A unified Data Model for EDA tool Integration

Patrick Groeneveld

EDPS 2005 Monterey

Summary

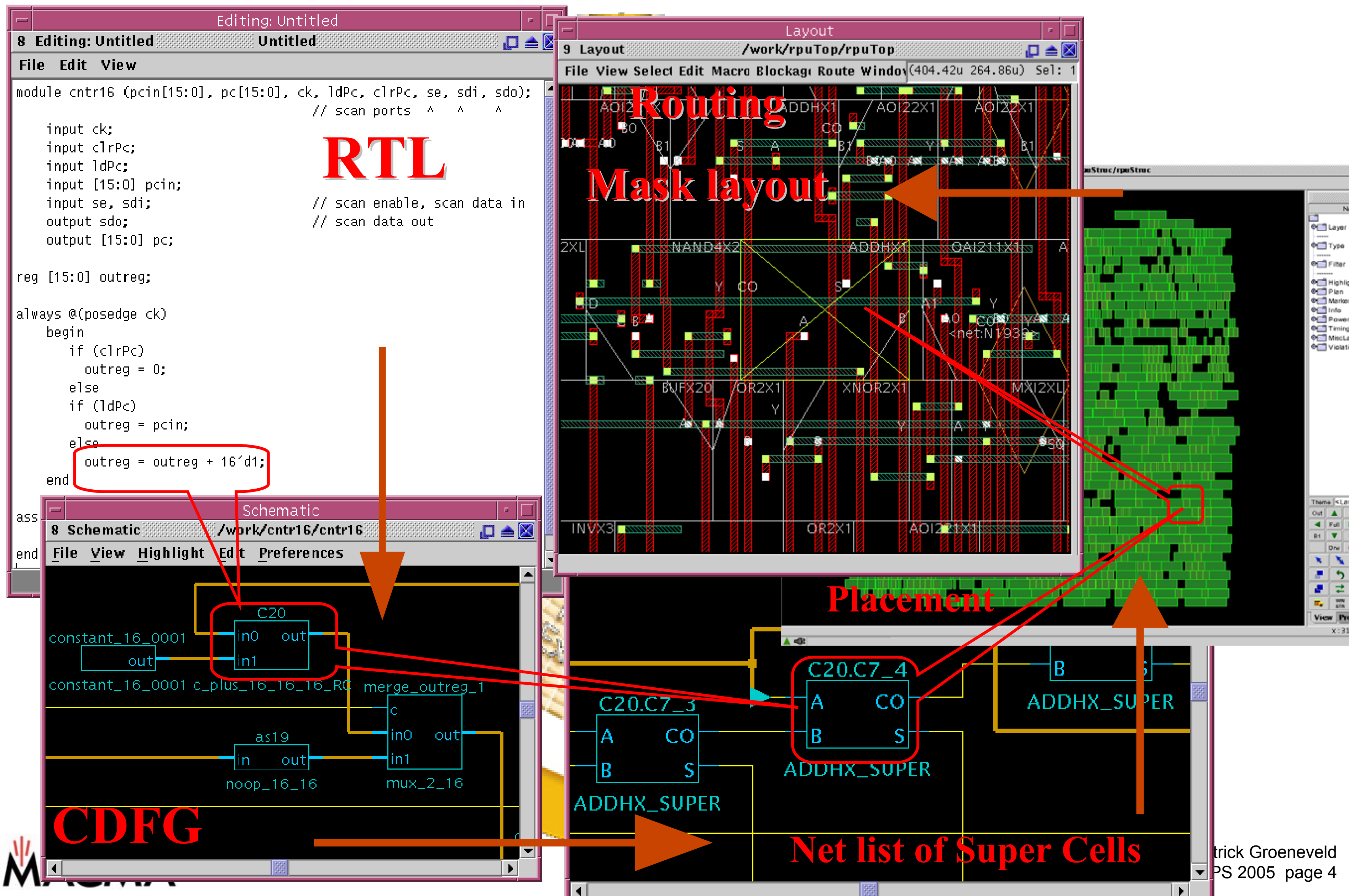
- History and guiding concepts
- Objects in the data model
- Data structure for rectangle
- TCL access to data model
- GUI and Volcanoes
- STA
- Conclusions

Magma, August 1997

Terra Bella Avenue, Mountain View, CA

MAGMA

Goal: RTL to GDSII system



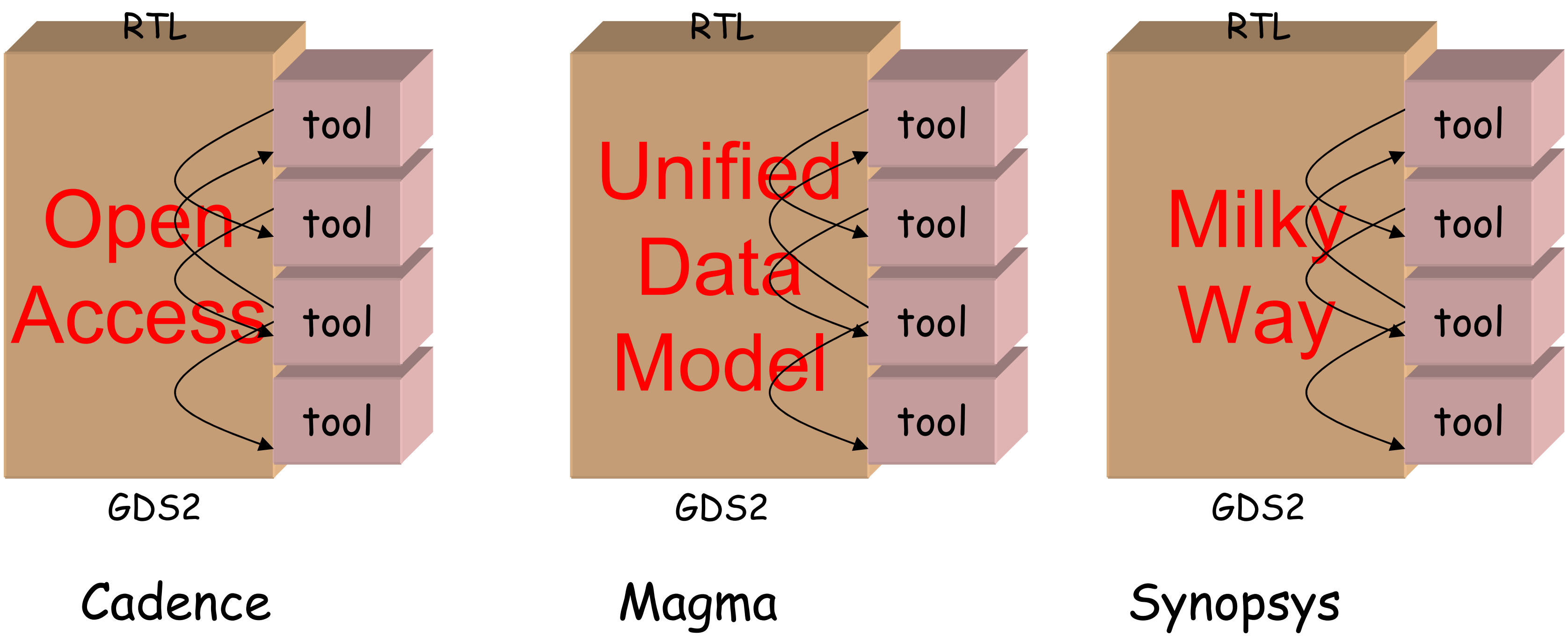
Initial Data Model design objectives

- Somehow link logical and physical world
  - Look for similarities
- Minimize implementation effort
  - Maximize code re-use: fewer lines = fewer bugs
- Efficient, fast:
  - Minimize tool communication overhead.
  - Since we anticipated significant communication
- Enable fast incremental operation
  - Keep data in-core
  - Timer, placer, routers

- Not as objective:
  - Foreign tool integration

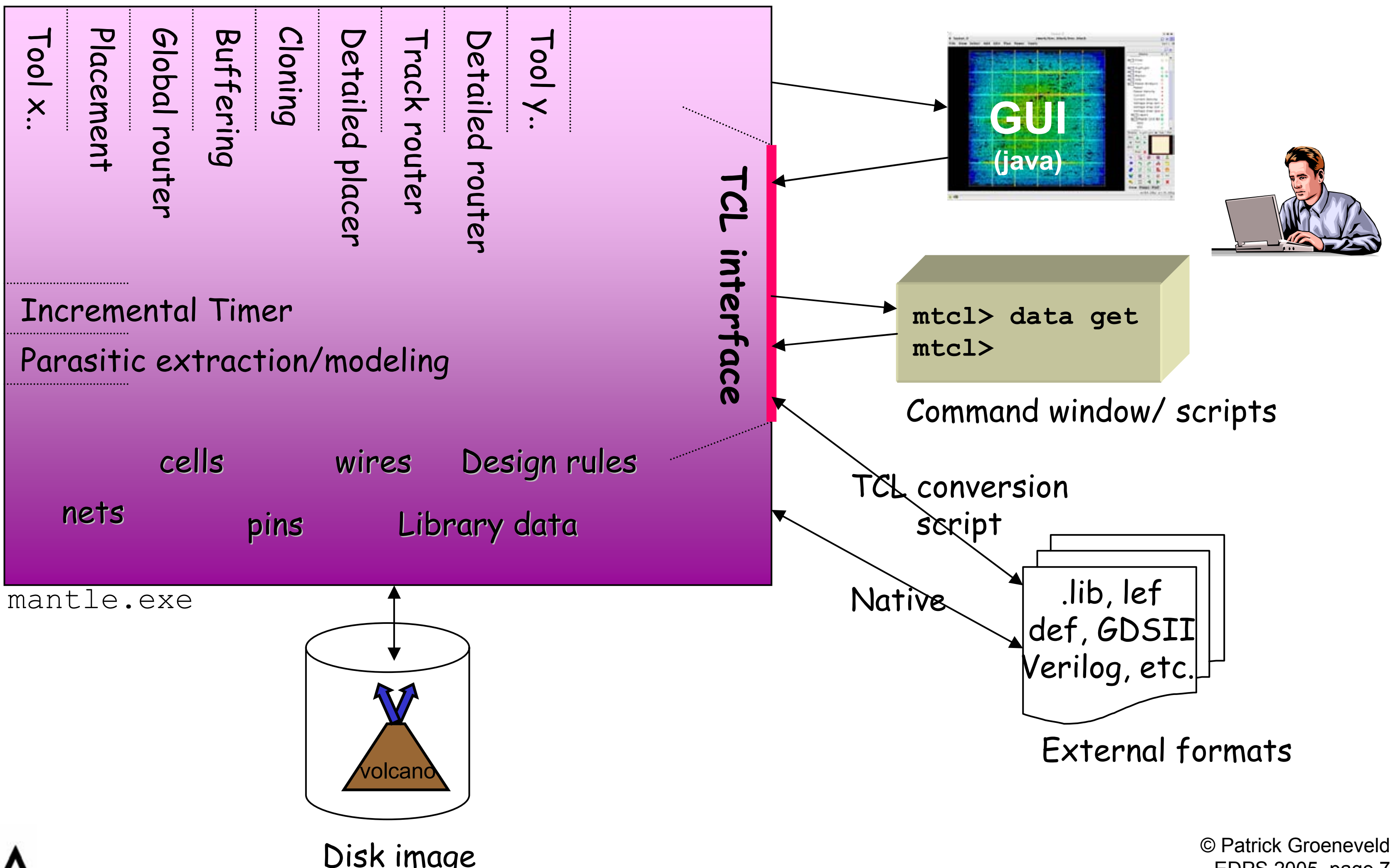
MAGMA

Interoperability for RTL2GDS2 flows



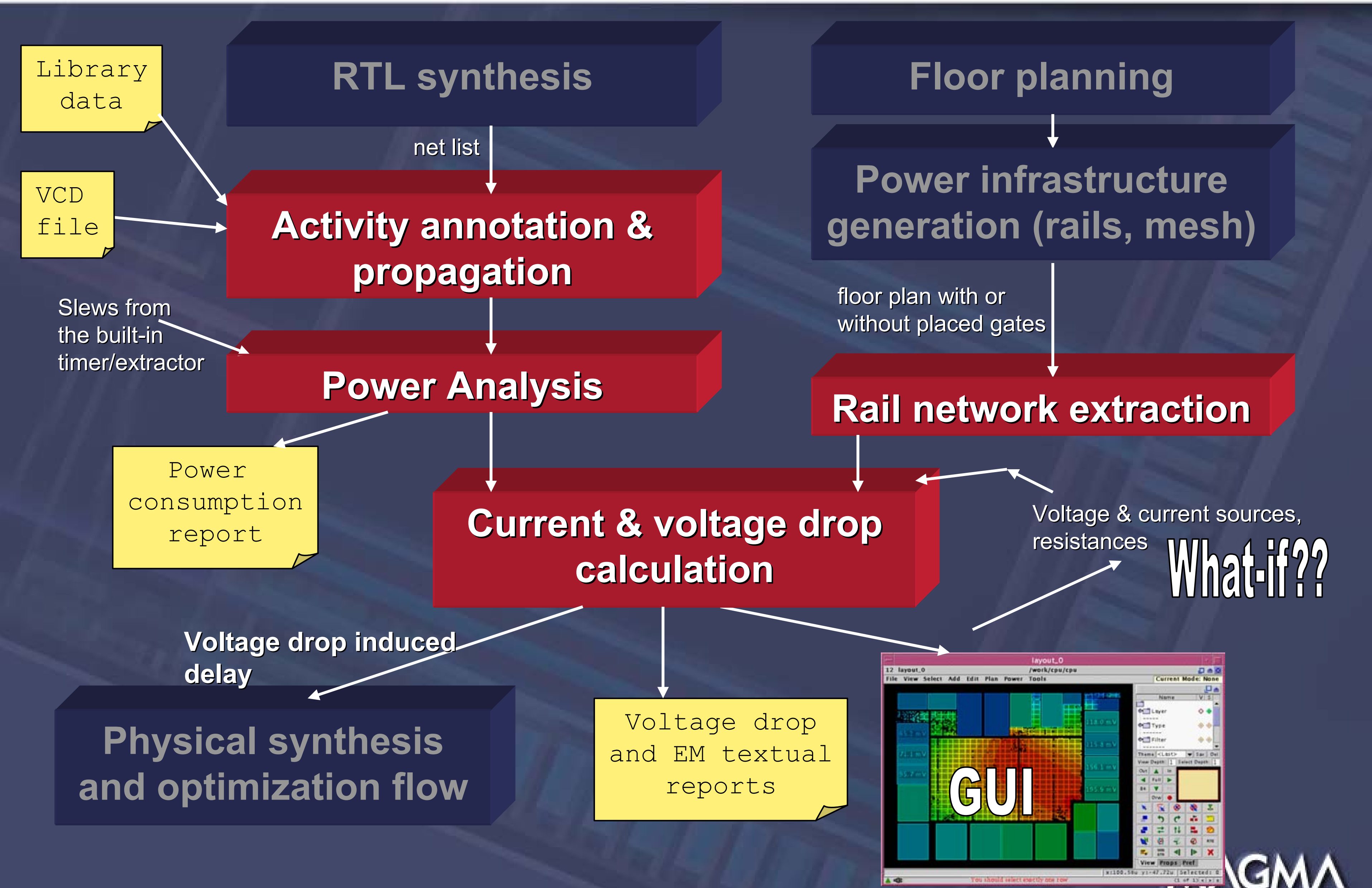
- Plugging a tool into a new environment is hard work
- Format/API is not the real tool integration problem.
- Instead, it's the interpretation of the exact meaning of the design data, and the tuning of the tool in the flow.
- Fixing the exact interpretation stifles innovation.

General architecture

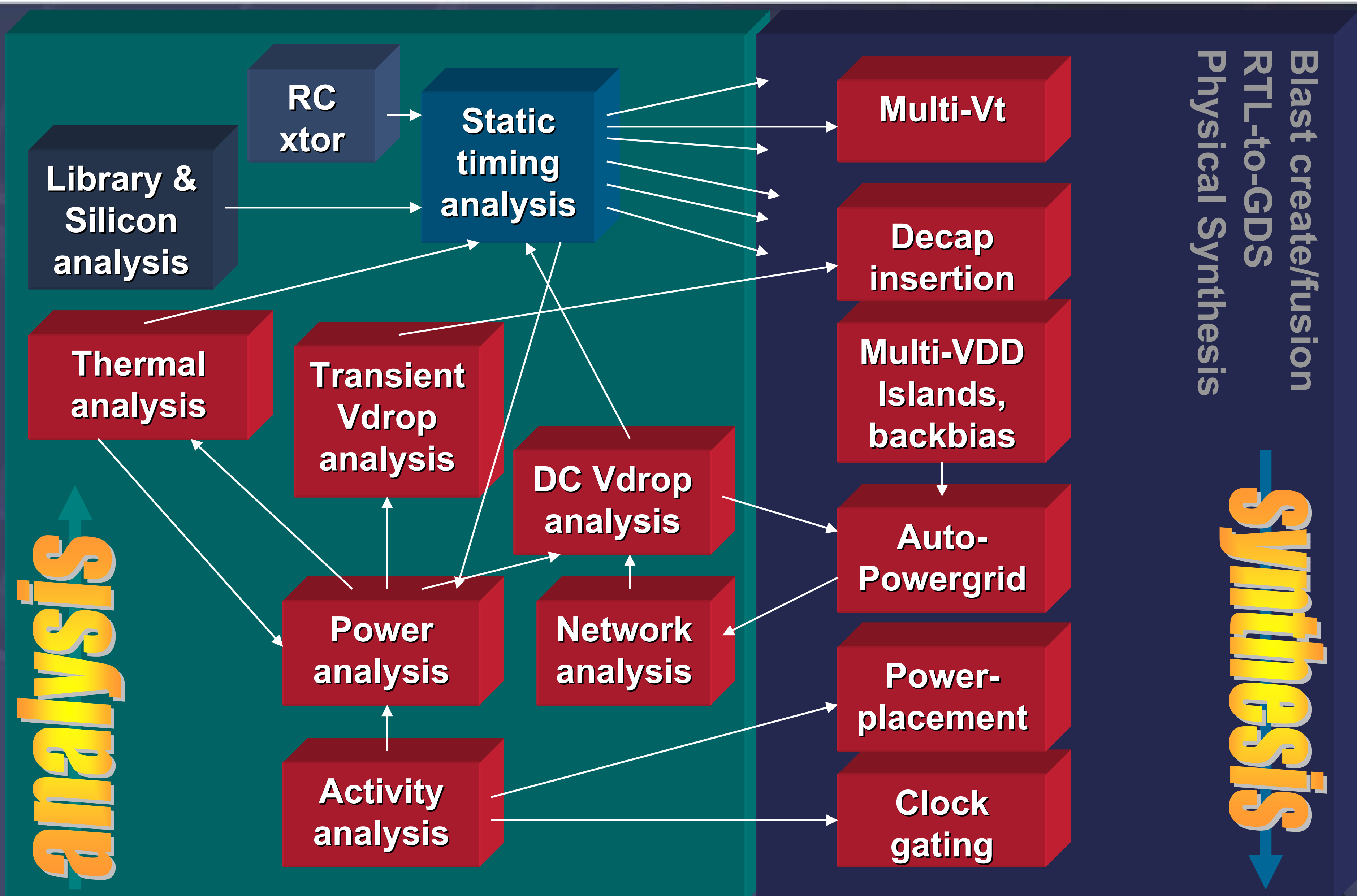


MAGMA

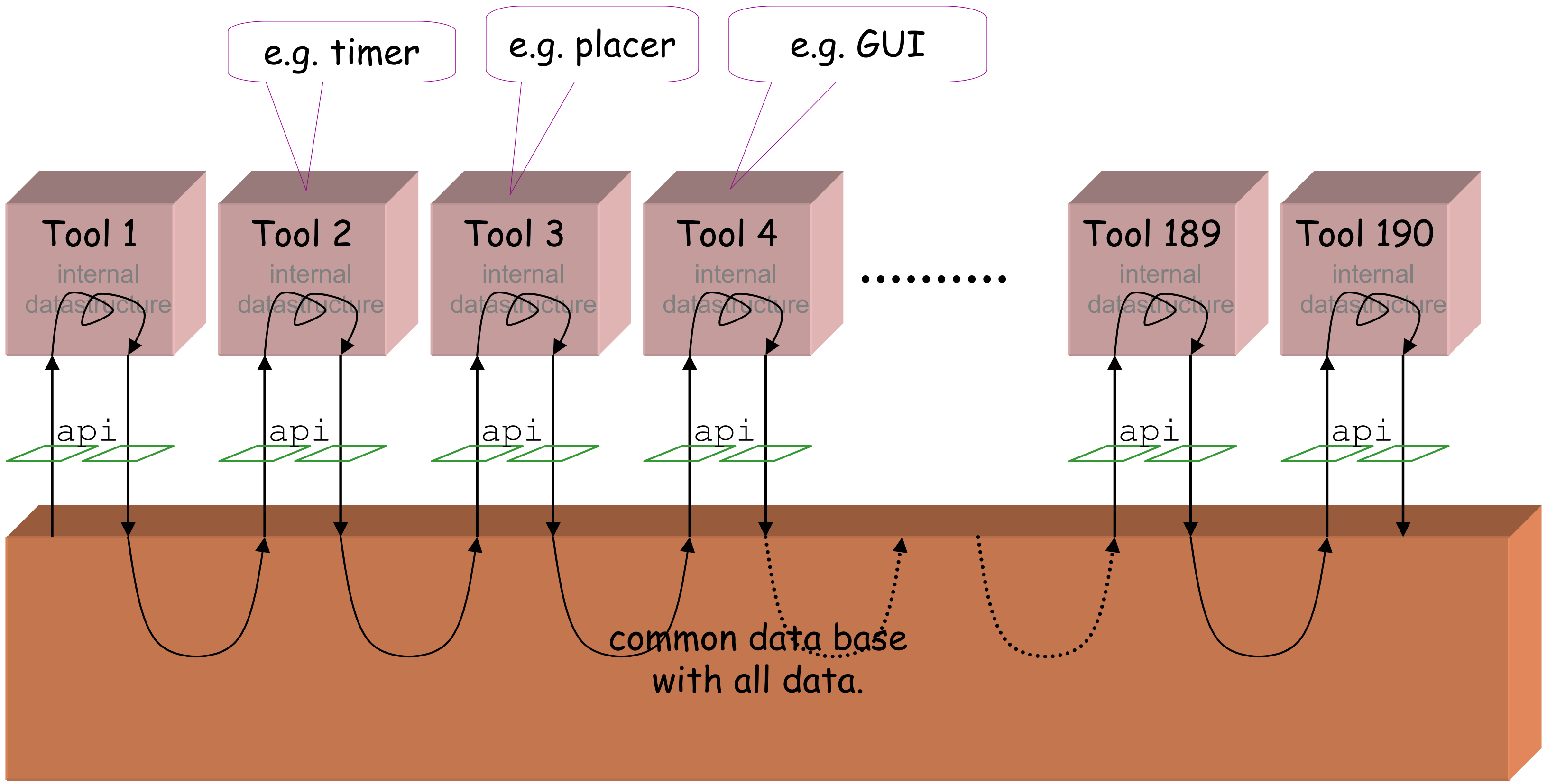
Blast Rail power analysis steps in magma flow



Complex tool interactions without disk access

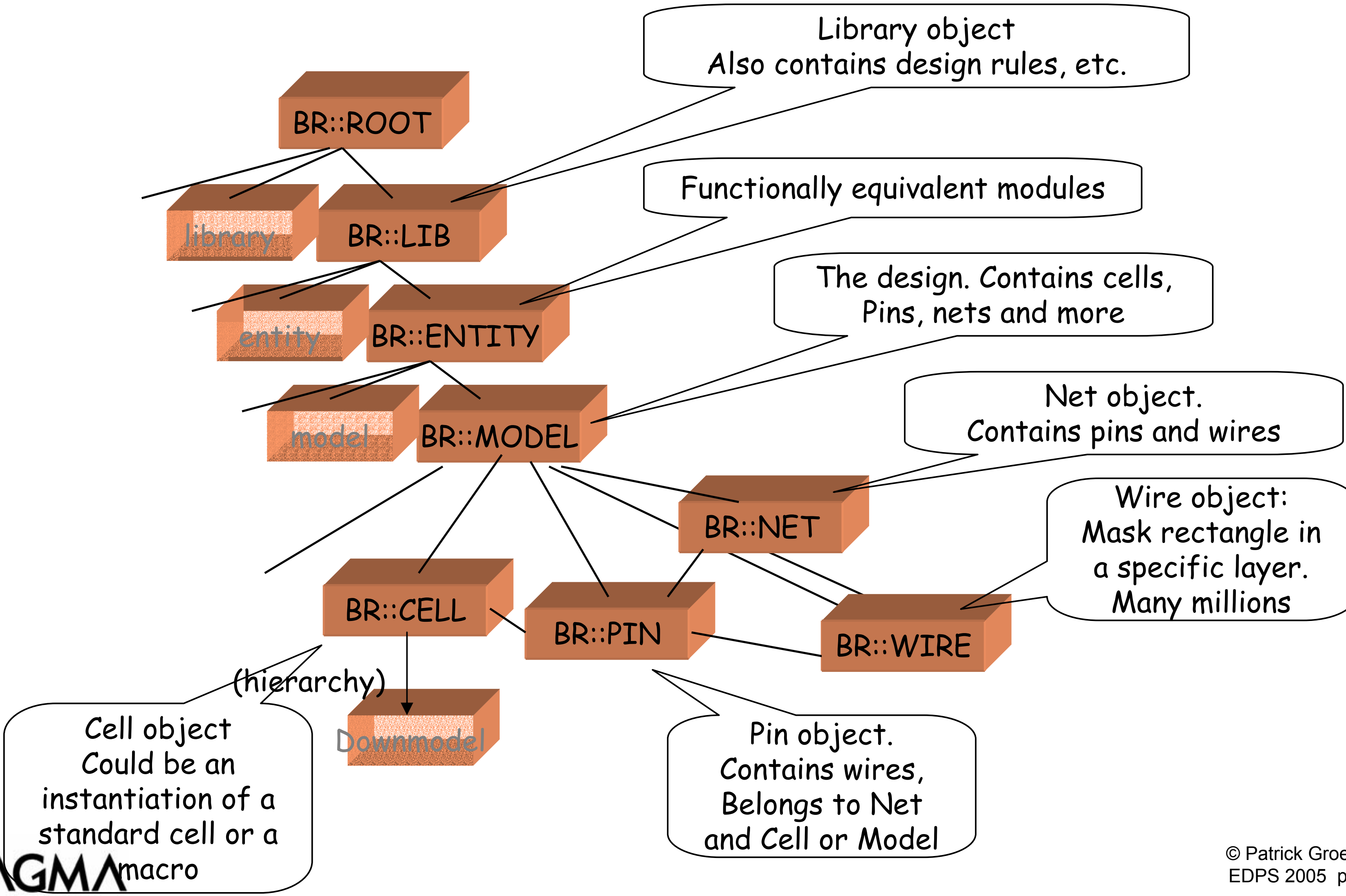


Transport database doesn't work well



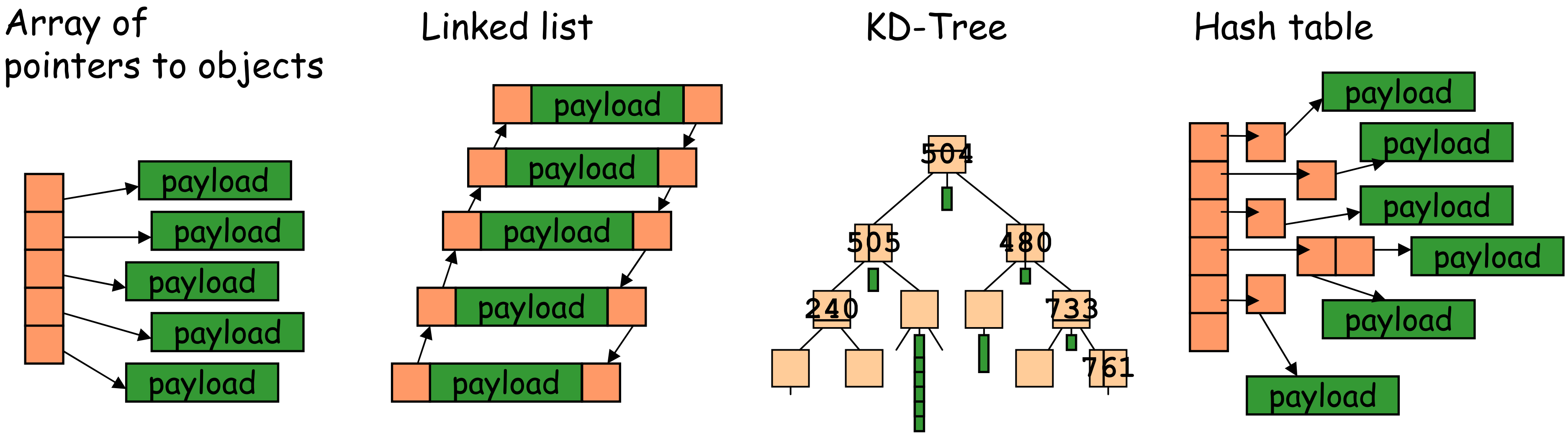
- The communication between tools cannot afford this slow format conversions.
- Data is duplicated.

Base objects in the in-core data model



MAGMA

Implementing ordered collections of objects

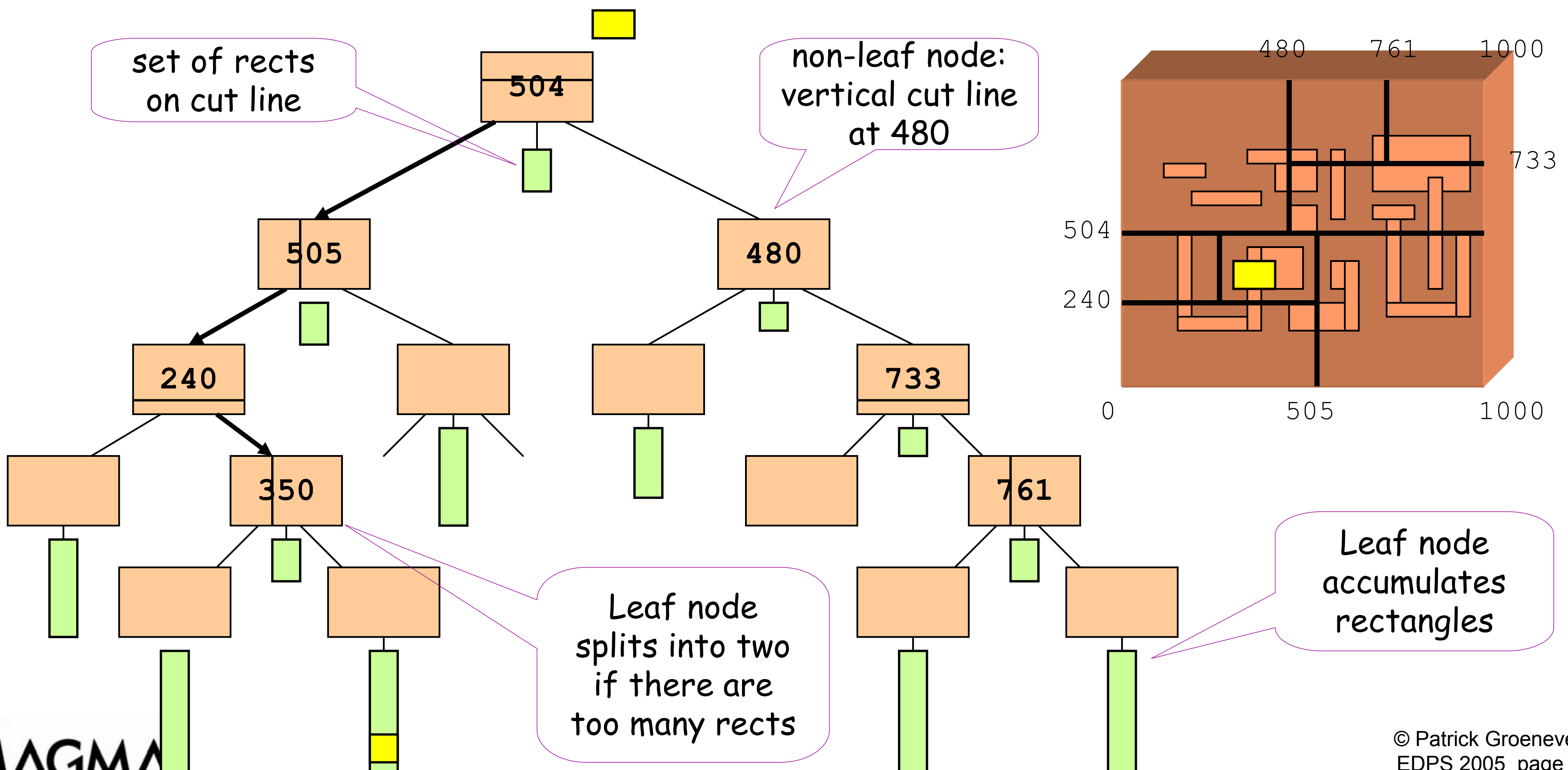


- Each structure has its own "sweetspot":

	Memory overhead	Find specific object	Find neighboring	Add/remove speed
Array	15%	n	n	bad
Linked list	30%	n	n	Very good
KD-Tree	20%	log(n)	log(n)	OK
Hash Table	30%	1	n	good

The structure in `GEO::KDTREE`

- This is a binary tree. The nodes of the tree contain a set of rectangles.
- A non-leaf node contains cuts the design in three pieces:
  - the part that is completely *left* (below) the cut line
  - the part that *touches* the line
  - the part that is completely *right* (above) the cut line



MTCL: access to data model through TCL

- Full access to the data model is provided through TCL
- Every object is uniquely 'addressable' by a text string.
- This addresses cell '`gate744`' in model '`display`':

```
mtcl> set c /work/display/display/cell:gate744
```

library — entity — model — Name of the cell

- This would list the nets in model $m:

```
mtcl> data list model_net $m
 /work/display/display/net:clock1, /work/display/display/net:enable,..
```

- This deletes a net:

```
mtcl> data delete object /work/display/display/net:clock2
```

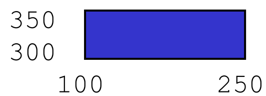
MAGMA

MTCL: addressing rectangles

- The millions of physical objects can be uniquely addressed by their coordinates in the string

`100n 300n 150n 50n METAL1 routing /work/display/display/net:n1244`

Left
Bottom
Width
Height
layer
layer type
'owner' of this box



Metal 1 wire of net n1244

`500n 700n 150n 50n METAL3 segment /work/display/display/net:n1244`

Metal 3 global routing segment of the net n1244

`100n 300n 10n 10n . outline /work/display/display/cell:c32`

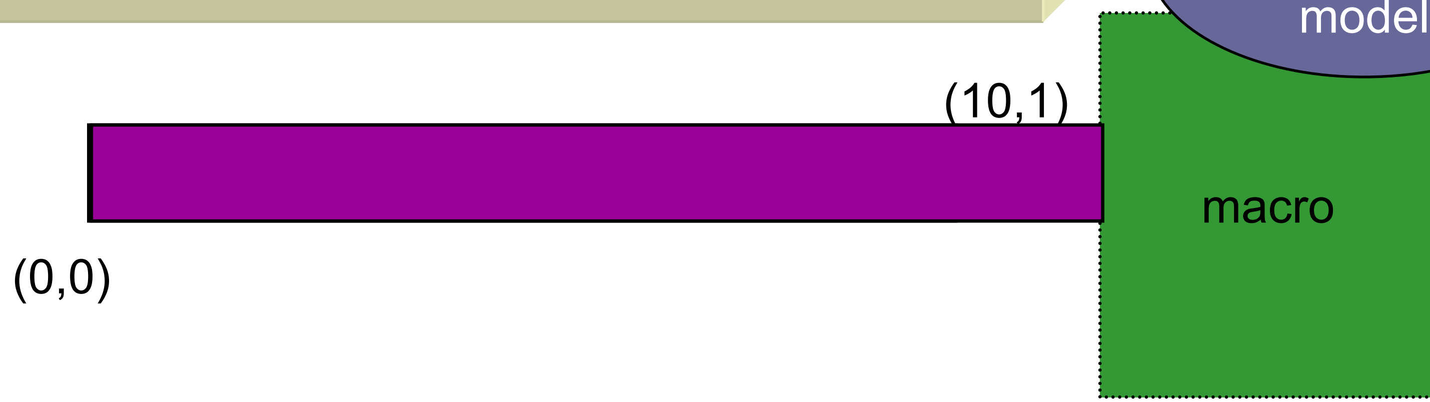
Cell c32 (and its placement)

MAGMA

Example: manual interaction through MTCL

- 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 $n"
data create box $box
```



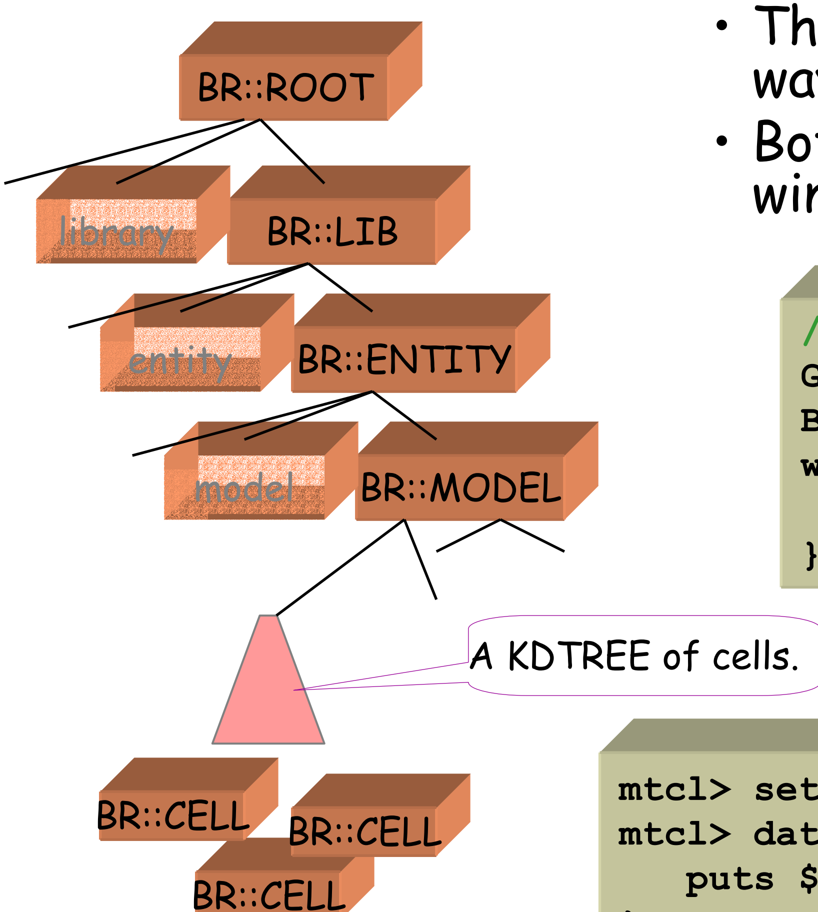
- Stretch the power line such that it touches the macro:

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

MAGMA

Finding the cells in a window

- The cells in our model (= `BR::MODEL`) are rectangles (= `GEO::RECT`).
- They are stored in a `GEO::KDTREE`. In that way, they are easily area-queryable.
- Both the C++ as the MTCL iterator take a window as optional argument.

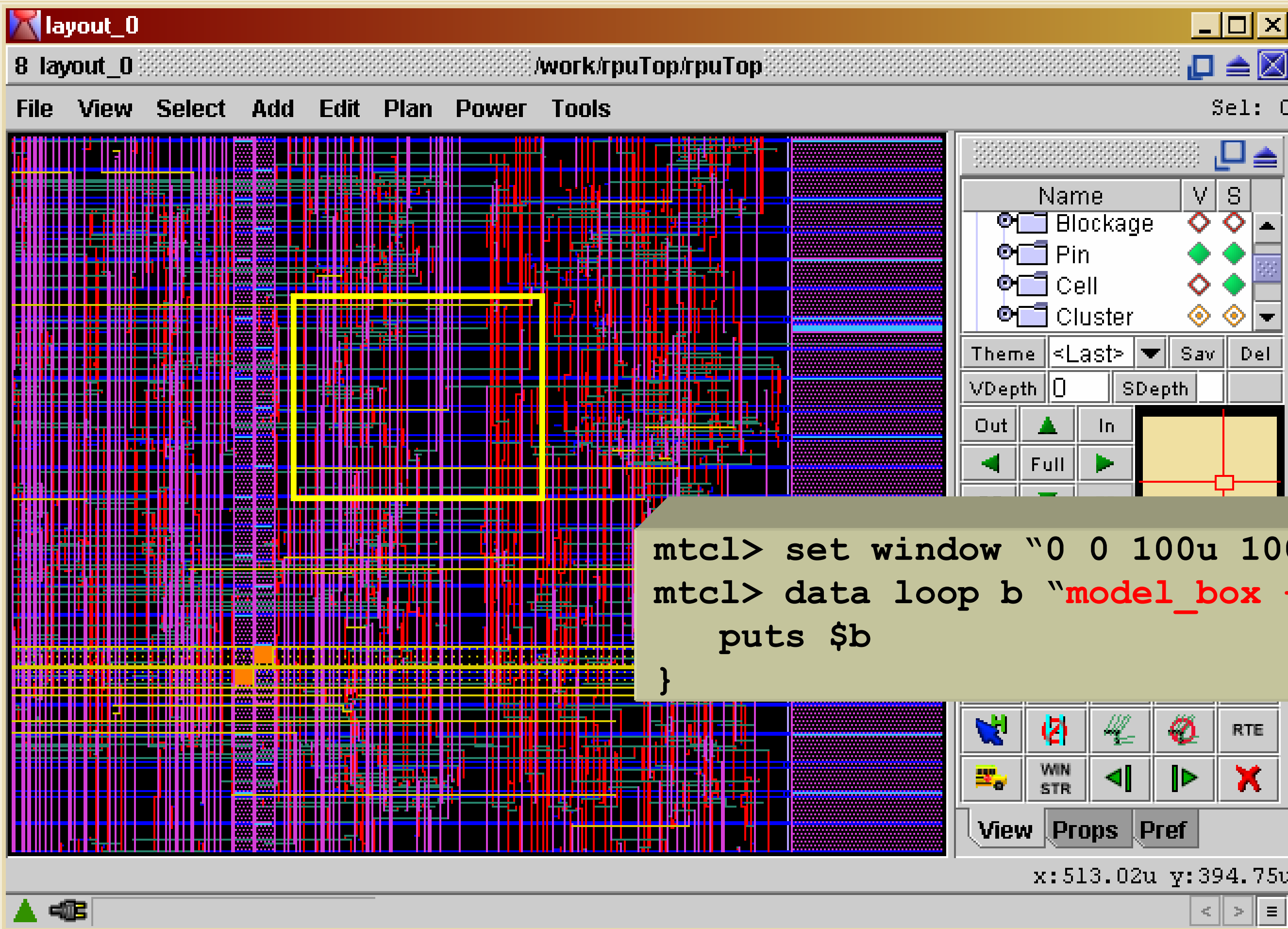



```
// prints all cells in the window
GEO::RECT window(0, 0, 100000, 100000);
BR::MODEL::CELL_ITER cit(model, &window);
while(cell = cit.next()) {
   cell->print();
}
```

```
mtcl> set window "0 0 100u 100u"
mtcl> data loop c "model_cell -window $window " $m {
   puts $c
}
```

MAGMA

Getting the wires in a window




```
mtcl> set window "0 0 100u 100u"
mtcl> data loop b "model_box -window $window" $m {
   puts $b
}
```

- This is based on the KDTREE area query. The complexity of the layer structure and the hierarchy is hidden behind this iterator.

Magma RTL-to-GDS script in TCL

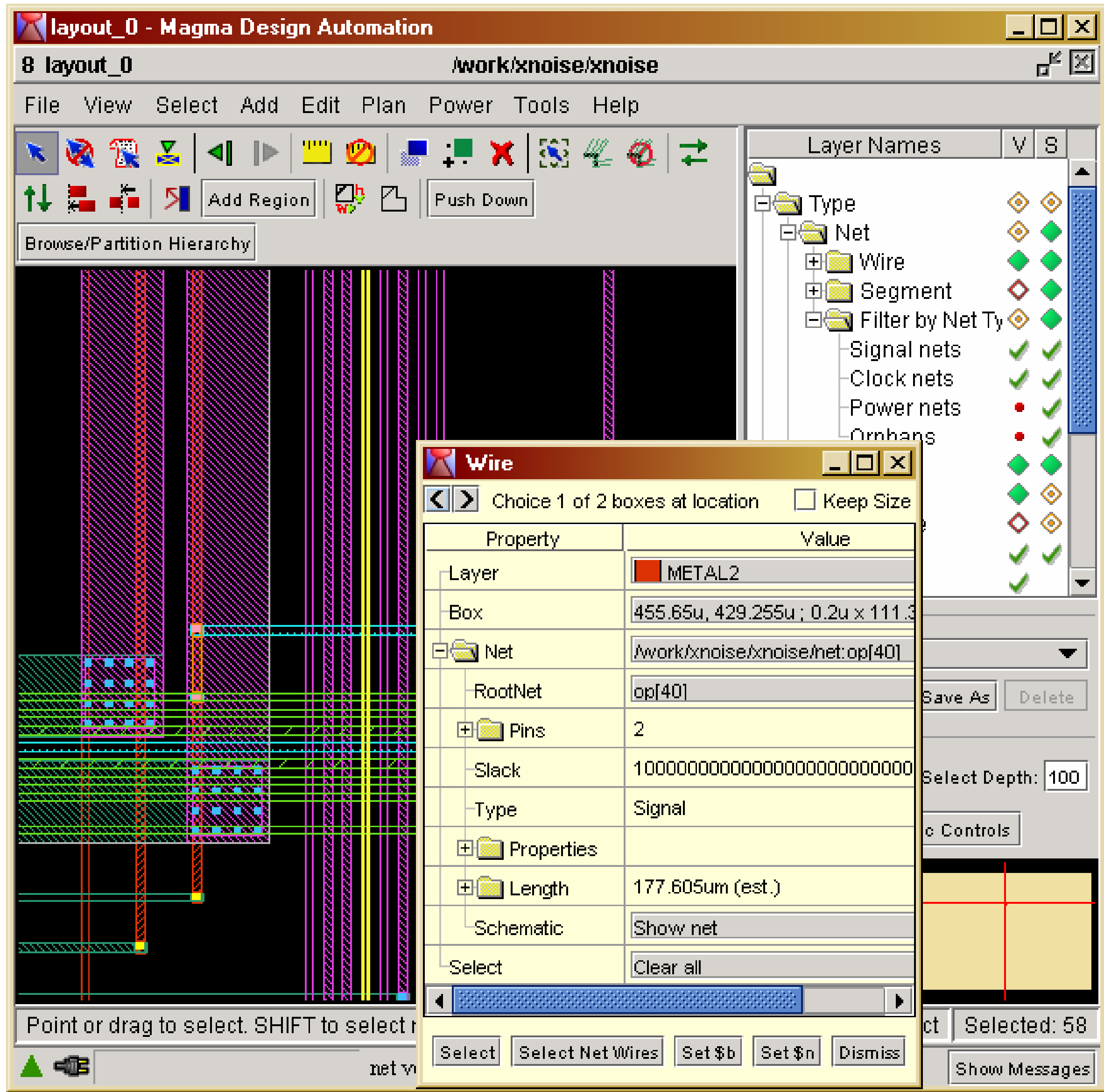
```
set m [import verilog mydesign.v]
import volcano library.volcano
fix rtl $m $l
fix time $m $l
fix plan $m $l
fix cell $m $l
fix clock $m $l
fix wire $m $l
export volcano mydesign.volcano
export gdsii $m mydesign.gds
```

```
check model $m -level final
run route stub $m
run route global $m -antenna
run route track $m -optimize noise
run route power $m -final
check route spacing_short $m
check route open -segment $m
run route final $m -singlepass
run route antenna $m
run route refine $m
run route final -incremental $m
check route drc $m
```

MAGMA

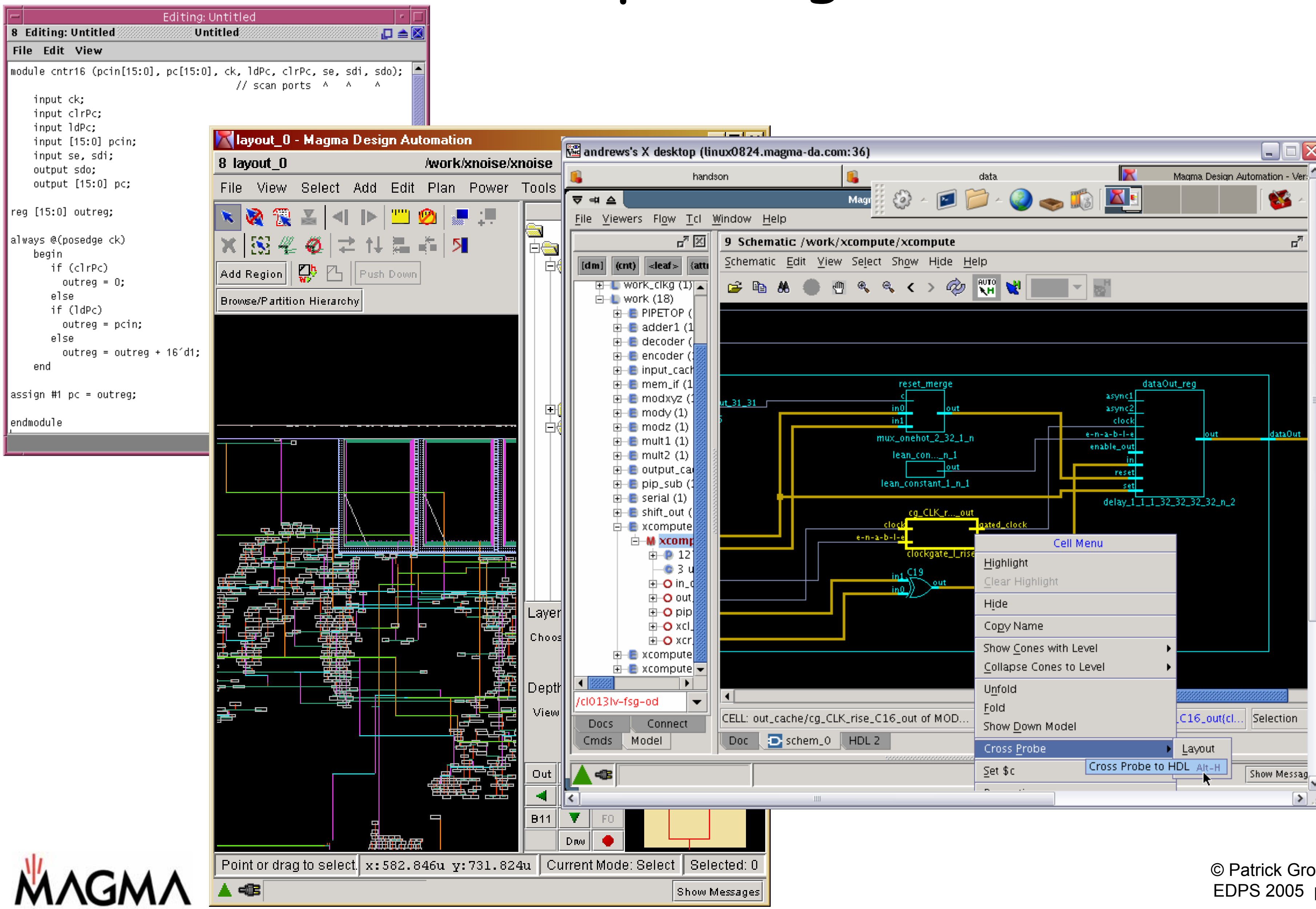
The GUI

- The GUI is a universal viewer and editor on the data model
  - A graphical extension of TCL access.
- All physical objects can be viewed, queried and modified.
- Use straightforward KDTREE iteration for drawing a window.
- 'right-mouse-button click' shows all properties of the object.
- Also displays timing paths, DRC's, hierarchy, voltage drop, etc. etc.



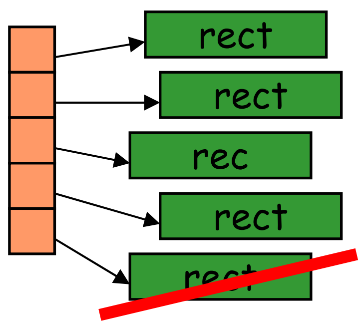
MAGMA

X-probing

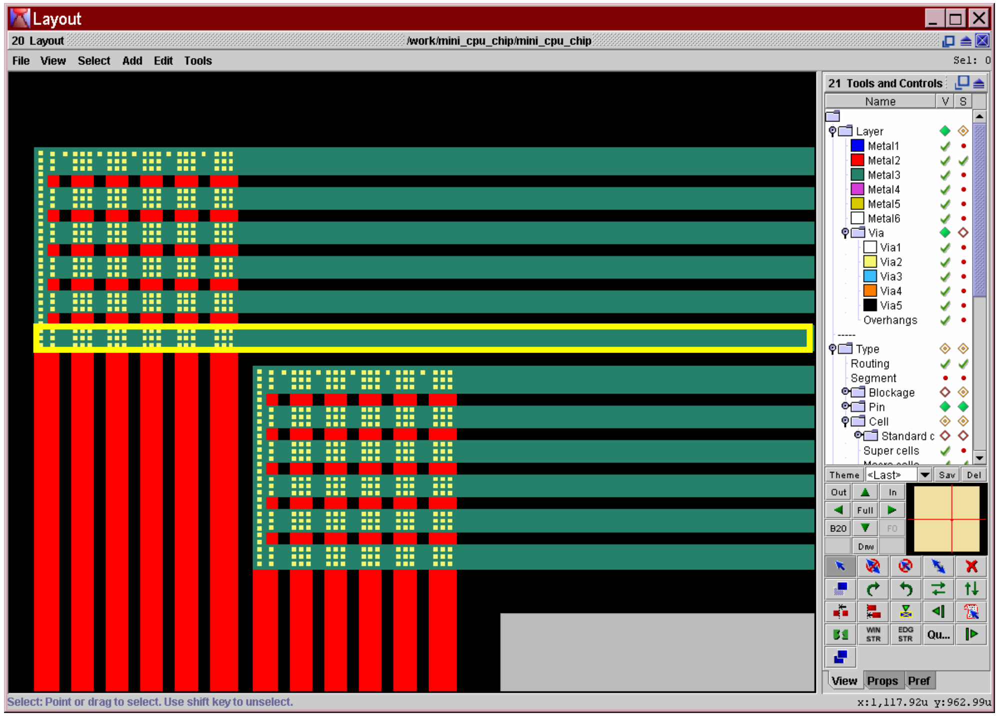

Editing: Untitled
8 Editing: Untitled Untitled
File Edit View
module cntr16 (pcin[15:0], pc[15:0], ck, ldPc, clrPc, se, sdi, sdo);
// scan ports ^ ^ ^
input ck;
input clrPc;
input ldPc;
input [15:0] pcin;
input se, sdi;
output sdo;
output [15:0] pc;
reg [15:0] outreg;
always @(posedge ck)
begin
if (clrPc)
outreg = 0;
else
if (ldPc)
outreg = pcin;
else
outreg = outreg + 16´d1;
end
assign #1 pc = outreg;
endmodule
layout_0 - Magma Design Automation
8 layout_0 /work/xnoise/xnoise
File View Select Add Edit Plan Power Tools
Add Region
Push Down
Browse/Partition Hierarchy
Point or drag to select x:582.846u y:731.824u Current Mode: Select Selected: 0
Show Messages
andrews's X desktop (linux0824.magma-da.com:36)
handson
data
Magma Design Automation - Ver
File Viewers Flow Tcl Window Help
9 Schematic /work/xcompute/xcompute
Schematic Edit View Select Show Hide Help
work (18)
PIPETOP
decoder
encoder
modxyz
mody (1)
modz (1)
mult1 (1)
mult2 (1)
serial (1)
xcompute
reset_merge
mux_onehot_2_32_1_n
lean_constant_1_n_1
dataOut_reg
delay_1_1_1_32_32_32_32_n_2
cg_CLK_r..._out
gated_clock
clockgate_l_rise
C19
dataOut
Cell Menu
Highlight
Clear Highlight
Hide
Copy Name
Show Cones with Level
Collapse Cones to Level
Unfold
Fold
Show Down Model
Cross Probe
Layout
Cross Probe to HDL Alt-H
Set $c
CELL: out_cache/cg_CLK_rise_C16_out of MOD...
Selection
Docs
Connect
Cmds
Model
Doc
schem_0
HDL 2
Show Messag

Engineering issue: safety

- TCL access to data model must be very robust.
- Major problem is the reference of 'dead objects'

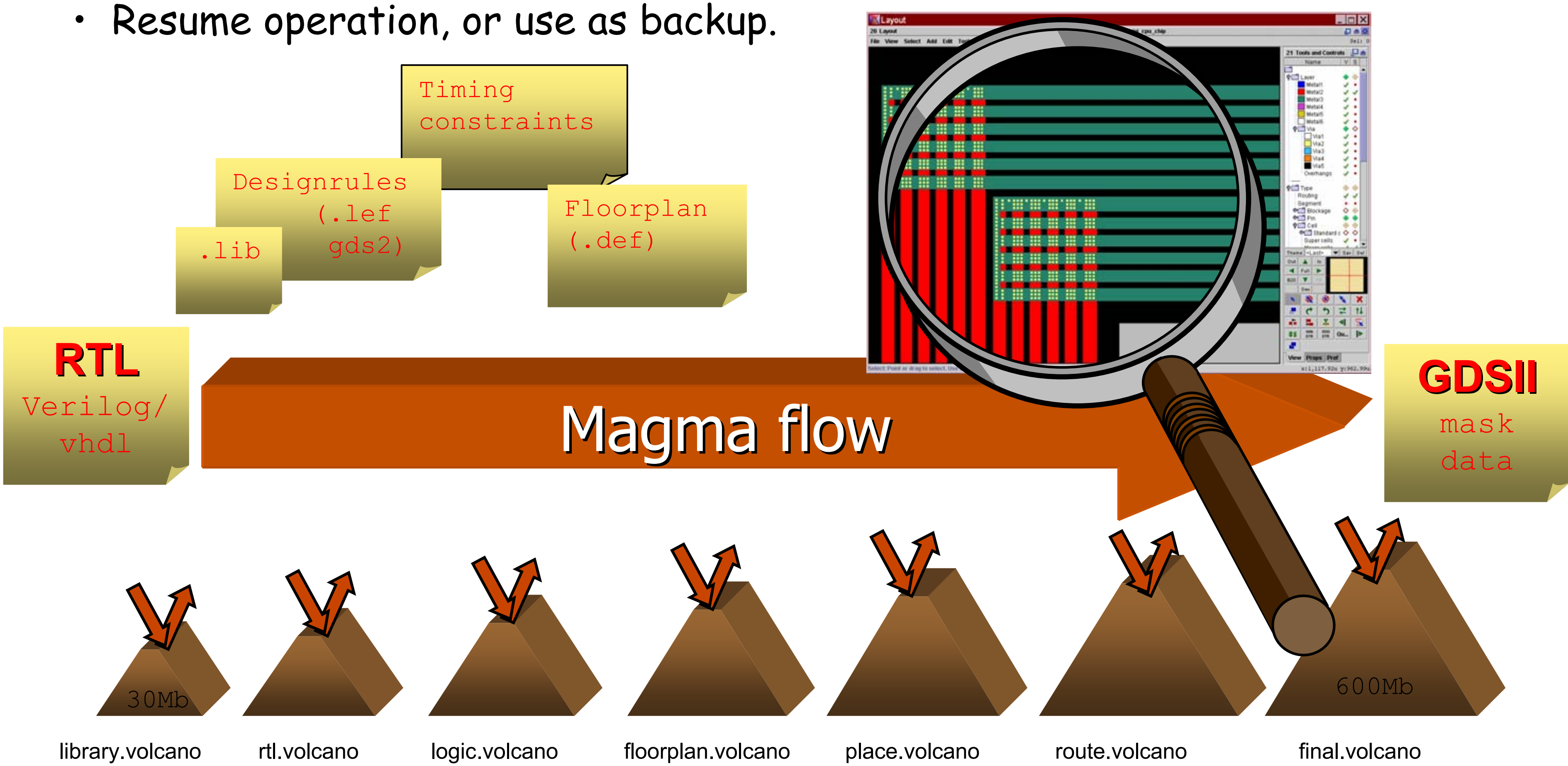


Selection set

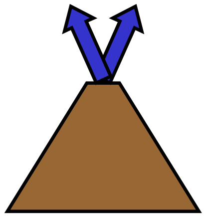


'Volcanoes': snapshots of the flow

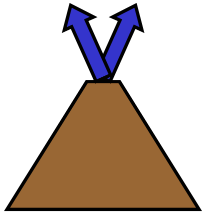
- The contents of the magma data model can be written to disk at any time during the design flow. A volcano contains a complete snapshot of *all* design data.
- Resume operation, or use as backup.



Volcanoes



Library.volcano



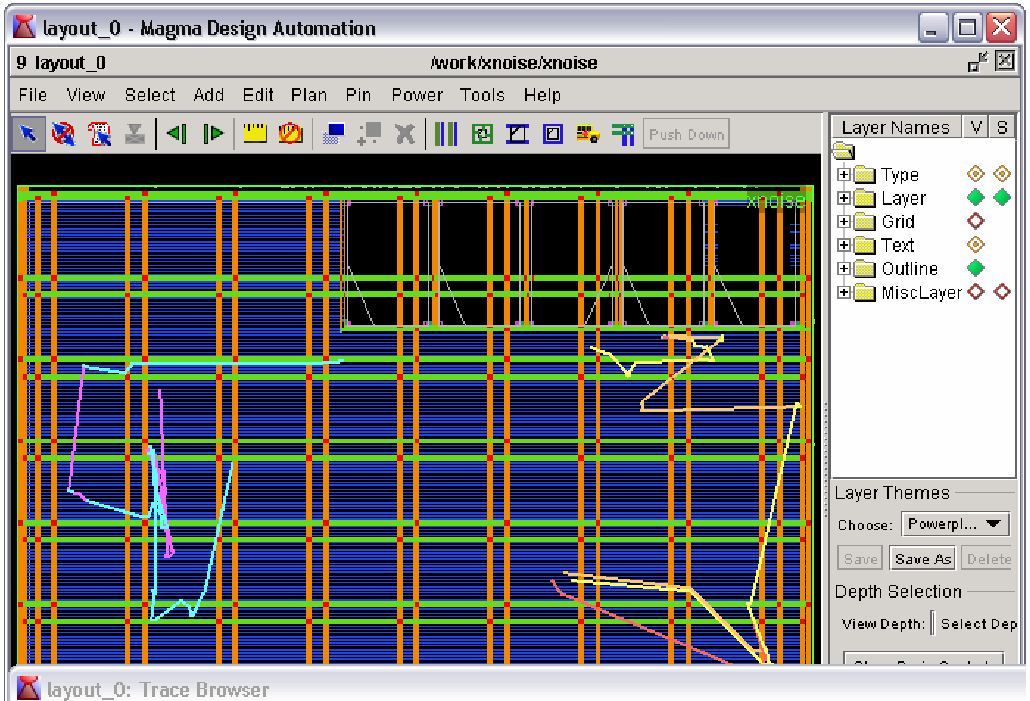
Optimize.volcano

- Essential for team cooperation. Flows can be cut into pieces.
- Key is the all-or-nothing concept: a single volcano is enough!
- Format is binary, but portable across platforms.
- Built in compression reduces disk image.

```
# script part 1
source mysettings.tcl
…
fix_whatever $m $1
…
export volcano part1.volcano
exit
```

```
# script part 2
source mysettings.tcl
import volcano part1.volcano
fix_somethingelse $m $1
…
export volcano part2.volcano
exit
```

Built-in incremental Timer



i	color	start	end	slack	delay	visible	display
1		decode_i1/store2_reg[0]/Q	decode_i1/op_cache2_64_reg[63]/D	18.795	4,642.833	✓	flyline
2		decode_i1/store2_reg[0]/Q	decode_i1/op_cache2_64_reg[61]/D	23.129	4,683.145	✓	flyline
3		decode_i1/store2_reg[0]/Q	decode_i1/op_cache2_64_reg[59]/D	55.673	4,578.812	✓	flyline
4		decode_i2/store_reg[3]/Q	decode_i2/op_cache1_64_reg[62]/D	65.713	4,520.215	✓	flyline
5		decode_i2/store_reg[3]/Q	decode_i2/op_cache1_64_reg[37]/D	66.743	4,378.746	✓	flyline

- Incremental.
  - The data model records any changes. Values are cached and recalculated when necessary.
- Timer has 4 different levels of accuracy to evaluate wire load:
  - WLM, HPW, global route, detailed route
- Performs 'on-the-fly' parasitic extraction of wires.

MAGMA

Summary

- Fast high-capacity in-core data model:
  - Data is stationary in Data Model
  - Tools operate on it.
- Simple object-oriented structure
- Deep access to all data enables extensive customization.
- Built-in timer, extractor and DRC
  - Data is correct and up-to-date at any time.
- Data model was key for the success of Magma
  - Used on many hundreds of tape-outs.

MAGMA