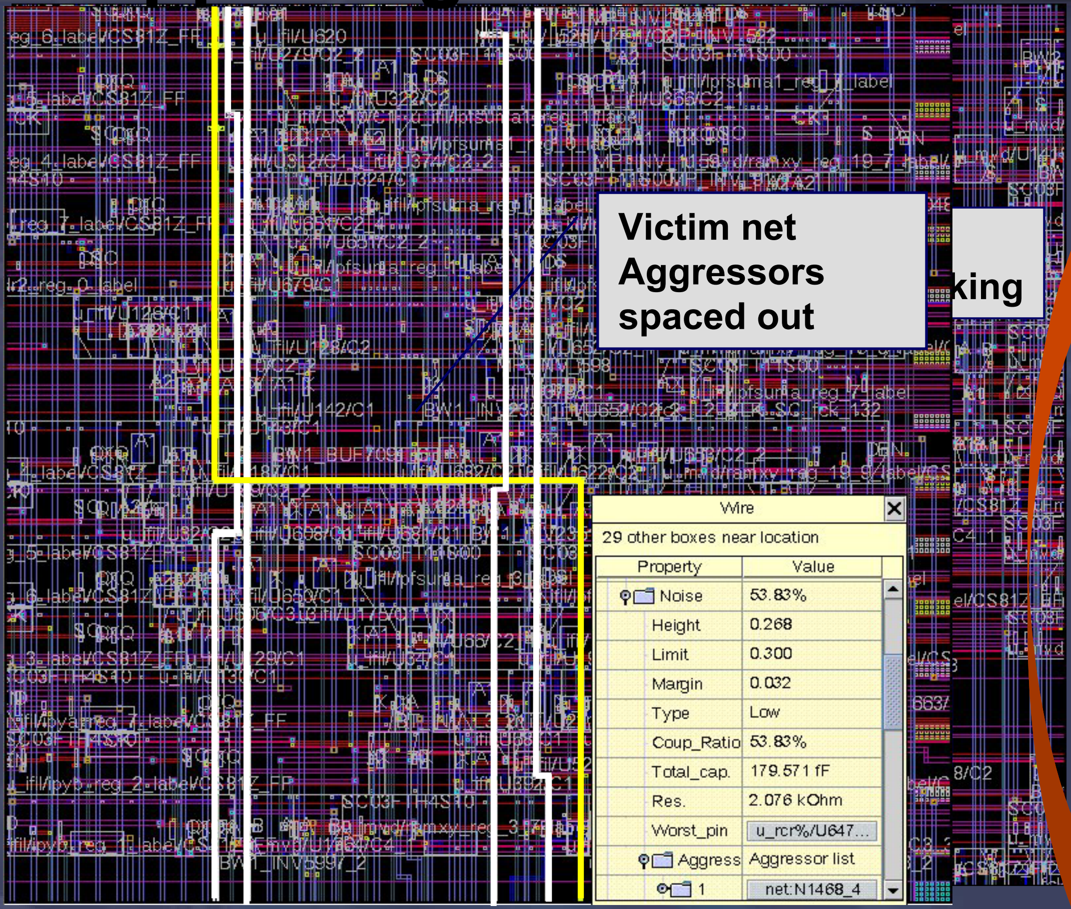
MAGMA

Tight & efficient tool integration



```
module cntr16 (pcin[15:0], pc[15:0], ck, ldPc, clrPc, se, sdi, sdo);
                                       // scan ports  ^   ^    ^
    input ck;
    input clrPc;
    input ldPc;
    input [15:0] pcin;
    input se, sdi;                     // scan enable, scan data in
    output sdo;                        // scan data out
    output [15:0] pc;

reg [15:0] outreg;

always @(posedge ck)
    begin
       if (clrPc)
         outreg = 0;
       else
       if (ldPc)
         outreg = pcin;
       else
         outreg = outreg + 16'd1;
    end

ass

end
```

Example of the strength of the data model: Supporting DSM crosstalk repair



Wire

29 other boxes near location

Property	Value
Noise	53.83%
Height	0.268
Limit	0.300
Margin	0.032
Type	Low
Coup_Ratio	53.83%
Total_cap.	179.571 fF
Res.	2.076 kOhm
Worst_pin	u_rcr%/U647...
Aggress	Aggressor list
1	net:N1468_4

- **Post-route Silicon repair flow:**
  - Update parasitic extraction
  - Update timer
  - Determine/filter problem (e.g. crosstalk noise)
  - Re-route victim:
    - redo-global route
    - redo-detailed route
  - And/or resize and buffer:
    - insert & place gate
    - redo-global route
    - redo-detailed route
  - Update parasitic extraction
  - Update timer
  - Etc. etc. etc.

- **Similar flows apply to hold time fixing, PVT issues, etc**

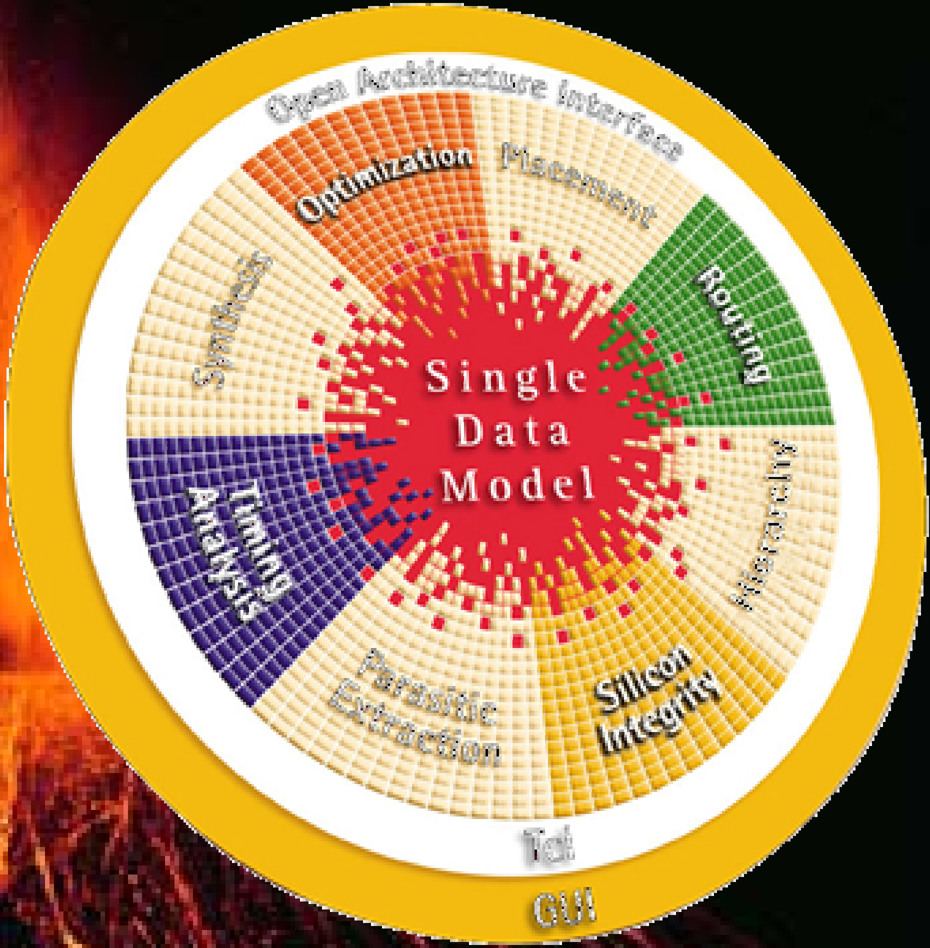
Common Data Model supports incremental design and analysis

- You can add, change or delete
  - any object in the datamodel
  - at any time
- Changes are tracked by the data model, it keeps itself consistent at all times
- The incremental analysis tools detect changes automatically
  - … and update only the affected parts
- The timer, extractor and DRC engines are brutally incremental
  - They are never run explicitly
- Result: all tools have access to most current data
  - Massive tool interoperability

MAGMA

Interfacing with the data model through TCL

- **Complete access:**
  - inspect, modify or delete any object or attribute
- **TCL scripts drive the flow**
- **The Graphical User Interface communicates through TCL**
  - easy configuration and adaptation



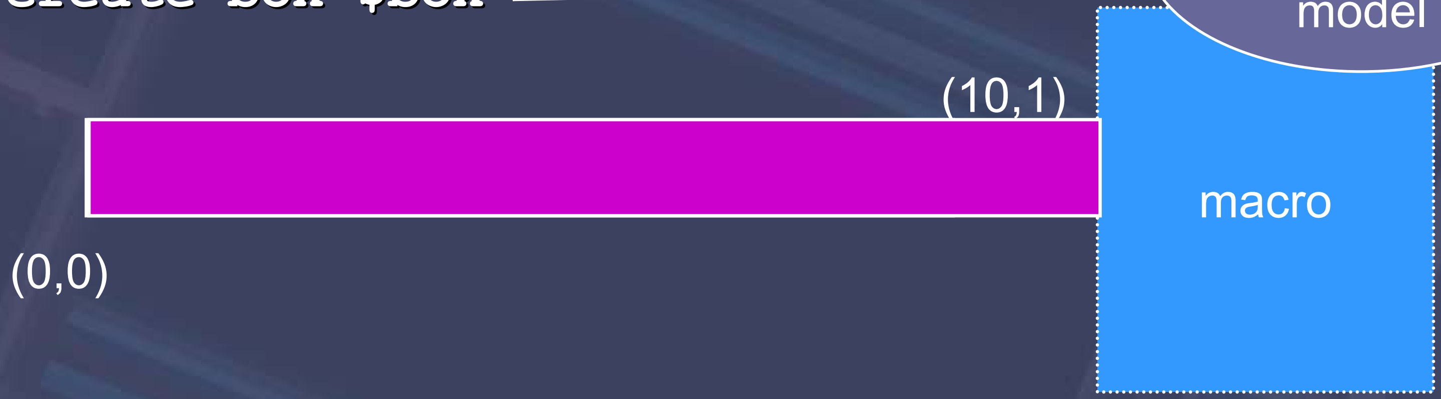
Example: interaction through TCL

- Create a box (a M1 wire owned by a net called ‘newnet’)

```
set n [data create net $m -name newnet]
set box “0 0 10u 1u M1 routing $newnet”
data create box $box
```

set up10u x 1u wire at part of

create the wire (= box) in the data model



- Stretch the wire such that it touches the macro

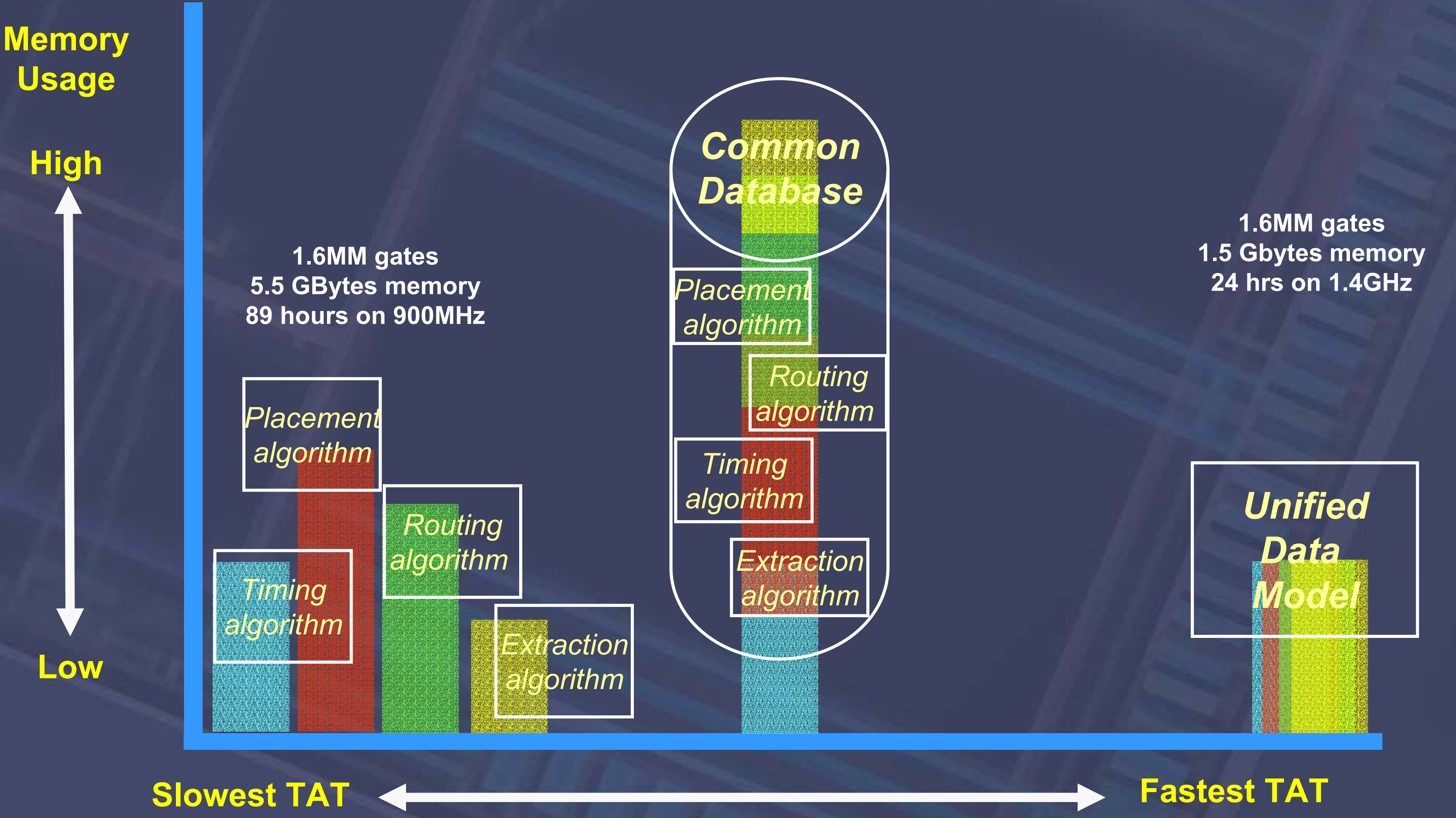
```
set macrobox [data only model_outline $macro]
data put $box right [query box left $macrobox]
```

Common Data Model: *the* enabling technology for RTL to GDSII Design Closure

- **Physical Synthesis =**
  - Logic optimization
  - Placement
- **Design Closure =**
  - RTL Synthesis
  - Design planning
  - Logic optimization
  - Placement
  - Global routing
  - Detailed routing
  - Signal Integrity
  - Etc…

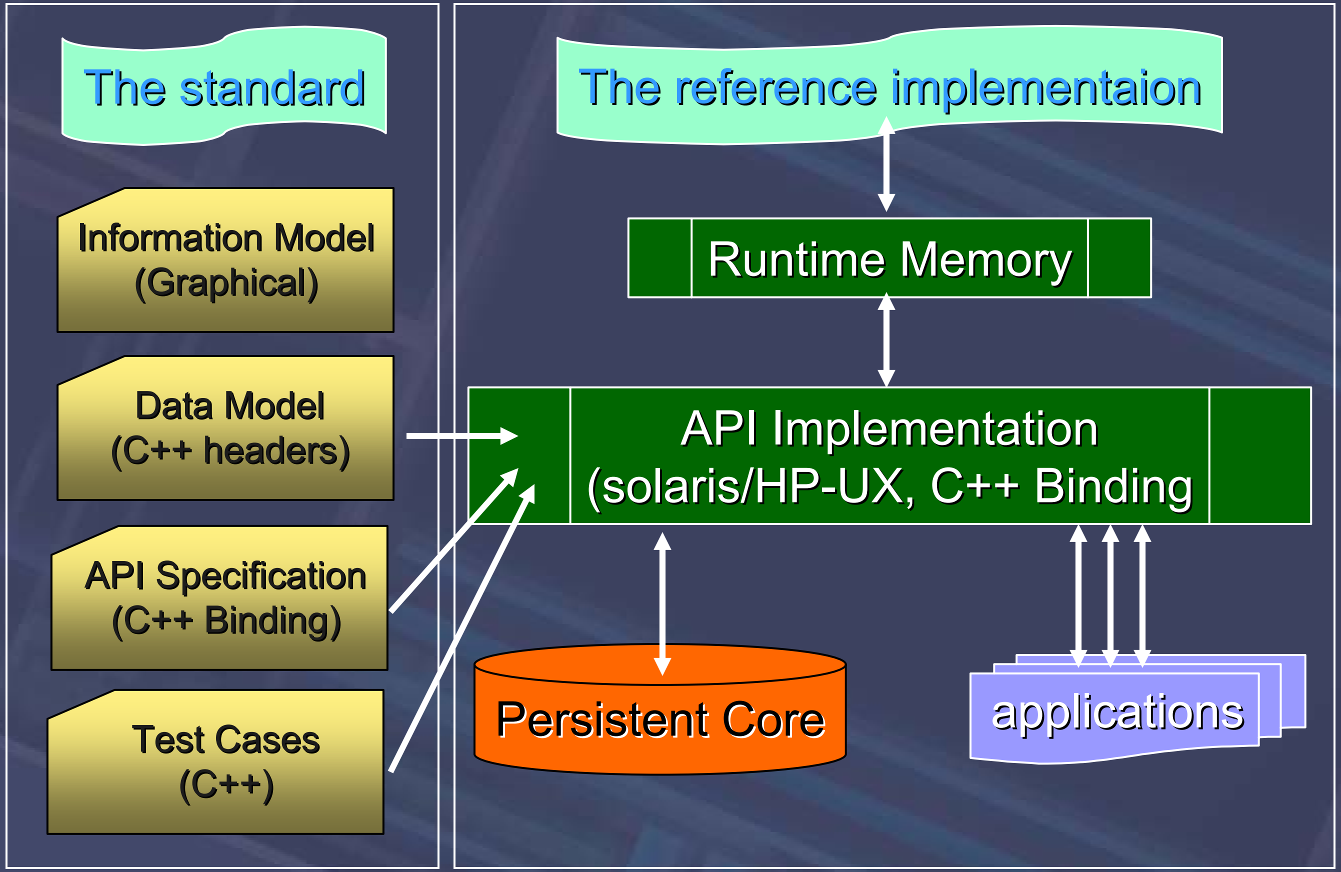
MAGMA

Integration through a database will always fall short



MAGMA

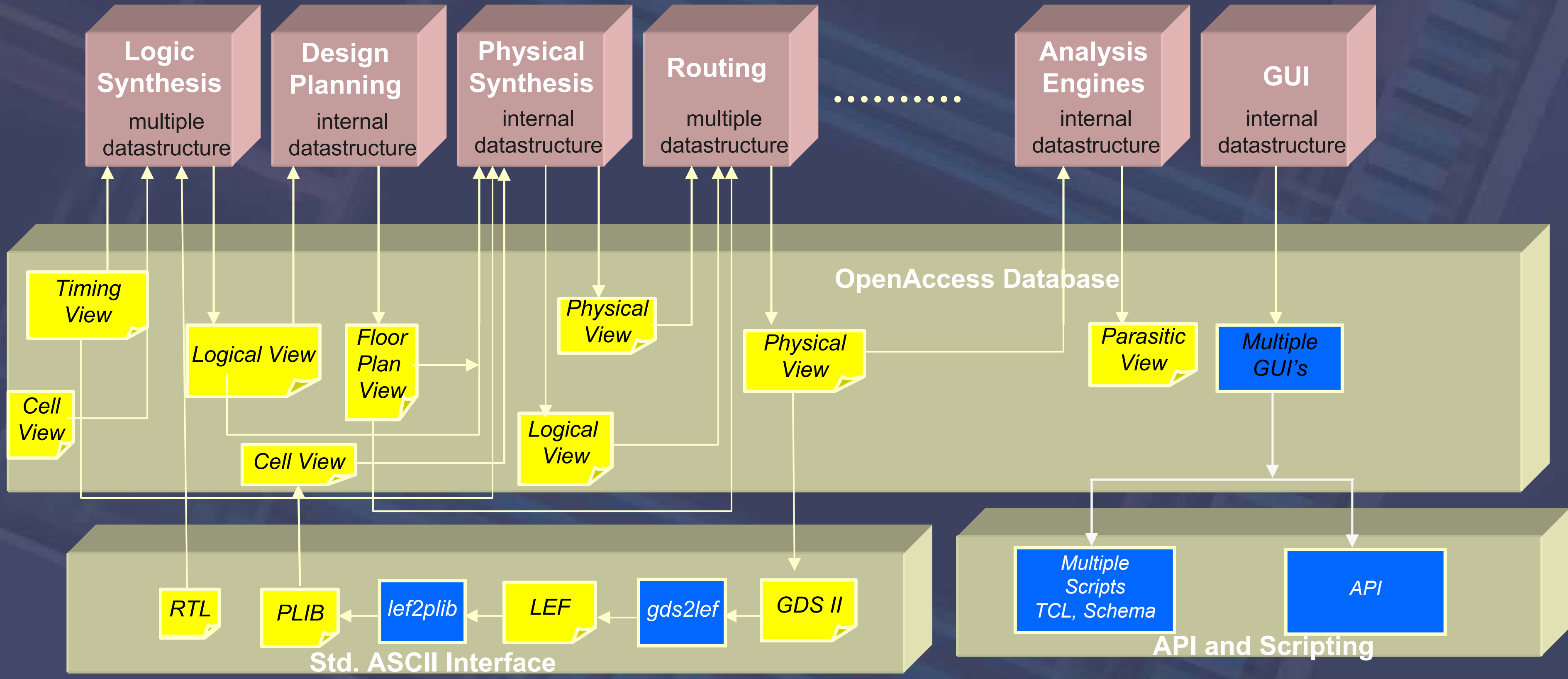
What is OpenAccess?


The standard
Information Model (Graphical)
Data Model (C++ headers)
API Specification (C++ Binding)
Test Cases (C++)
The reference implementaion
Runtime Memory
API Implementation (solaris/HP-UX, C++ Binding
Persistent Core
applications

- **Open API and source code**
- **Flexible license model**
- **High capacity & performance**
- **Advanced features:**
  - Area queryability
  - Callbacks ("don't abuse these")
  - Application defined data objects (sparse or dense)
  - Compression/hashing
  - On demand loading
  - Thread safe
  - Access management

MAGMA

Can OpenAccess be the Common Database?



Can OpenAccess be the Common Data Model?

- **Yes, in principle, but…**
  - EDA = Flow + Data model + Algorithms
  - Current generation of point tools not architected for advanced flows
  - Current generation of fully integrated tools are already on the market
    - Extremely wide API to datamodel
  - Proprietary data models provide competitive advantage
    - Control over performance and capacity
    - Carefully tuned to optimize flow
    - Quick turnaround time for bugs and enhancements
  - Many person years already invested in optimizing and debugging proprietary datamodels

MAGMA

Summary

- Tight tool integration is crucial <= 90nm
- Common data model enables integation
- "best in class point tools" – nostalgia
- Flows, flows, flows
- OpenAccess
  - Magma is committed to interoperability
  - Magma wholeheartedly supports OpenAccess
  - We will read/write to it like any other standard interchange format
  - Unlikely to replace our proprietary data model

MAGMA