Tuning EDA flows (using TCL)

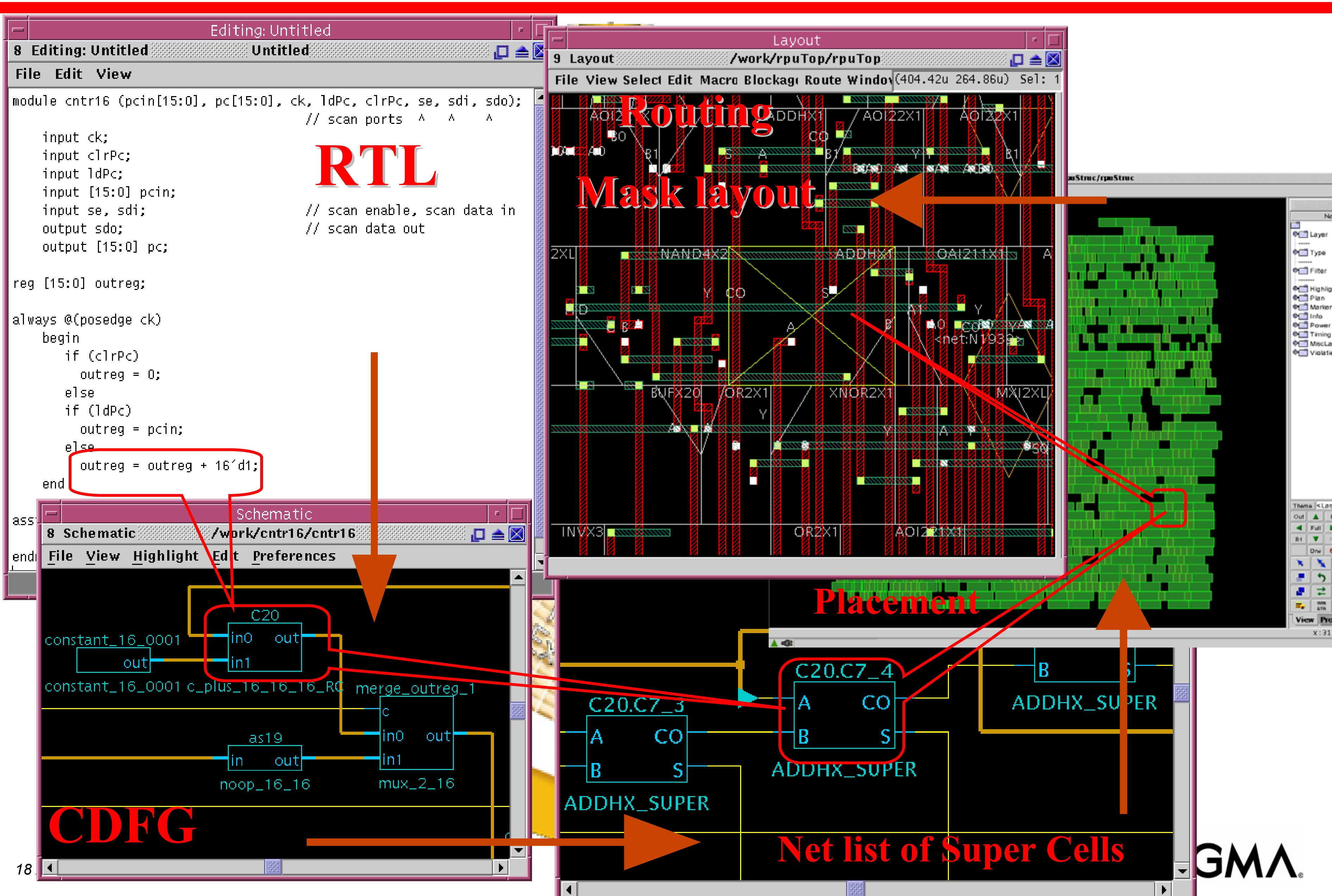
Patrick Groeneveld
Chief Technologist, Magma Design Automation
EDP 2008 - Monterey

Summary: Tuning EDA flows using TCL

- **The Good:**
  - TCL Enables well-integrated tool suites
- **The Bad:**
  - TCL is Easily chaotic, not fast, not pretty
- **The Ugly:**
  - The real problem is somewhere else!

MAGMA

From RTL to GDS2 in a single executable



Blast Rail power analysis steps in magma flow

Library data

VCD file

RTL synthesis

net list

Activity annotation & propagation

Slews from the built-in timer/extractor

Power Analysis

Power consumption report

Floor planning

Power infrastructure generation (rails, mesh)

floor plan with or without placed gates

Rail network extraction

Current & voltage drop calculation

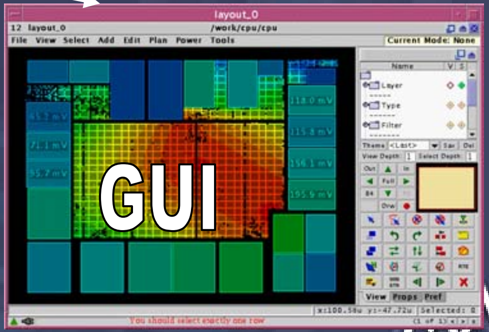
Voltage & current sources, resistances

What-if??

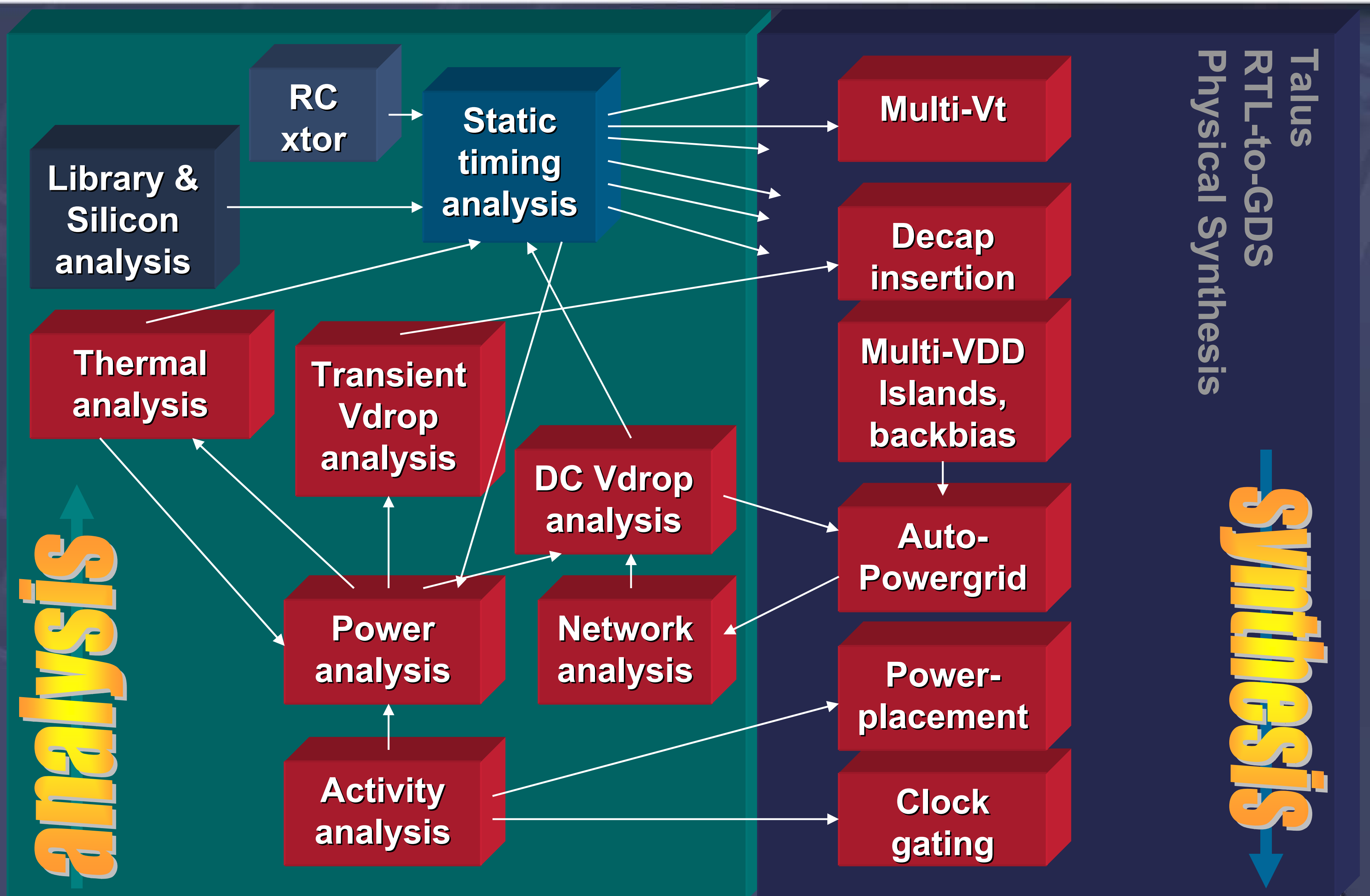
Voltage drop induced delay

Physical synthesis and optimization flow

Voltage drop and EM textual reports



Complex tool interactions without disk access



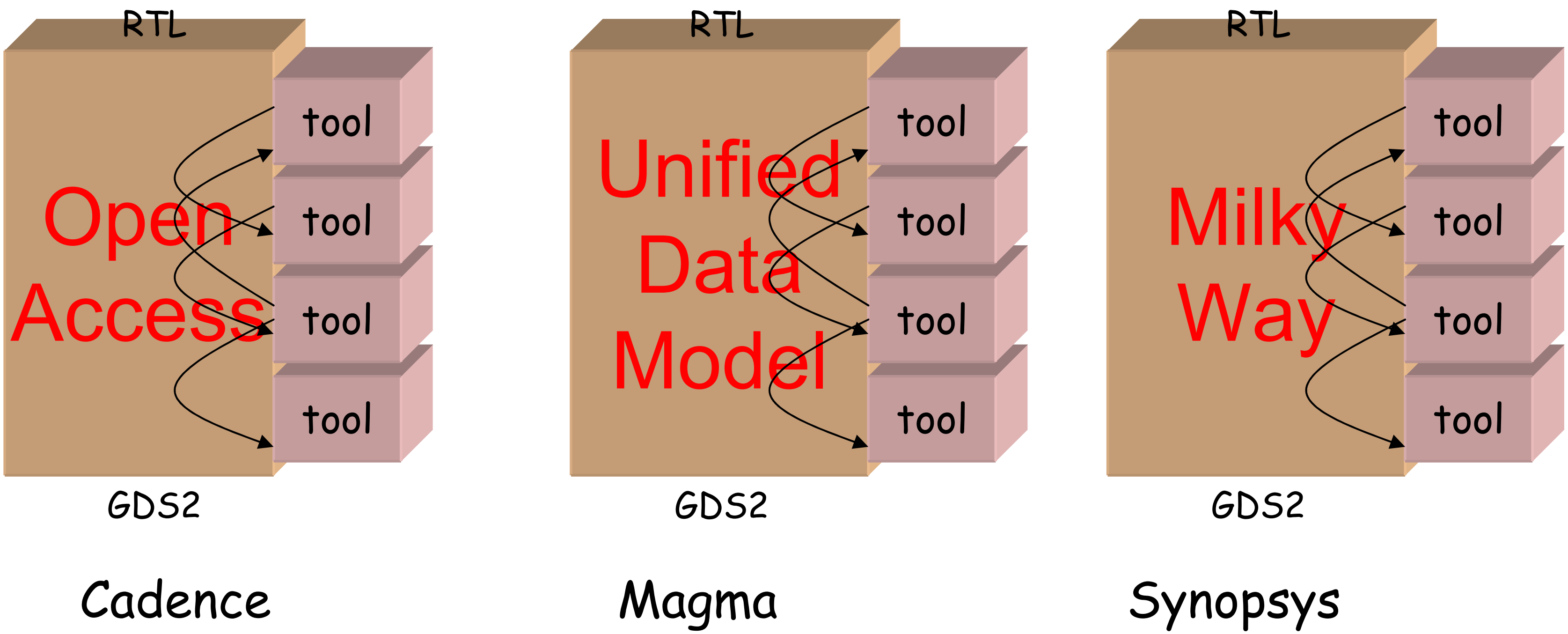
My view on Interoperability and standardization (Not necessarily Magma's official view)

Interoperability is like peace on earth

So… is OA the UN?

MAGMA

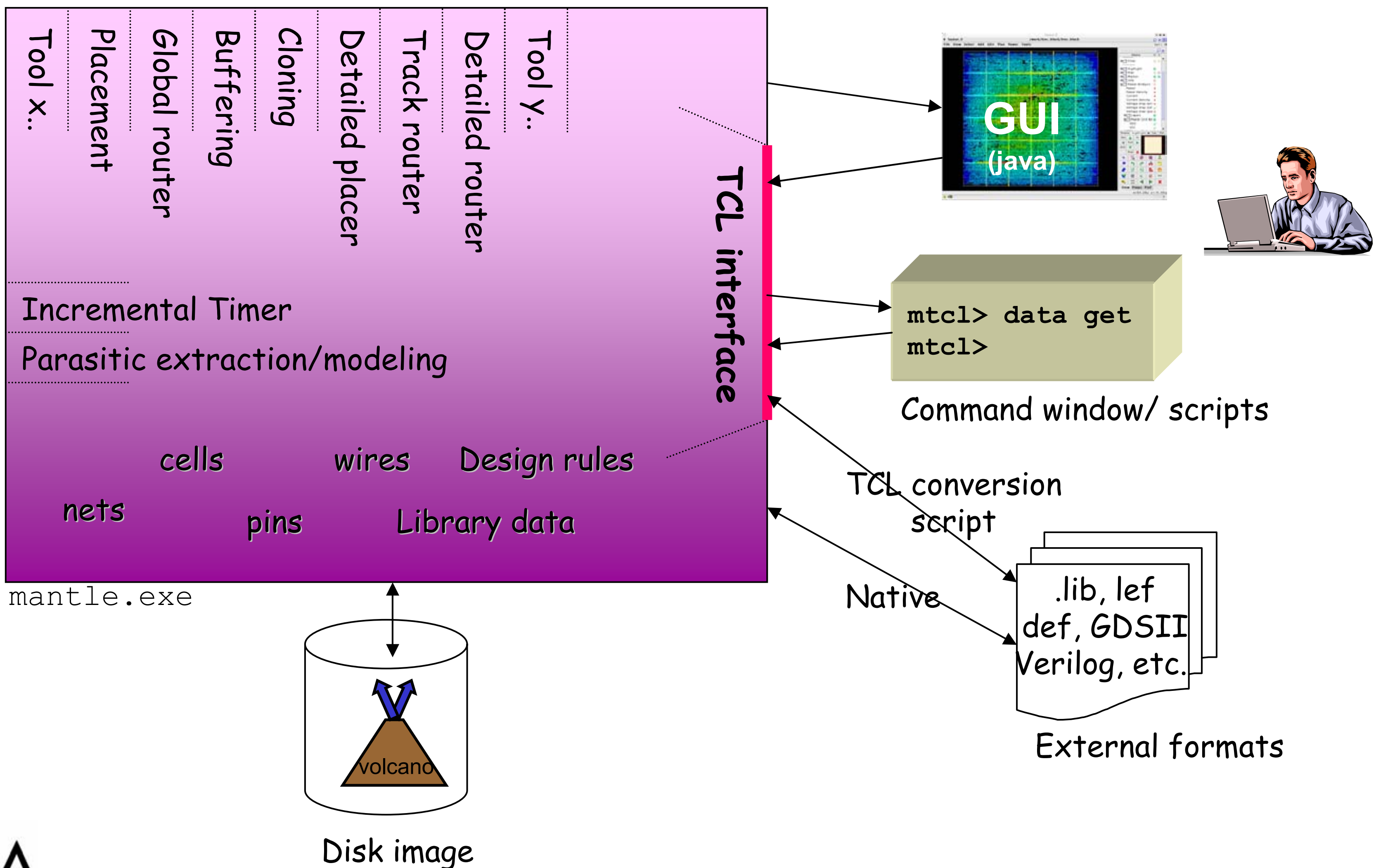
MIX and MATCH does not work!



- Plug = easy (formats are not so relevant)
- Play = hard (What does it mean? how does it perturb the flow?)

- I'm in the 'play' business
- Standardization may have it used, but its not here!

General architecture



MAGMA

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
```

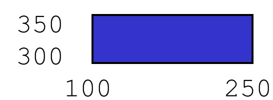
MAGMA

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

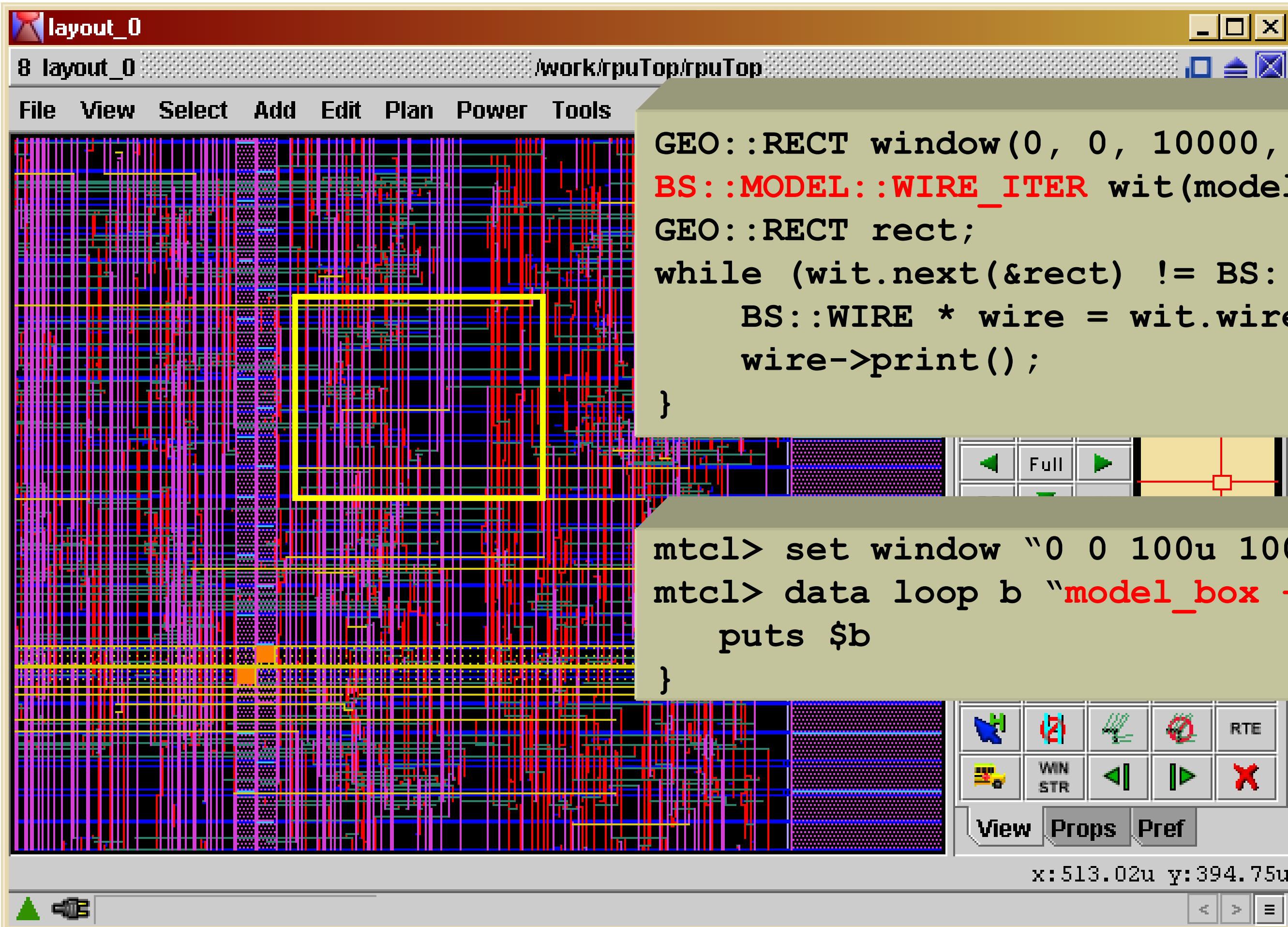
Metal 3 global routing segment of the net n1244

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

Cell c32 (and its placement)

MAGMA

Getting the wires in a window




```
GEO::RECT window(0, 0, 10000, 10000);
BS::MODEL::WIRE_ITER wit(model->basalt(), window);
GEO::RECT rect;
while (wit.next(&rect) != BS::NOT_A_LAYER) {
     BS::WIRE * wire = wit.wire(); // 0 if virtual
     wire->print();
}
```

```
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.

TCL: the good and the bad

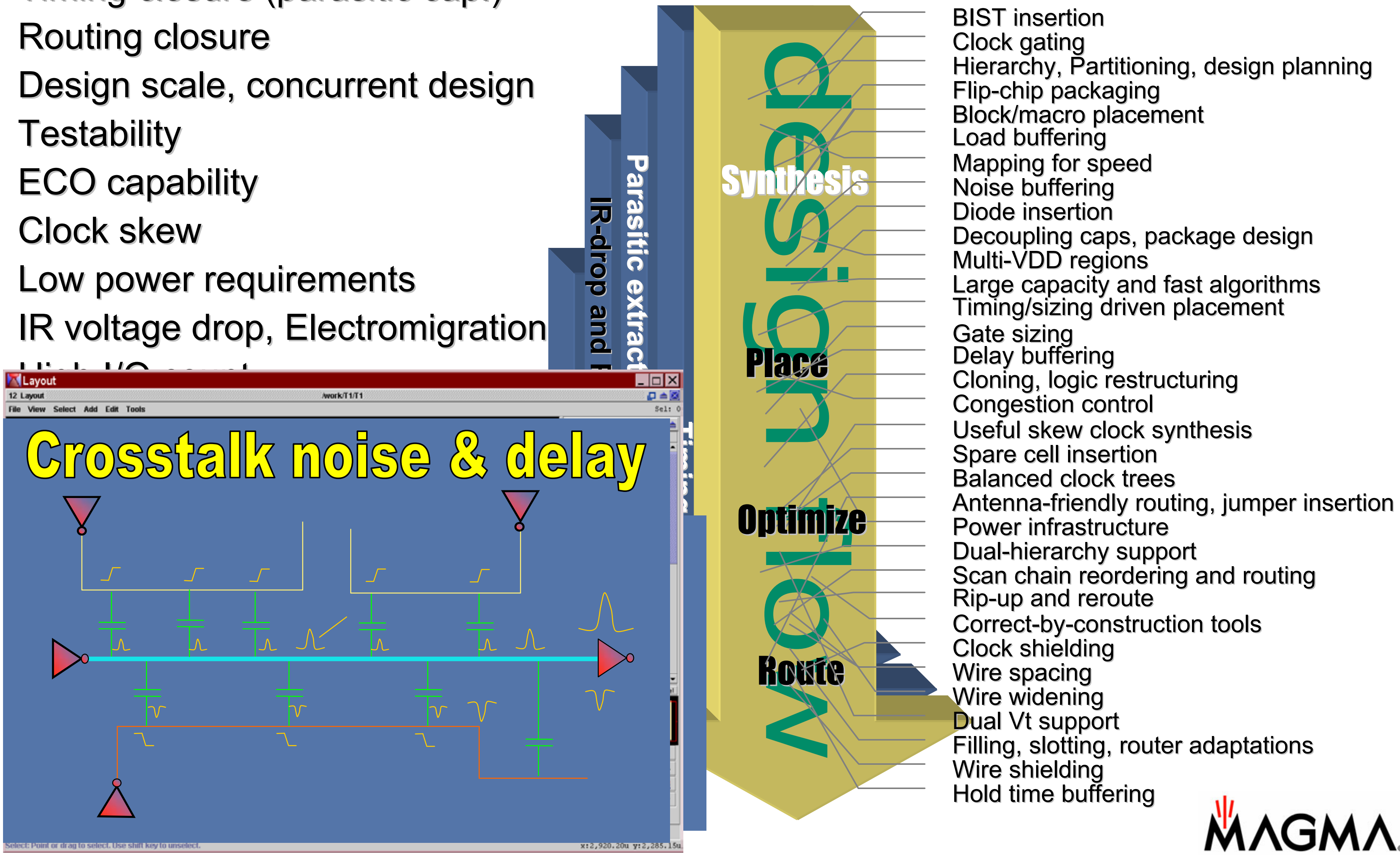
- The good:
  - Easy to learn and comprehend
  - Deep integration with data model
- The bad:
  - Slow compared to python
  - No easy integration of binary tools
  - Does not encourage clean code
  - A single typo can kill a run.

And now for the ugly..

MAGMA

The anatomy of a Physical Synthesis flow

Timing closure (parasitic cap.)
Routing closure
Design scale, concurrent design
Testability
ECO capability
Clock skew
Low power requirements
IR voltage drop, Electromigration



MAGMA

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
```

The truth about physical synthesis

Synthesis Algorithms do only *one* thing well
- Cannot handle multiple objectives
- System is easily over-constrained

Algorithms must use *inaccurate models* of the physical reality

Algorithmic steps do things that could cause problems at later steps

We often need to start over iterate to recover such errors

The ABC of a well-engineered IC design flow

A: Avoid

Detect specific problem patterns early, fix them
- Relies on prediction which
- does not have to be extremely accurate.

B: Build

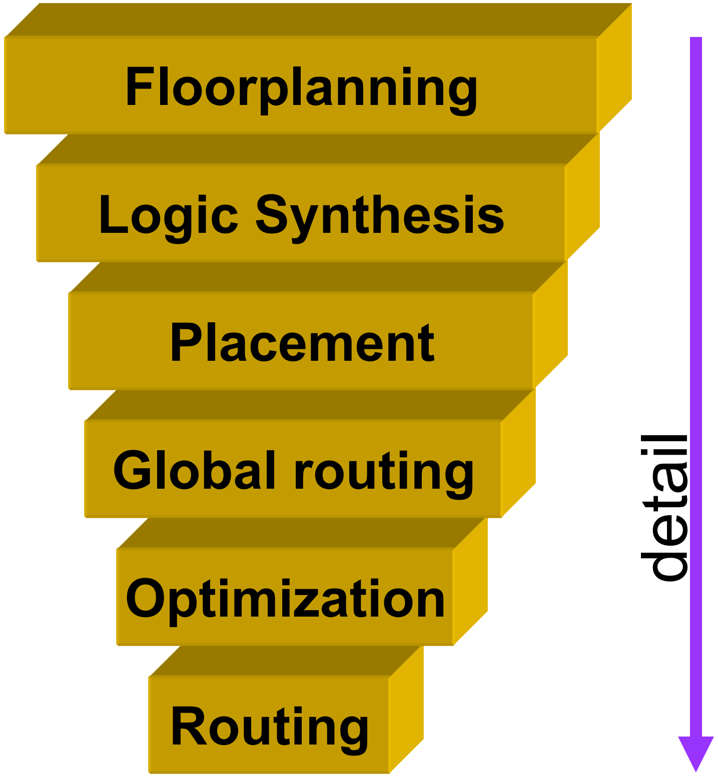
Synthesize using an algorithm on a simplified model.
- Capture 1st order effect of problem as objective.
- Shoot in the ball park, and hope for the best.

C: Correct

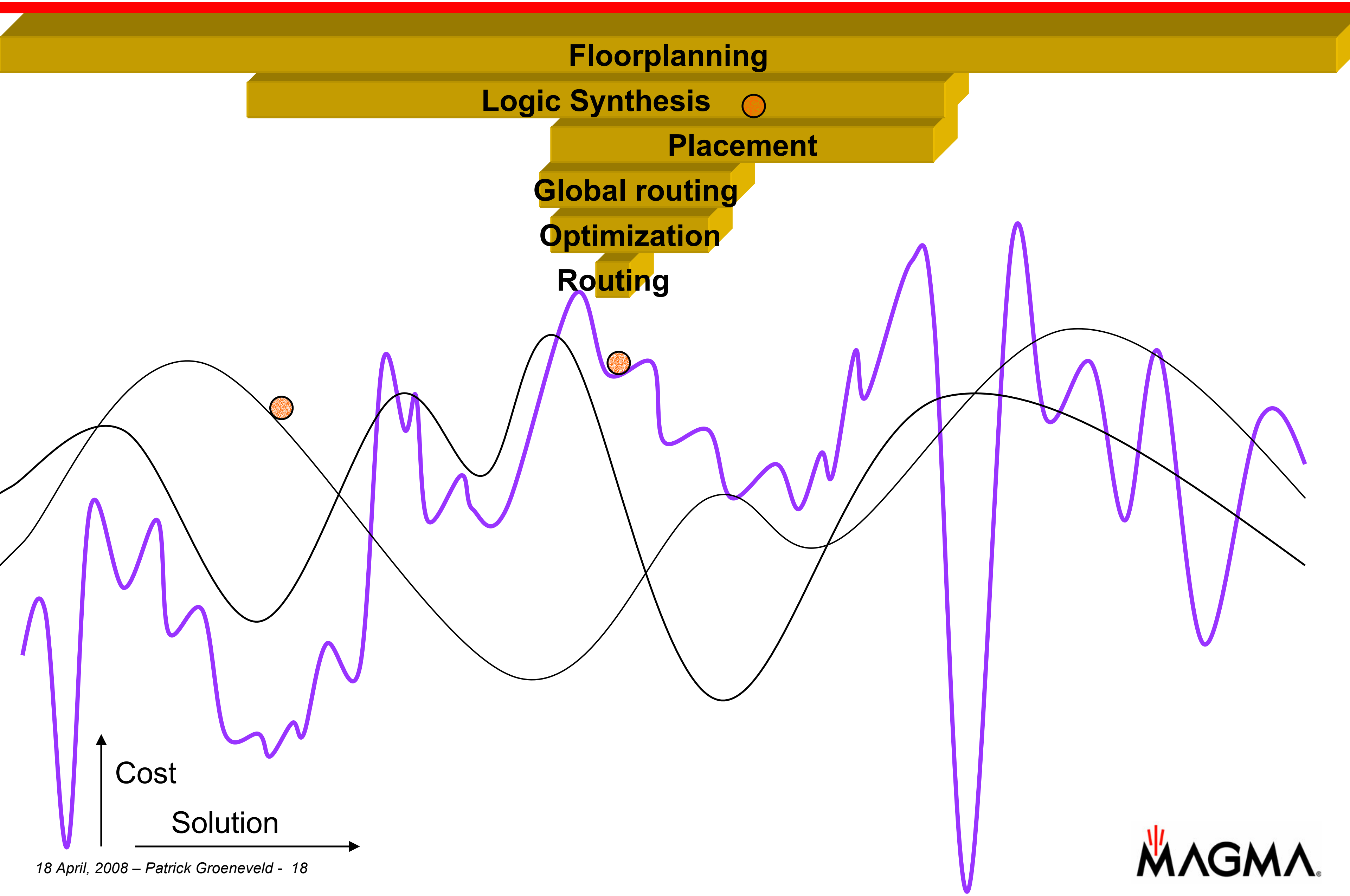
Perform accurate analysis, detect remaining problems and fix any problems by local modifications (ECO).
- This is typically slow and it
- might not work.
- If its real bad, iterate back to step A or B

Guiding principles during Physical Synthesis

- **Stepwise refinement**
  - Use a number of build steps, each fixing an objective and adding detail
- **Avoid Correction iteration like the plague**
- **Use *in*accurate analysis**
  - Ballpark is enough, You're far off anyway



Converging to a local optimum in a tool flow



MAGMA

The EDA flow as a pachinko machine

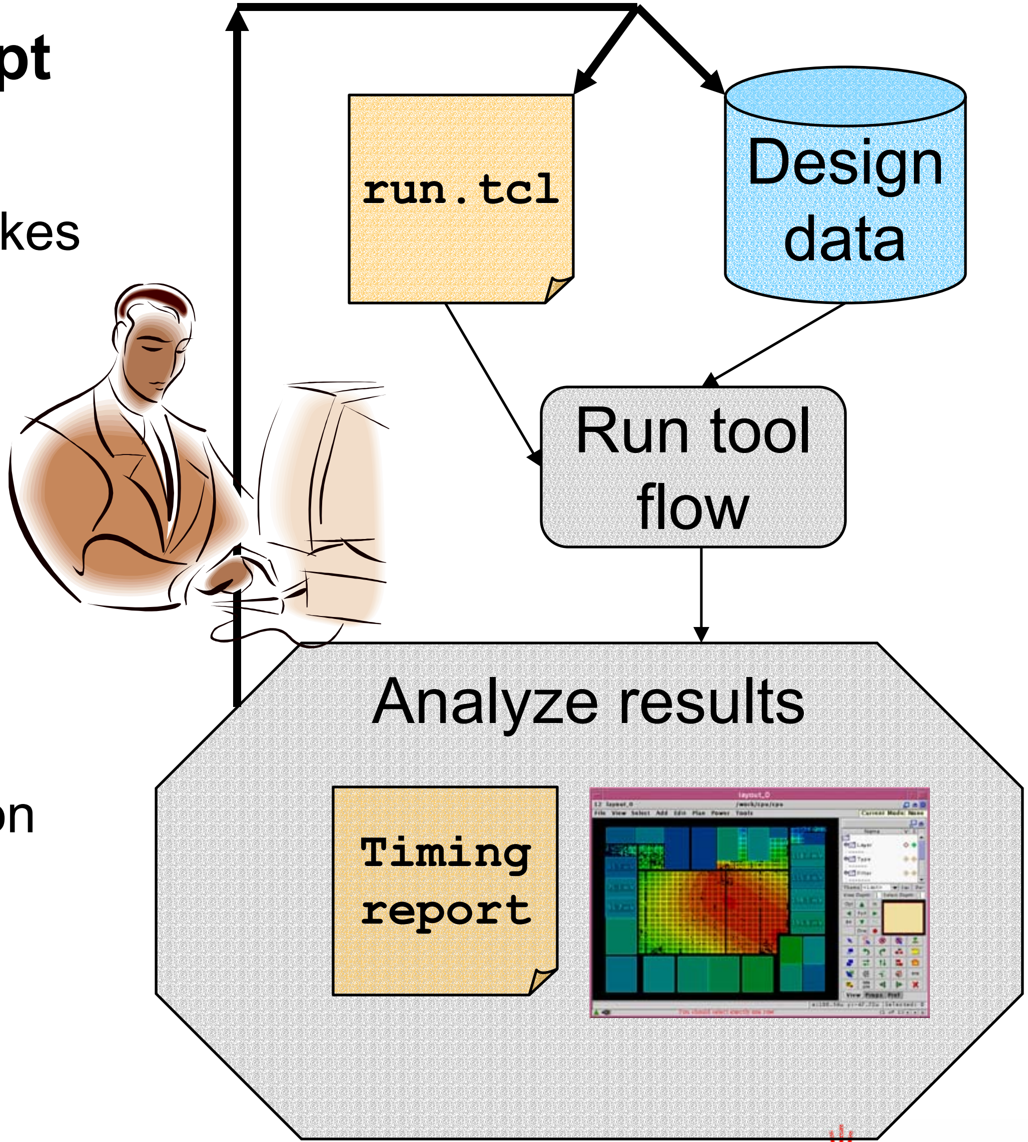
- **Run flow:**
  - End up an one of the local optima.
- **Re-run:**
  - typically get same results
    - (Multi-processing alert!!)
- **Re-run with small change**
  - Could be huge difference
- **Changes:**
  - Irrelevant order changes
  - Additional steps/algorithms
  - Changing constraints, tuning, etc.

- **Good/bad results depend on:**
  - 'ease' of the design
  - Flow set-up/tuning
  - Design structure (e.g. data paths)
  - **Coincidence**

MAGMA

"Only a bad carpenter blames his tools", *NOT*

- **Tuning of the TCL script**
- **First time:**
  - Poor local optimum, mistakes
- **Tune flow+data**
  - Better local optimum.
- **But:**
  - Loop is slow
  - Tools talks gibberish
  - Result depend too much on experience of engineer.
  - Hacks are design-specific



MAGMA

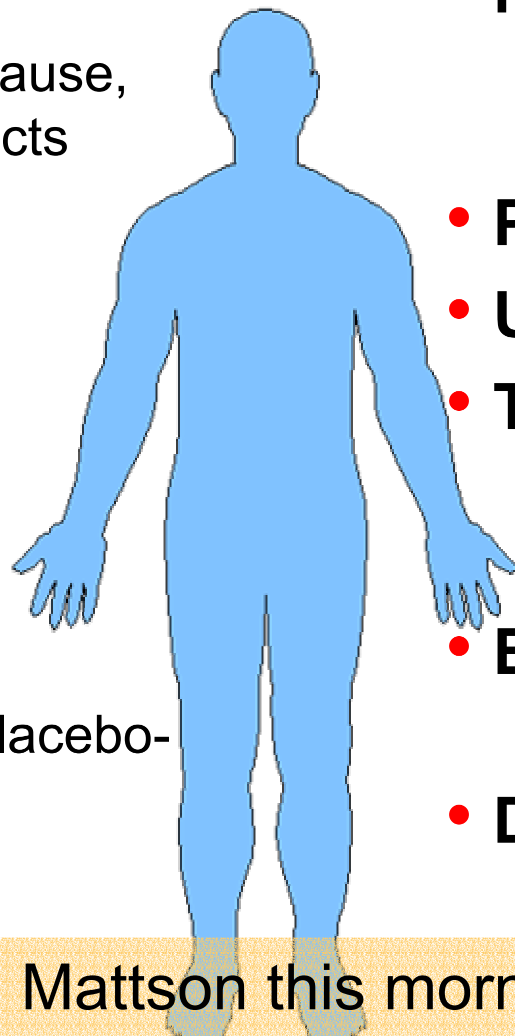
EDA Flow tuning for best out-of-the-box results

- **Goal:**
  - Improving the chance of ending up in a good local optimum. (that is: move the mean for better QOR)
- **That requires:**
  - Good understanding of cause, actions, side-effects
  - Statistical evidence of efficacy
- **Issue:**
  - Effects and side-effects are hard to predict
  - How to distinguish design-specific noise from real improvements?

Not easy!

MAGMA

Analogy with the medical field



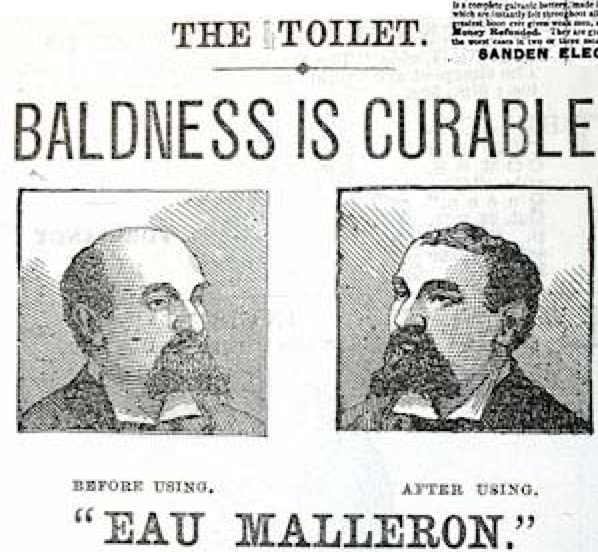
- **New drug**
  - Biological model of cause, actions and side-effects
- **Develop it**
- **Test tube test**
- **Test on animals**
  - Efficacy,
  - side effects
- **Clinical trials**
  - Large double-blind placebo-controlled tests
- **FDA-approval**

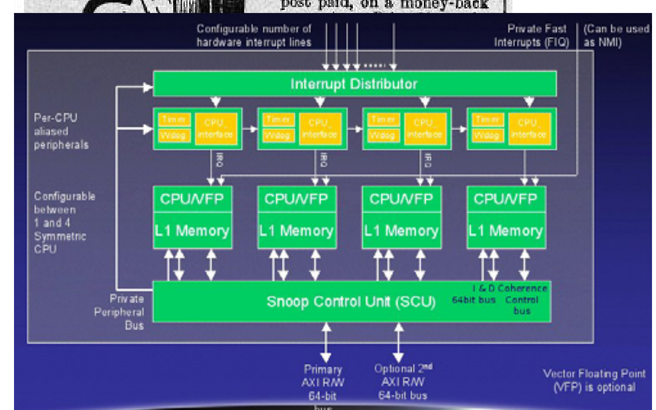
- **New flow component**
  - Based on electrical/ physical plausability
- **Program it (C++/TCL)**
- **Unit test**
- **Test on small testcases**
  - Debug program
  - Efficacy, side effects
- **Beta test**
  - Hope that customers use it
- **Deployment**
  - Go for it!

Tim Mattson this morning:
“Engineers: think it, build it, demo it, declare victory”

Using skeptical wisdom from the medical field

- **Unproven methods are “Quackery”**
  - Examples: homeopathy, **multiprocessor throughput scaling**, chiropractic, **structured placement**, acupuncture, **DFM**, holistic/herbal supplements, **plug and play EDA interoperability**, probiotics, etc. etc.





MAGMA

Using skeptical wisdom from the medical field (2)

- **“Humans are amazingly good at self-deception”**
  - This looks soooo good, therefore this *must* work
- **“If it has no side effects, it probably has no effects either”**
  - Example: improving temperature gradients is gonna cost you! So is improving yield. Are you really willing to pay based on the evidence?
- **“Do not confuse association with causation”**
  - “I took this airborne pill, and I did not get sick”
  - “I used this DFM optimizer, and the chip yields!
- **“The plural of ‘anecdote’ is ‘anecdotes’, *not* data”**
  - Result could be a random effect, or another side effect
  - No substitute for unbiased placebo-controlled tests
  - Only large data sets are statistically relevant

MAGMA

Conclusion for EDA: academic view

- **Weak empirical academic standards:**
  - Order of magnitude too few test cases
  - Test cases based on artificial data or flows
  - Many opportunities for **bias**
- **Reluctance to publish ‘negative results’**
  - Publication pressure encourages intellectual dishonesty
  - Comparisons/field tests are rare (or poor at best)
- **Most papers are *not* trustworthy**

MAGMA

Conclusion for EDA

- **EDA Business view:**
  - Totally allergic to negative results!
  - Too much focus on ad-hoc fixes/features rather than out-of-the-box
  - Desire to please customers, rather than fundamentally improve tool.
  - Results of secret 'bake-off' benchmarks are not fully analyzed

- **A more scientific approach would result in *significantly* better out-of-the-box**
  - Find local optimum that's closer to global one
  - Saving serious engineering effort.
  - Even given current set of algorithms