MAGMA®

The parallelism mirage

Patrick Groeneveld,
Chief Technologist, Magma Design Automation

EDP Monterey April 2009

Core challenge: Two-fold Complexity Increase

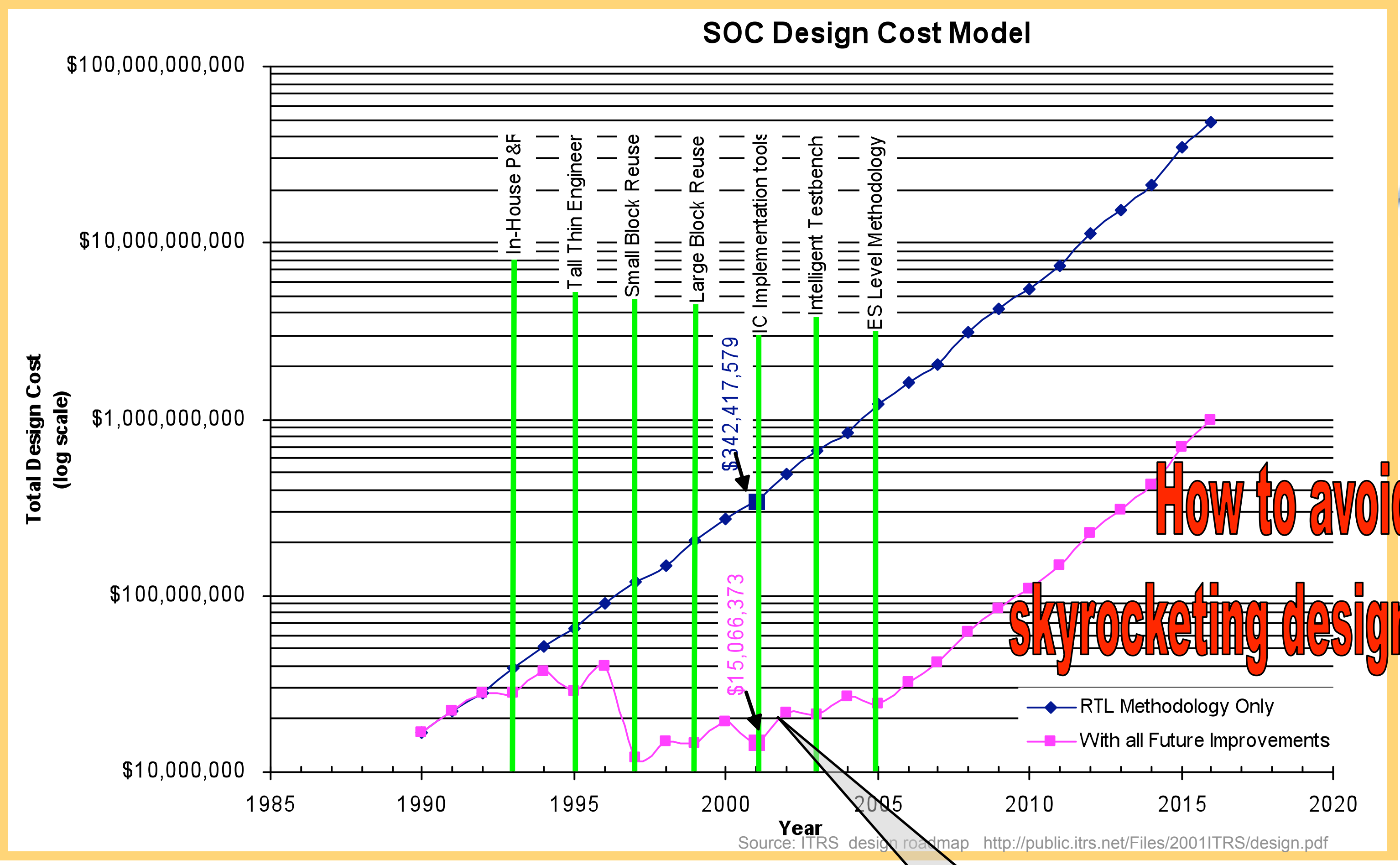
- **System complexity:**
  Dealing with the sheer size of the SoC
  - 1 Billion transistors, 100M+ gates

Exponential

- **Silicon complexity:**
  Dealing with the physics of manufacturing technology
  - Electrical parasitics.
  - Leakage & dynamic power
  - Process variability & manufactura

More design iterations

ITRS Roadmap: Design cost and Design Automation



Source: ITRS design roadmap http://public.itrs.net/Files/2001ITRS/design.pdf

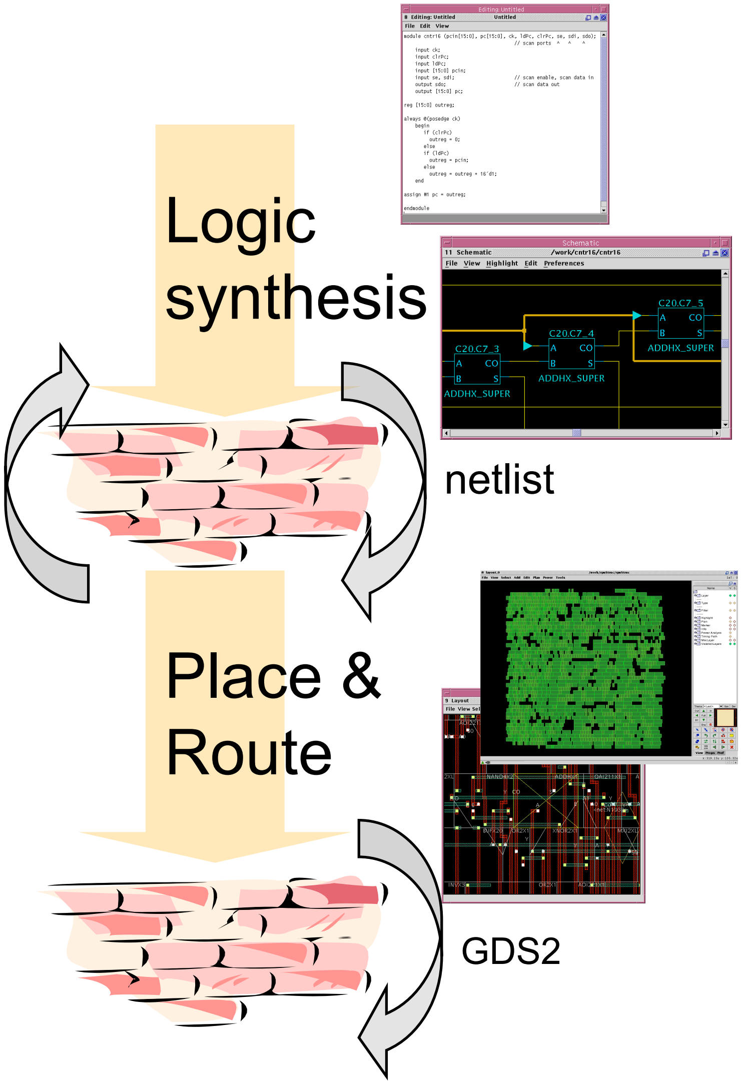
How to avoid skyrocketing design effort?

Physical Synthesis

April 13, 2009 – EDP 2009- 3

Time-warp back to the previous century

Design community 1997:
"**Mr. EDA:** ***tear down this wall!***"



```
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

assign #1 pc = outreg;

endmodule
```

Logic synthesis

netlist

Place & Route

GDS2

Fab

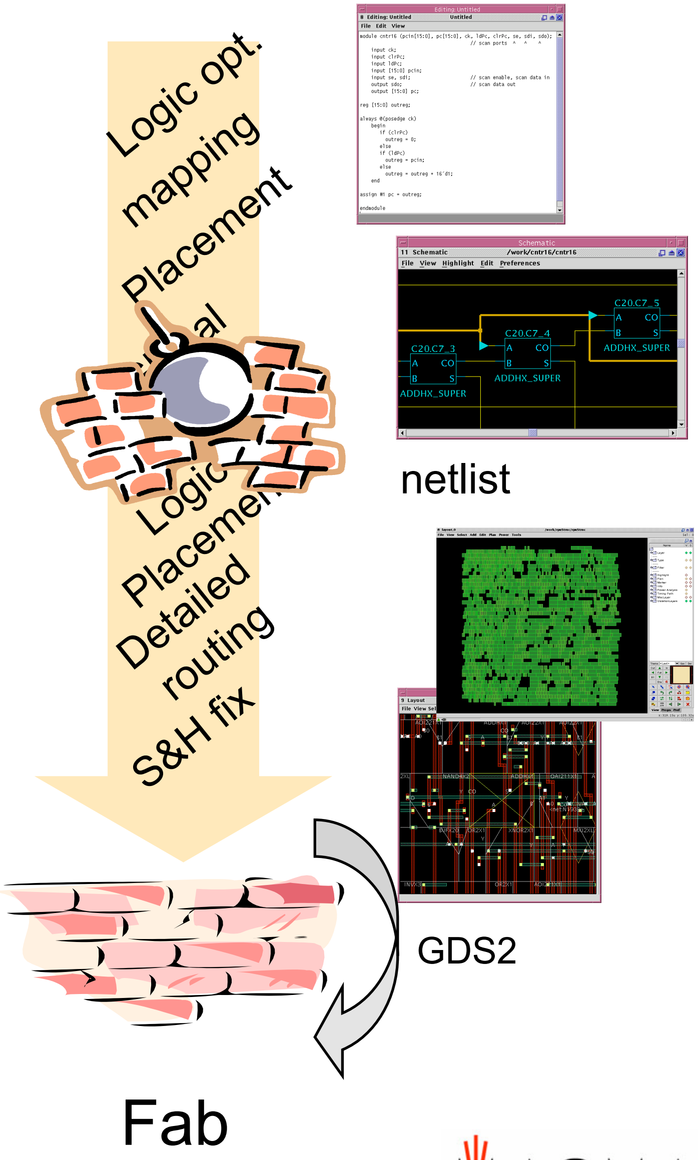
Magma in august 1997

Terra Bella Avenue, Mountain View, CA

The Magma revolution

- **Replaced wall by a ‘smooth’ design flow**
  - Gradual transition in a *single* executable

- **Avoided and simplified timing closure iteration.**
  - It’s the Flow, stupid!



```
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

assign #1 pc = outreg;

endmodule
```

ITRS 2007: recommendations

International Technology Roadmap for Semiconductors

"Ideally, the time-wasting **iteration between logic and layout synthesis** in today's design methodologies could be eliminated by fusing these stages to simultaneously optimize the logical structure as well as layout of a circuit."

ITRS Design Roadmap 2007

"To avoid excessive guardbanding due to poor estimates, logical design and eventually system-level design must become more closely linked with physical design."

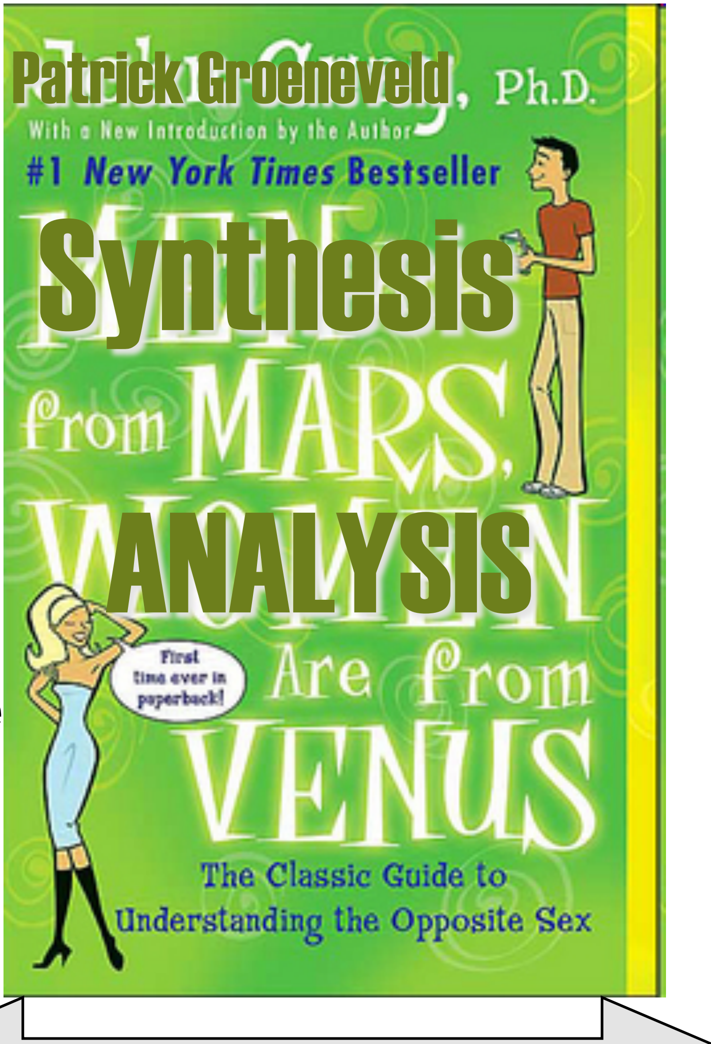
ITRS Design Roadmap 2007

Synthesis is from Mars, Analysis is from Venus

- Sign-off tools:
- Verification, extraction, STA, spice, DRC, LVS

- Highly accurate
- Big and slow

- Is the ‘whiner’


Patrick Groeneveld, Ph.D.
With a New Introduction by the Author
#1 New York Times Bestseller
Synthesis
from MARS,
ANALYSIS
Are from
VENUS
First time ever in paperback!
The Classic Guide to Understanding the Opposite Sex

- Implementation tools:
- RTL synthesis, Placement, Routing, Optimization, Humans

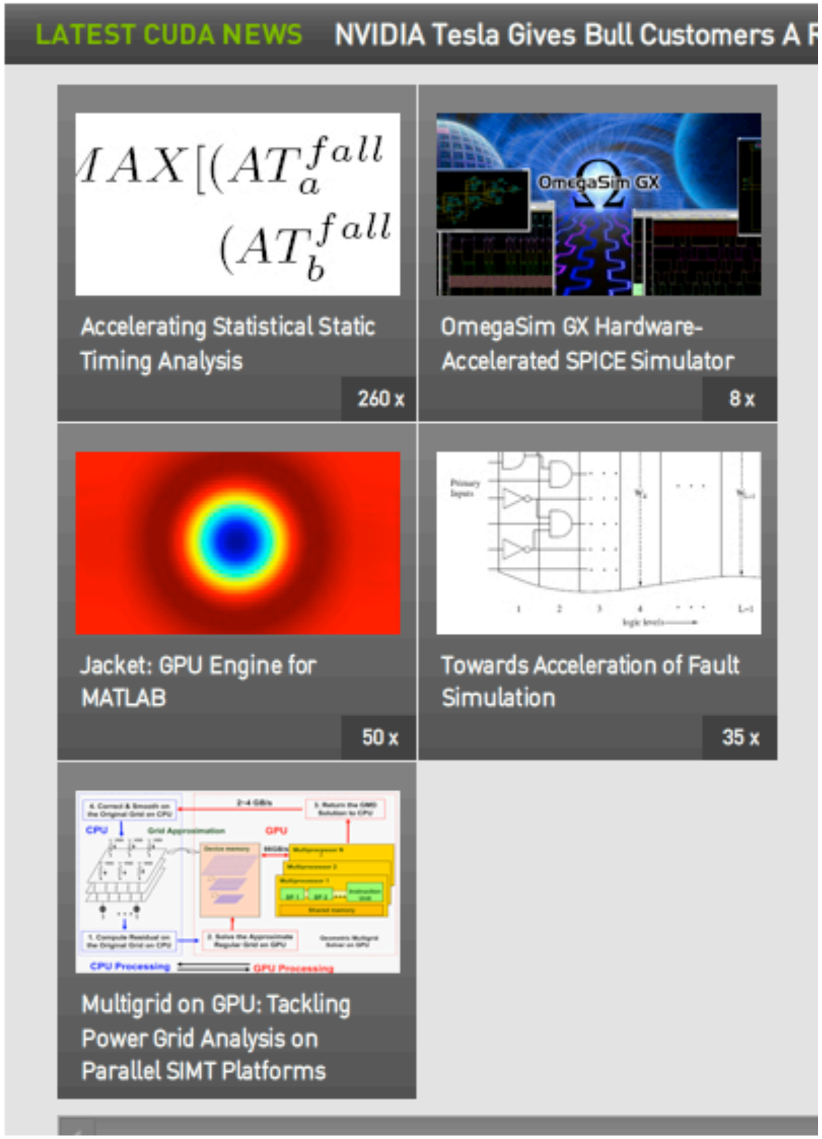
- Poor accuracy
- Lean, mean

- Is the ‘hacker’

Need to make this work

MAGMA

CUDA & EDA: What’s wrong with this picture??



LATEST CUDA NEWS NVIDIA Tesla Gives Bull Customers A F

Accelerating Statistical Static Timing Analysis

260 x

OmegaSim GX Hardware-Accelerated SPICE Simulator

8 x

Jacket: GPU Engine for MATLAB

50 x

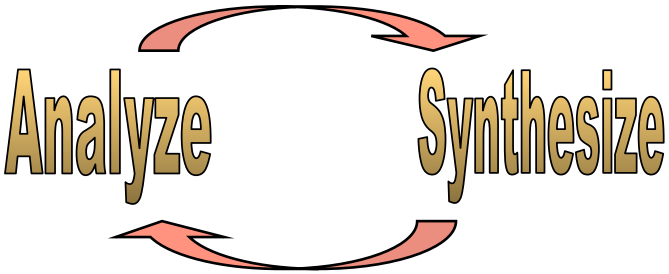
Towards Acceleration of Fault Simulation

35 x

Multigrid on GPU: Tackling Power Grid Analysis on Parallel SIMT Platforms

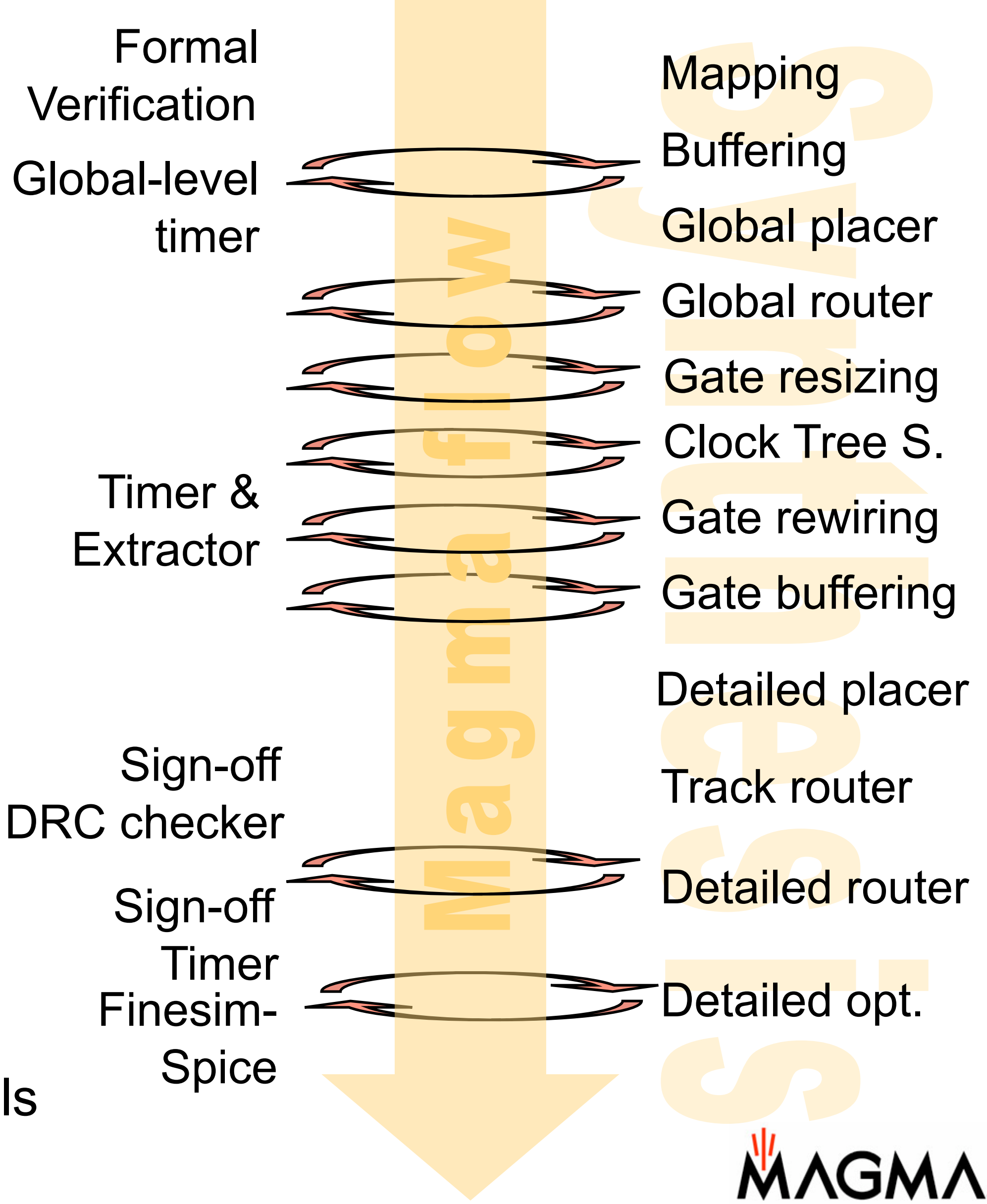
How IC design really works…

Iterate:


Analyze

Synthesize

- **Avoid loops:**
  - Correct-by-construction methods
  - ABC flow
- **Speed up loop by:**
  - Reducing analysis accuracy
  - Tricks: incremental analysis
  - **Running tasks in parallel**
  - Take away walls between tools


Formal Verification

Global-level timer

Timer & Extractor

Sign-off DRC checker

Sign-off Timer

Finesim-Spice

Mapping

Buffering

Global placer

Global router

Gate resizing

Clock Tree S.

Gate rewiring

Gate buffering

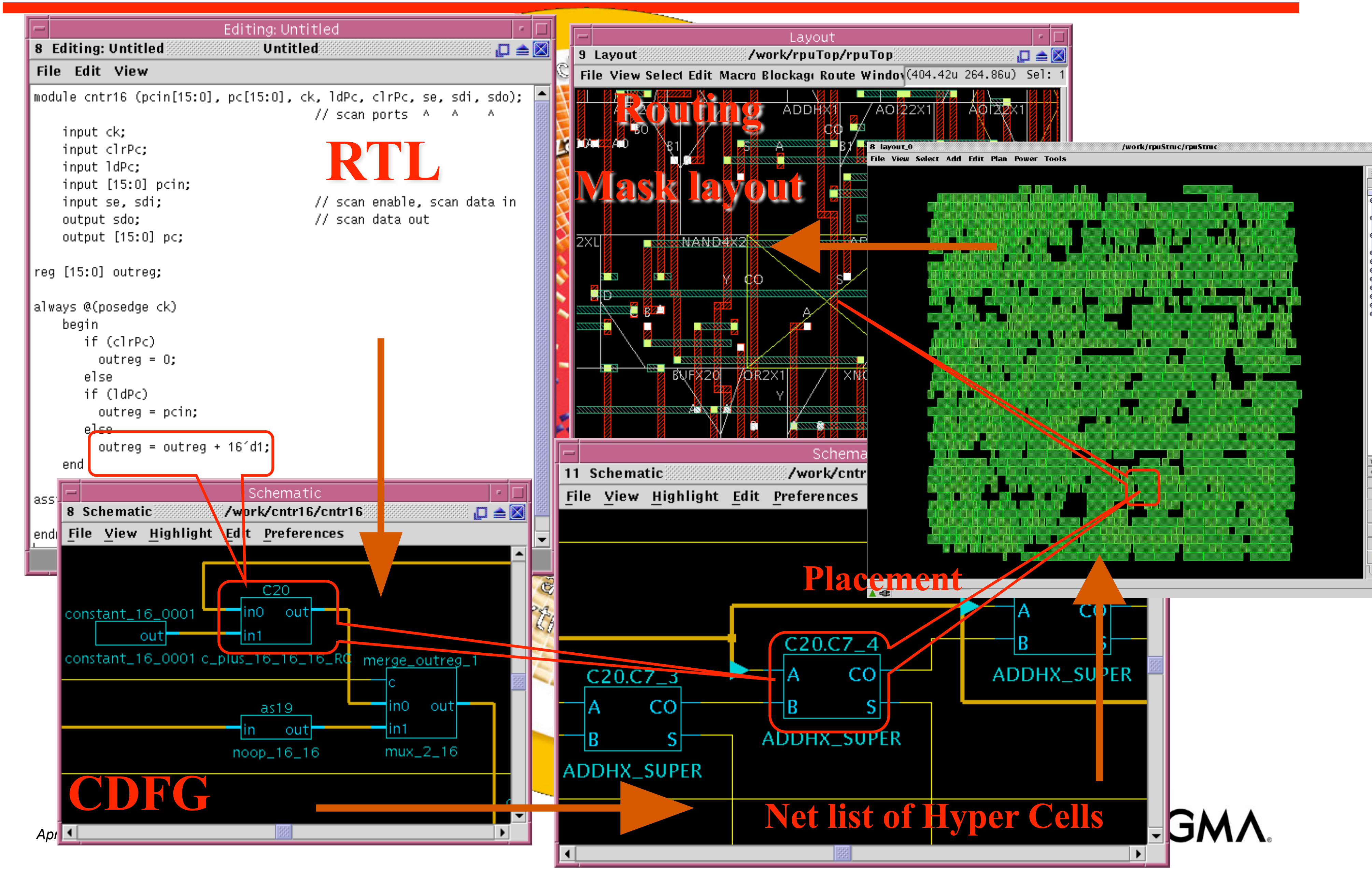
Detailed placer

Track router

Detailed router

Detailed opt.

Data model greases the wheels



```
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
endı
```

RTL

CDFG

Net list of Hyper Cells

Placement

Routing

Mask layout

Editing: Untitled
8 Editing: Untitled Untitled
File Edit View
Schematic
8 Schematic /work/cntr16/cntr16
File View Highlight Edit Preferences
C20
in0 out
in1
constant_16_0001
out
constant_16_0001 c_plus_16_16_16_RC
merge_outreg_1
c
in0 out
in1
mux_2_16
as19
in out
noop_16_16
Layout
9 Layout /work/rpuTop/rpuTop
File View Select Edit Macro Blockage Route Window (404.42u 264.86u) Sel: 1
ADDHX1
AOI22X1
AOI22X1
CO
B0
B1
A0
S
A
B1
2XL
NAND4X2
Y
CO
D
B
A
BUFX20
OR2X1
XNO
Y
8 layout_0 /work/rpuStruc/rpuStruc
File View Select Add Edit Plan Power Tools
Schema
11 Schematic /work/cntr
File View Highlight Edit Preferences
C20.C7_3
A CO
B S
ADDHX_SUPER
C20.C7_4
A CO
B S
ADDHX_SUPER
A CO
B S
ADDHX_SUPER

Design is a trade-off

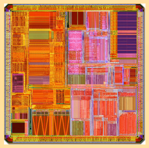
Speed

Leakage power

Area

Yield, robustness

Corners



Run time

Human design effort

Dynamic power

Routability

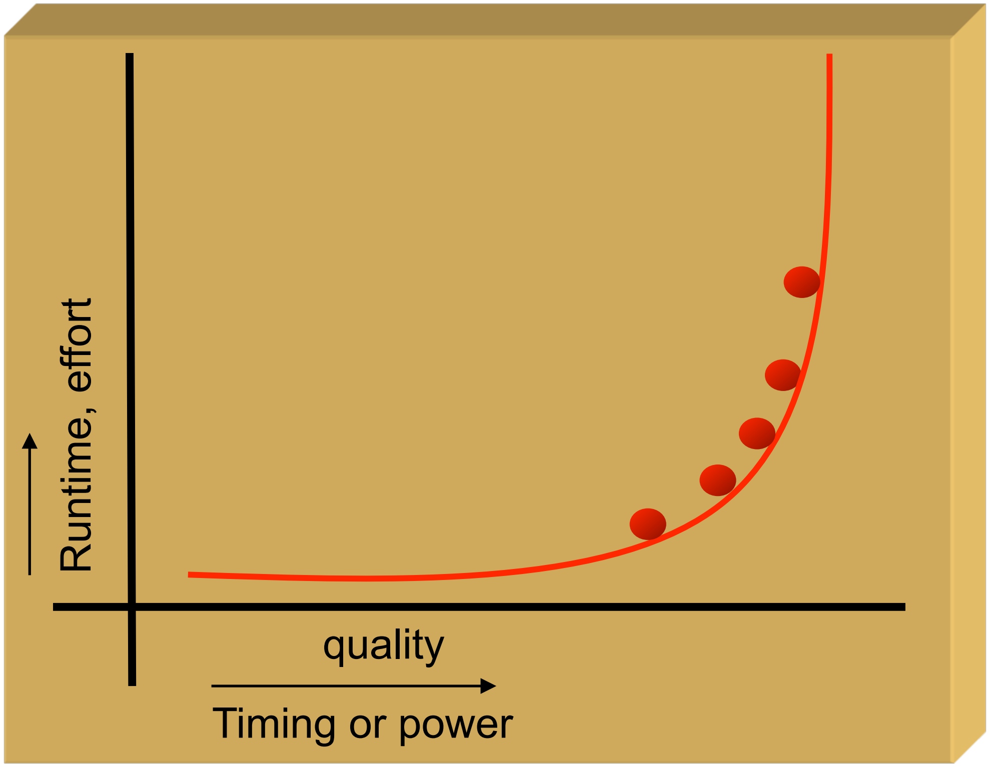
Manufacturability

The nature of the IC design ‘beast’

Fact:

Pushing *all* objectives costs:

- **Human design effort and**
- **Run time**

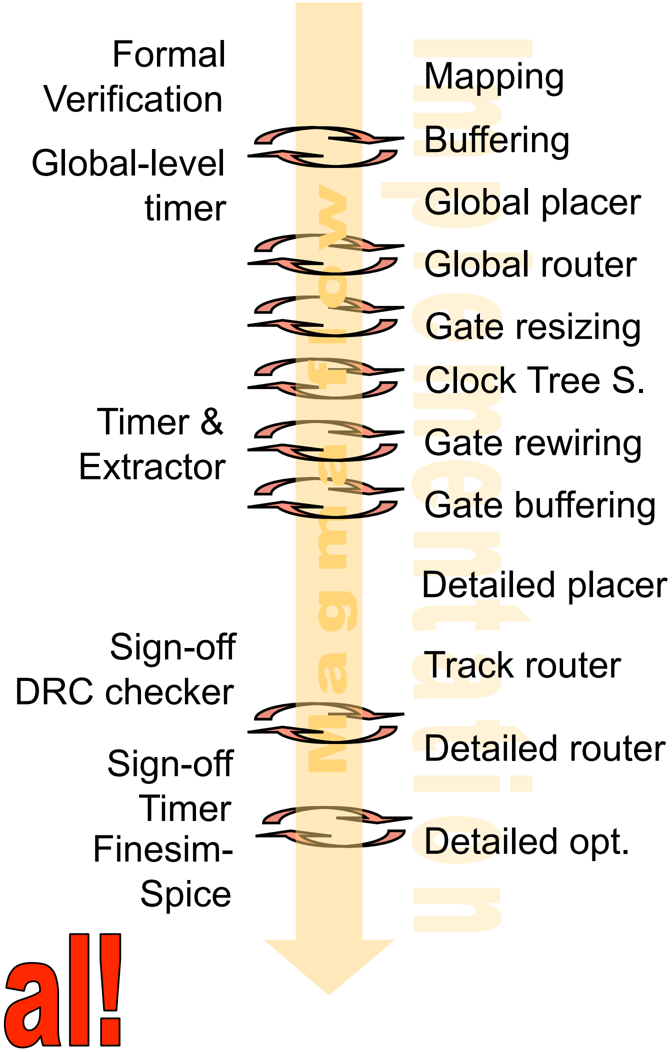


MAGMA

Building a design flow for Multiple objectives

Fact:

No single implementation step can deliver the optimum trade-off at 32nm



Formal Verification

Global-level timer

Timer & Extractor

Sign-off DRC checker

Sign-off Timer Finesim-Spice

Mapping

Buffering

Global placer

Global router

Gate resizing

Clock Tree S.

Gate rewiring

Gate buffering

Detailed placer

Track router

Detailed router

Detailed opt.

Optimal is not Optimal!

MAGMA

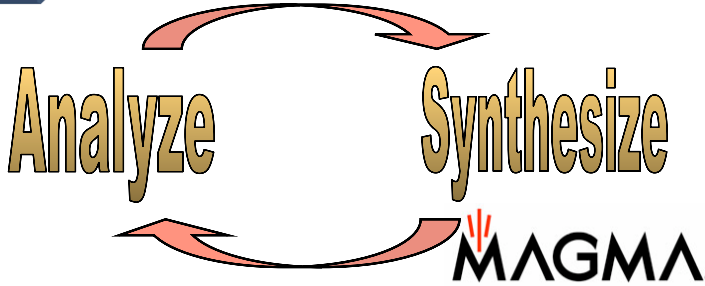
The truth about RTL2GDS2 Design Automation

Zero tolerance for sloppiness

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

Algorithmic steps do things that could cause problems at later steps

Algorithms must use *inaccurate models* of the physical reality



The ABC of a well-engineered design flow

A: Avoid

Predict problems, avoid many 'by construction'

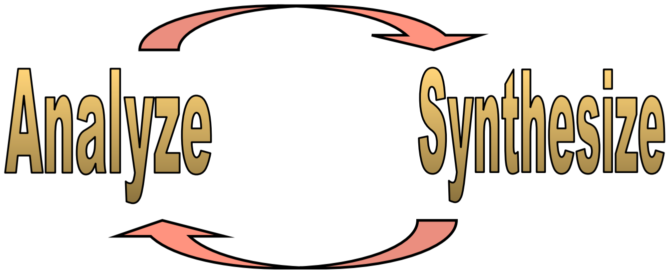
B: Build

Synthesize using an algorithm

Implementation is from Mars!

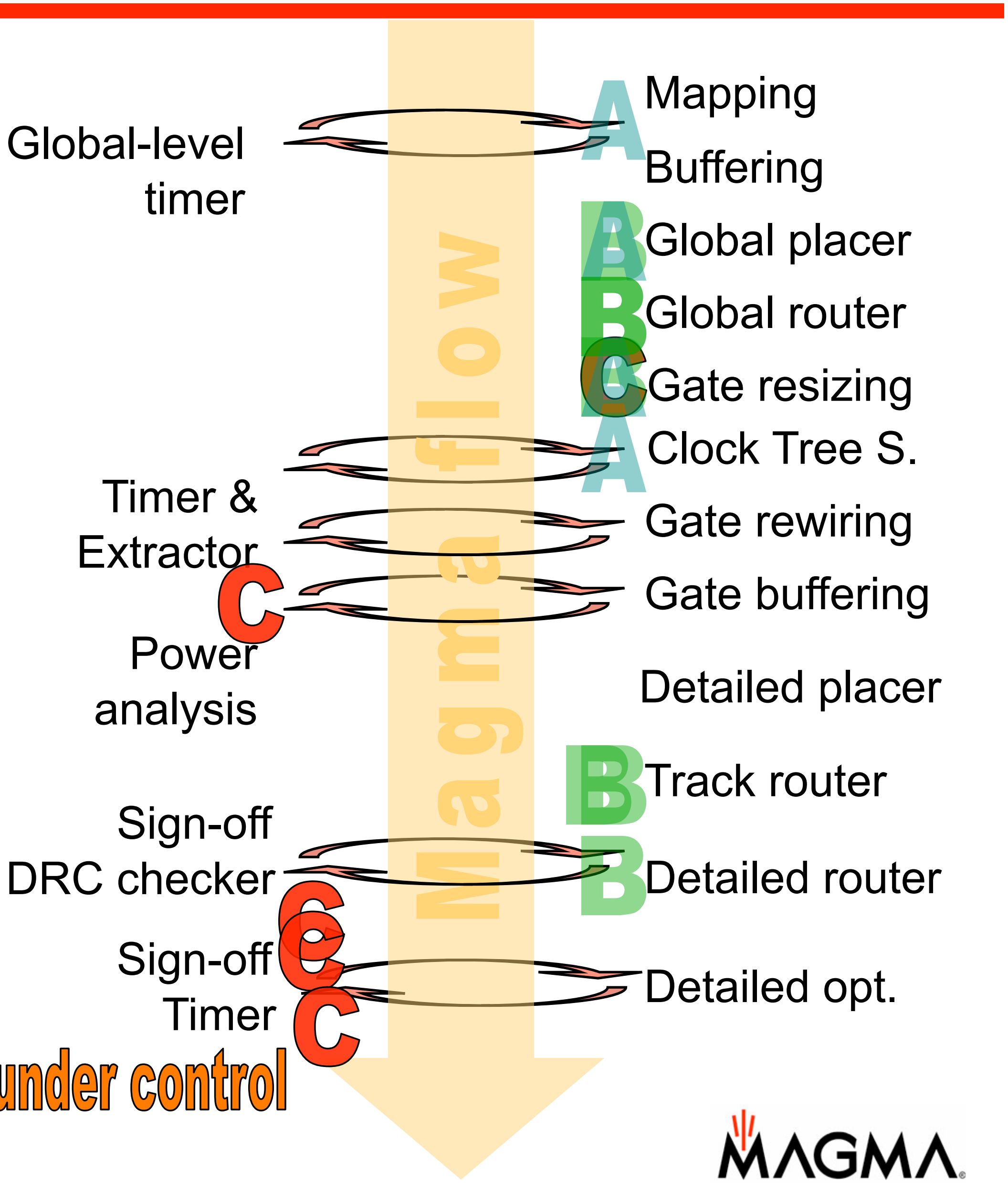
C: Correct

Fix each objective by incremental modifications (ECOs).



Many ABC's

- **Timing closure:**
  - Pre-Buffering, logic optimization
  - Mapping, placement
  - Gate level optimization
- **Routing closure:**
  - congestion driven placement
  - Global, track & detailed routers
  - Incrementally fix DRCs
- **Variability robustness:**
  - add margin, robust clock trees
  - Gate-level optimization
  - Fix setup and hold violations for each corner
- **Crosstalk:**
  - Oversize weak drivers, shielding
  - X-talk avoiding global and track routing
  - ECO-level x-talk fixing



Global-level timer

Timer & Extractor

Power analysis

Sign-off DRC checker

Sign-off Timer

Magma flow

Mapping

Buffering

Global placer

Global router

Gate resizing

Clock Tree S.

Gate rewiring

Gate buffering

Detailed placer

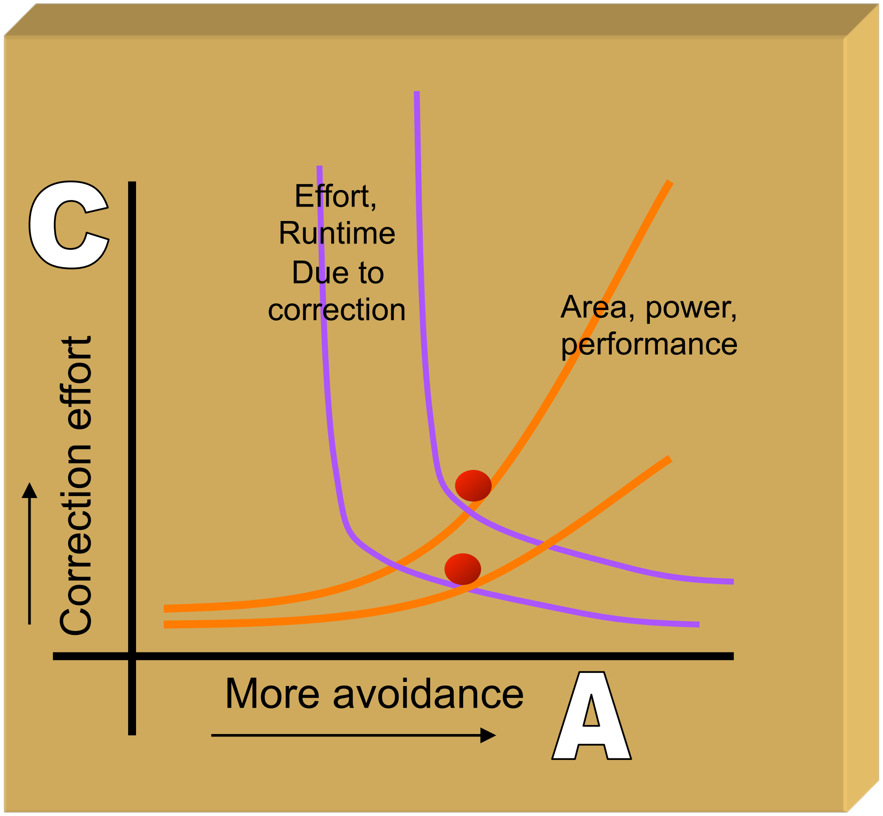
Track router

Detailed router

Detailed opt.

Issue: Keeping Correction steps under control

Trading off Avoidance and Correction



Productivity increase though:

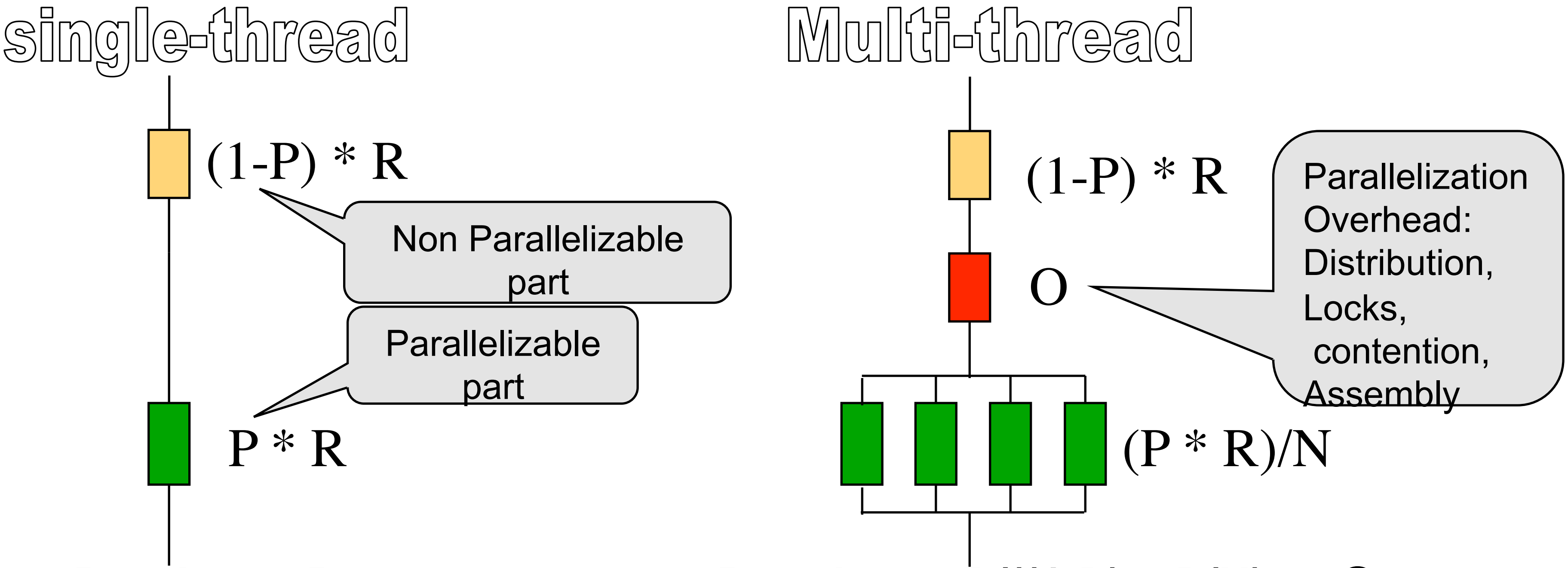
- **Effectively use parallel hardware.**

- **Intelligent avoidance**
  - Early in flow
  - Center process corners

- **Reduce cycles**
  - Converge faster
  - SITL

MAGMA

Amdahl’s law: Why parallelization gain tapers off



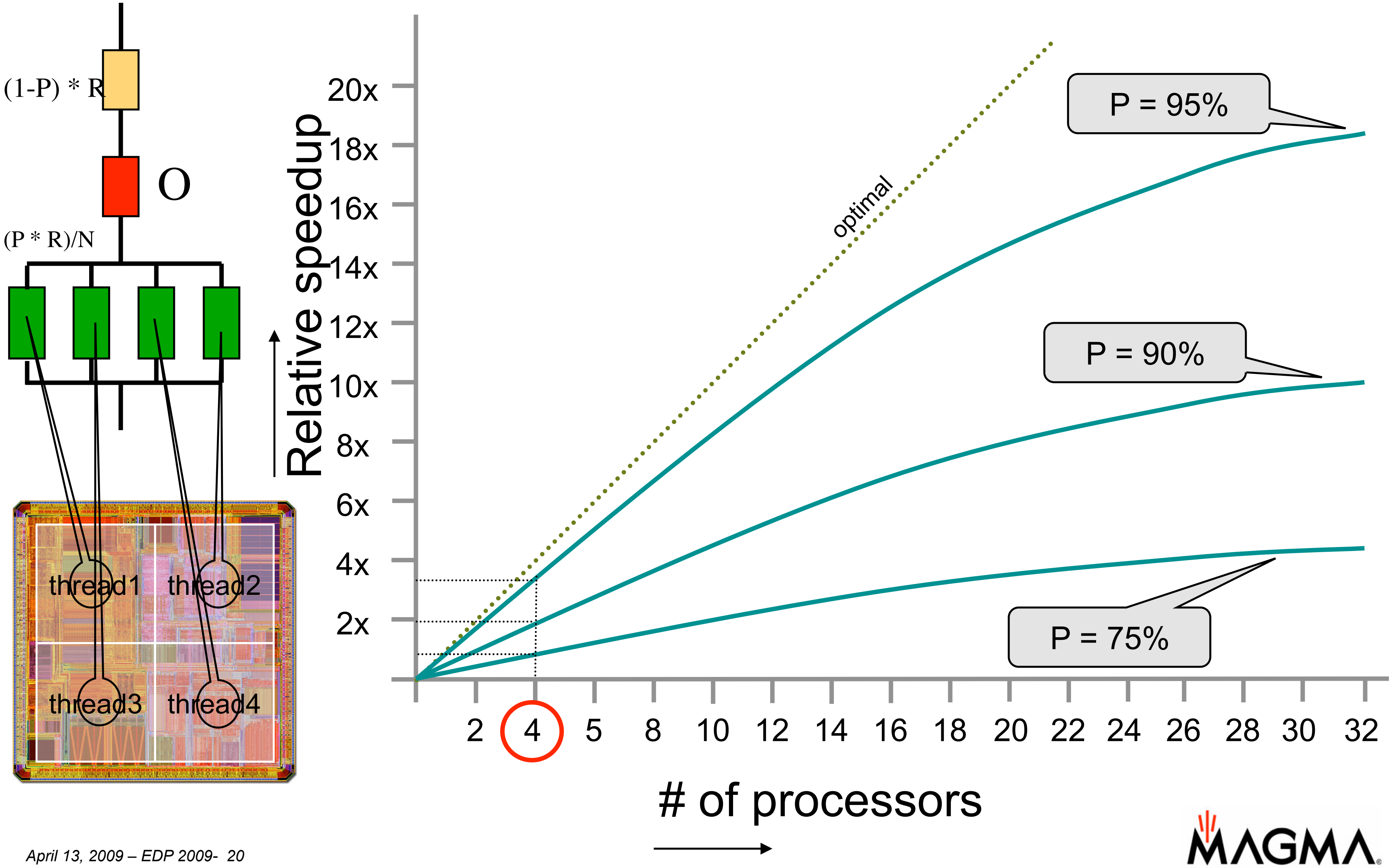
- **Runtime = R**
- **Run time = R/((1-P) + P/N)** + O

P	Maximum speedup	Reality
50%	2x	0.8x
80%	5x	2.0x
90%	10x	2.5x
95%	20x	2.8x

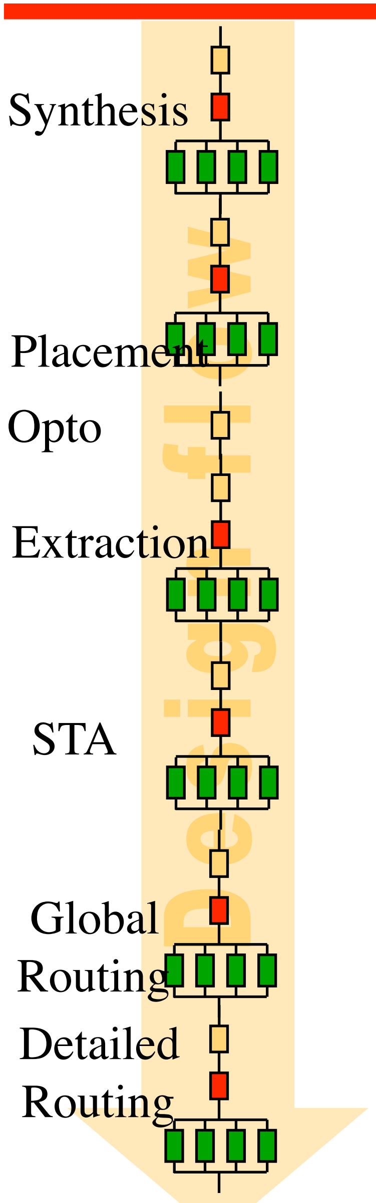
Must push P to 100%

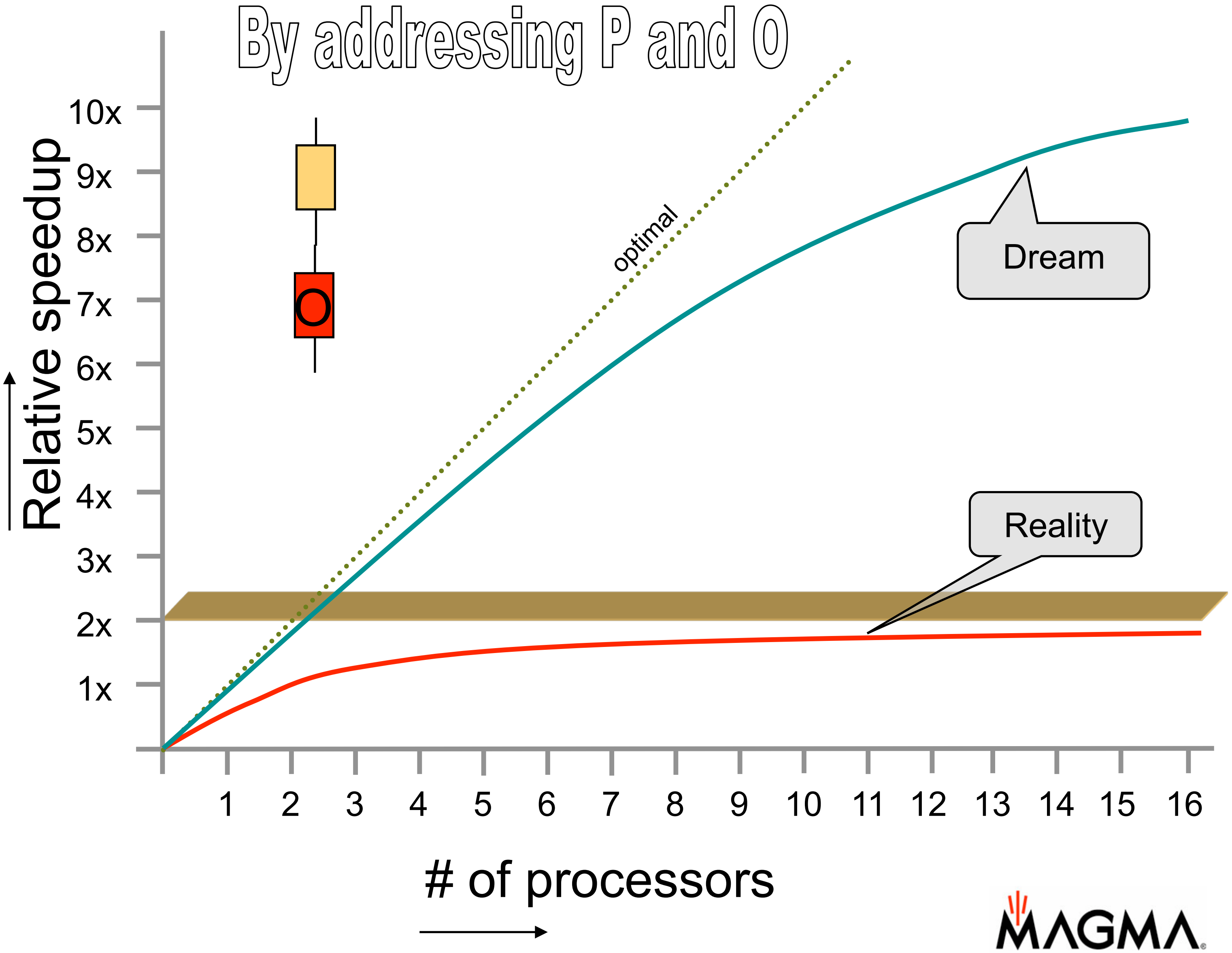
Must minimize O

Parallelizing a single step in the flow



Parallelizing the flow: Can we break the barrier?



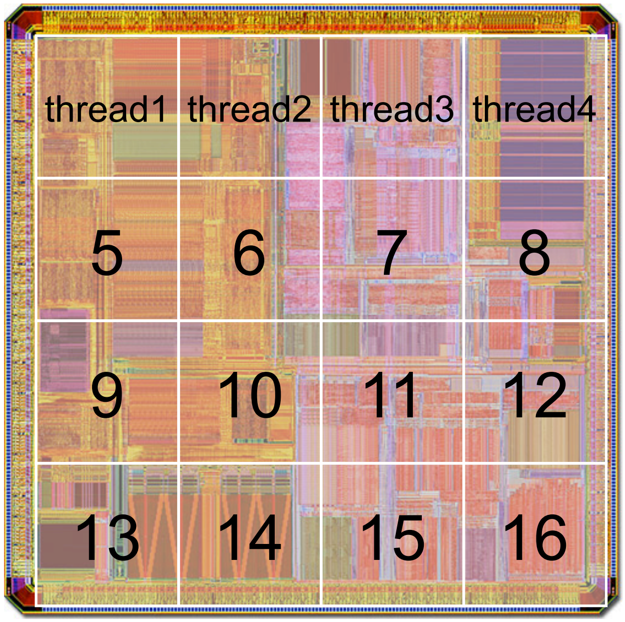


By addressing P and O

Parallelism overhead

- **Successful parallelization requires *very* low overhead**
- **What causes overhead?**
  - 1: Interactions between threads
    - **Dependencies**, locks, **unequal load distribution**
  - 2: Resource bottlenecks
    - I/O bandwidth to memory or disk
  - 3: Partitioning and re-assembly
    - After threads are done
    - Cost of fixing border problems

- **So the key is to define design tasks that are 100% independent**
  - That is why most analysis tools are easier to parallelize.
  - That is why routing is better parallelizable than optimization



Find independent partitions

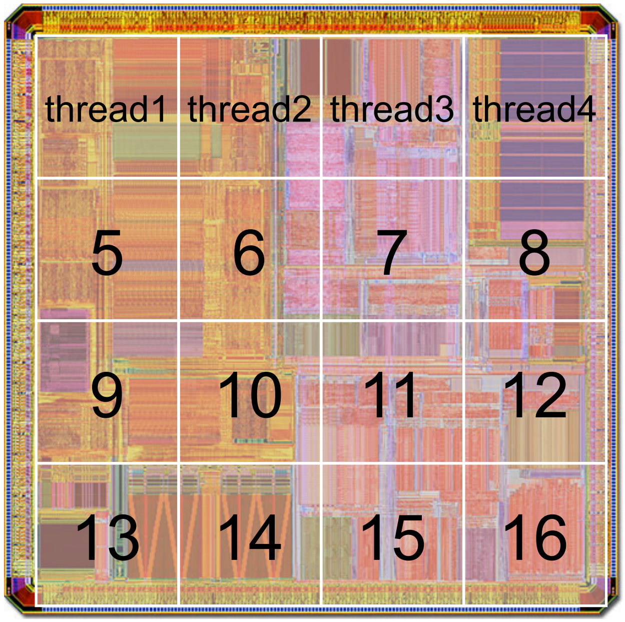
Partitioning is *Evil*

- **Why is it evil?**
  - Overall quality suffers
    - Cannot optimize across boundaries
  - Partitioning problem is not easy.
  - Good partitions take (non -parallelizable) effort!
    - Algorithmic
    - Need to duplicate data

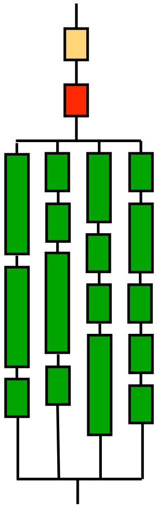
Partitioning:
A necessary evil
for the sake
of parallelism?

Repeatablity: parallelism's silent killer

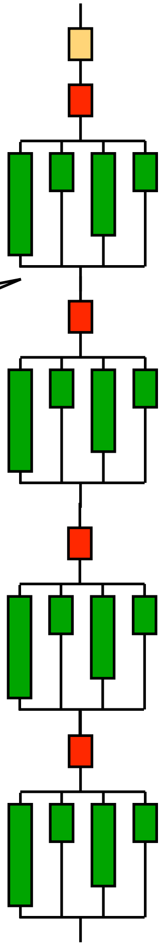
- **4 processors, 16 jobs to do.**



thread1	thread2	thread3	thread4
5	6	7	8
9	10	11	12
13	14	15	16



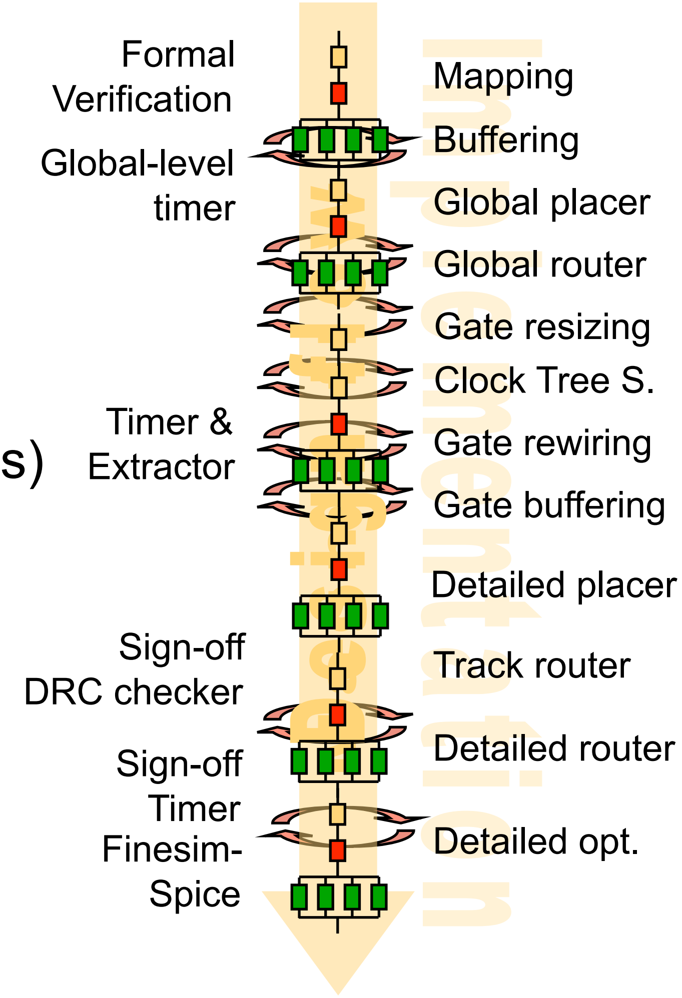
In case jobs are 100% independent



Need to sync

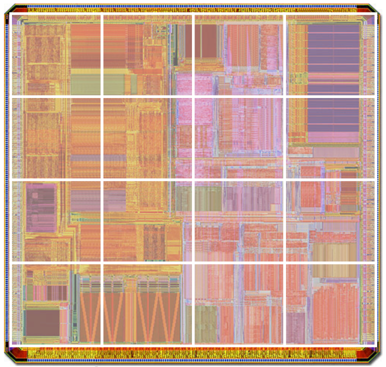
Hitting each parallelization sweetspot

- **Hierarchical internal representation**
- **Each tool needs a different view on this hierarchy:**
  - Logical synthesis (logical hierarchy)
  - Floorplan synthesis (modified hierarchy)
  - Coarse placer (flat with clusters)
  - Voltage island generation (floorplan objects)
  - Timer (tiles at flop boundaries)
  - Parasitic extraction (net + region based)
  - Global router (10 x 10 tiles)
  - Track router tiles (columns)
  - Detailed router tiles (50 x 50)
  - DRC checking (net-based, region based)

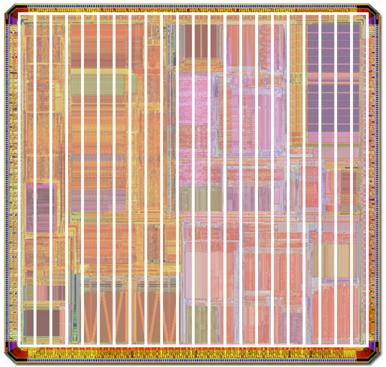


Finding partitions

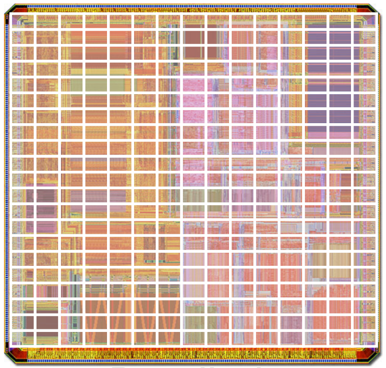
- To hit sweetspot, we need to partition in different ways throughout the flow



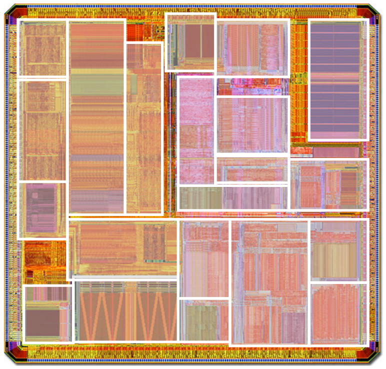
Coarse



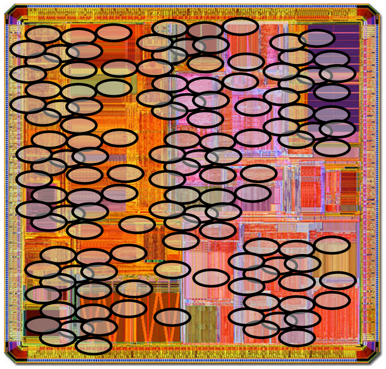
Scanline



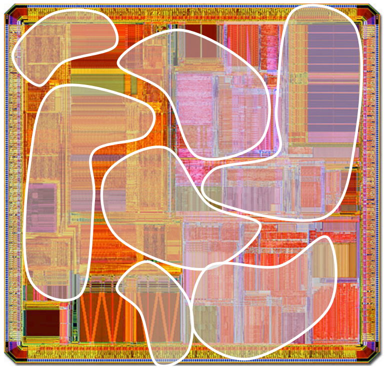
Detailed



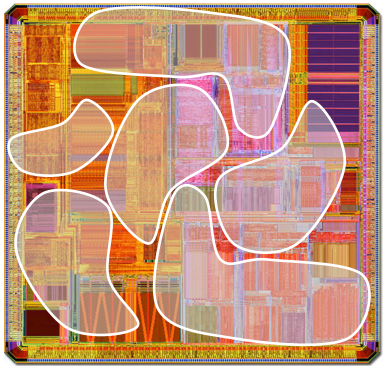
Logical/glassbox



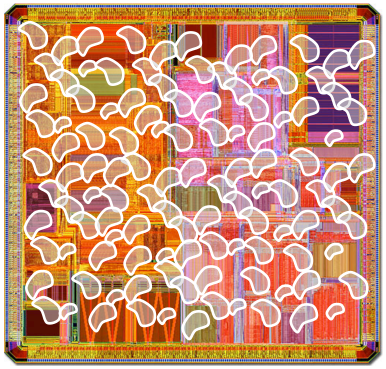
Clustered placement



Logical in flat placement

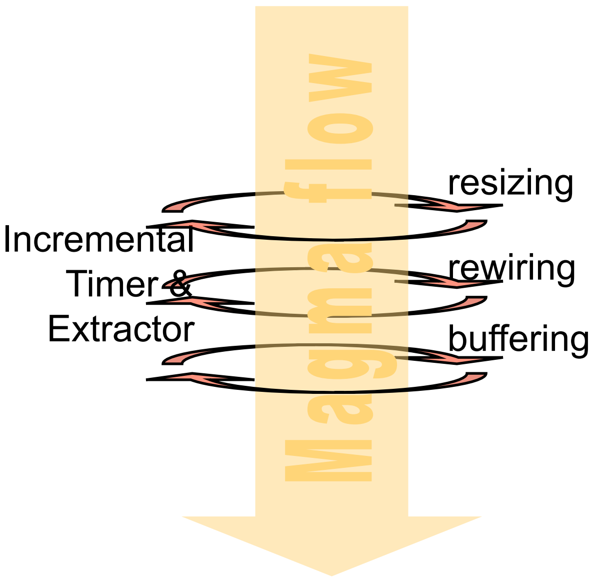


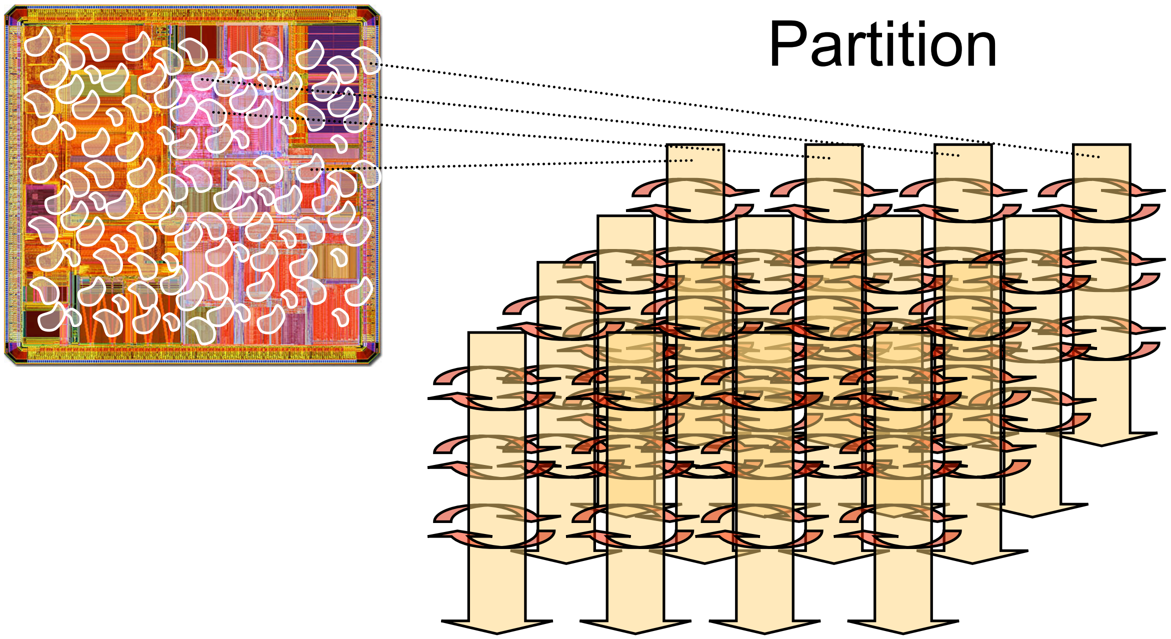
Timer



optimization

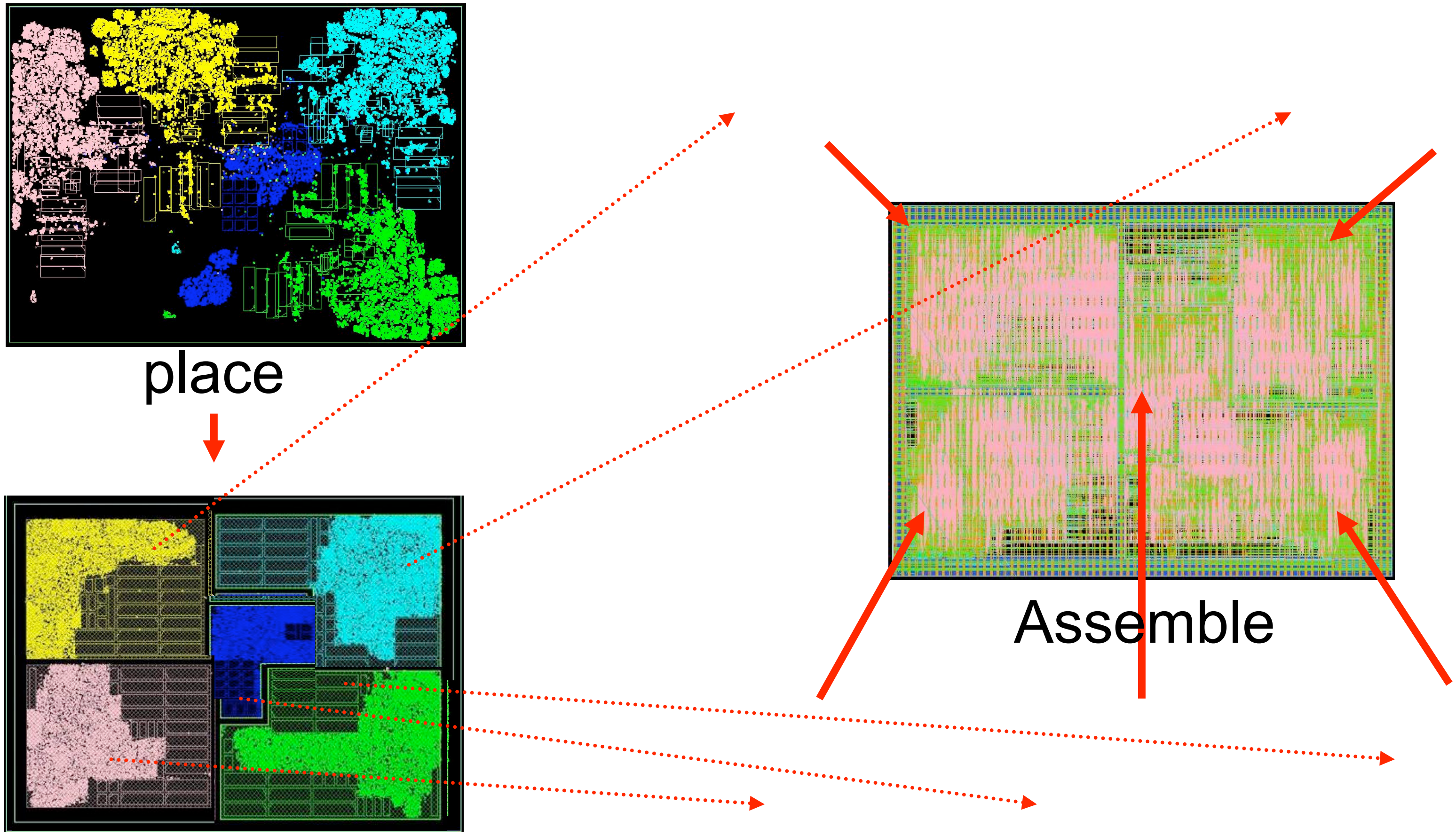
Using fine-grain partitioning in IC synthesis





- Hard to keep partitions independent (logically and physically)

Coarse-grain partitioning: ‘Hydra’



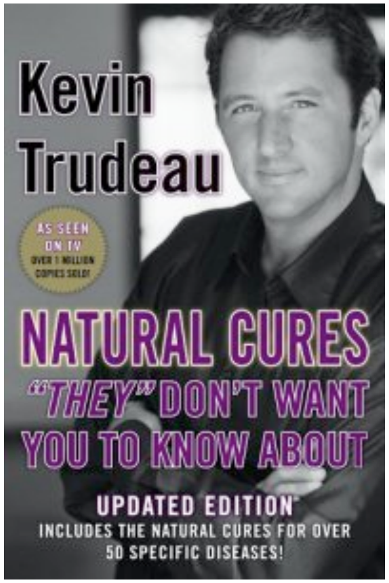
place

Partition/budget

Build each block in parallel

Assemble

Multiprocessing secrets
"they" don't want you to know about

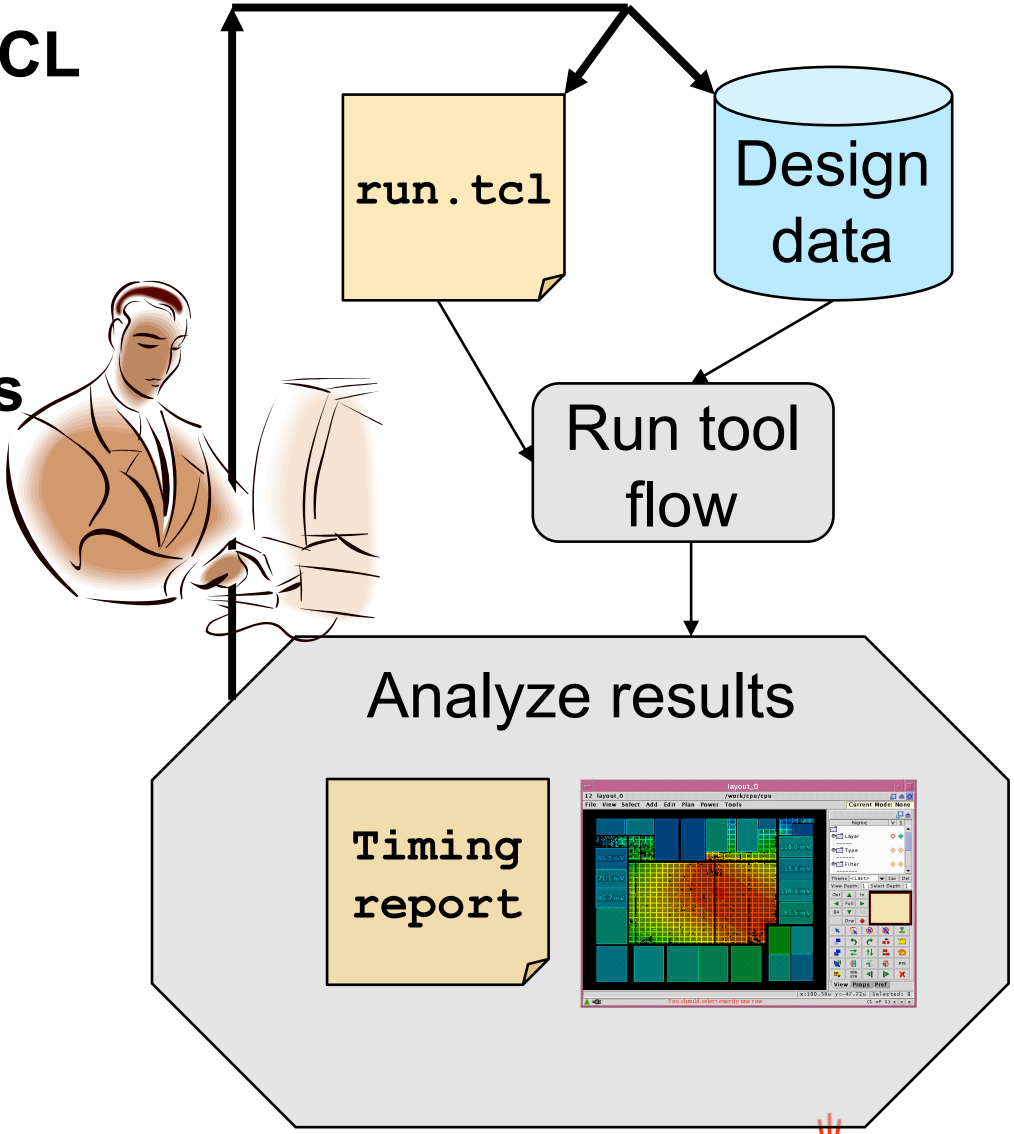

Kevin
Trudeau
AS SEEN ON TV
OVER 1 MILLION COPIES SOLD!
NATURAL CURES
"THEY" DON'T WANT YOU TO KNOW ABOUT
UPDATED EDITION
INCLUDES THE NATURAL CURES FOR OVER 50 SPECIFIC DISEASES!

- Code is hard to write
- Code is hard to debug
- Adds significant partitioning and assembly overhead
- Narrow sweetspot in EDA ***analysis*** tools:
  - DRC, SPICE, perhaps STA
- Synthesis algorithms are tough due to dependencies
  - Repeatability costs efficiency
- Amdahl's law still holds
  - Realistic gain maxes out at 4x
- Using parallelism **costs Quality of Result**

- Parallel EDA startups were spectacular failures
  - Monterey, Athena, Liga

How to really speed up: fewer design cycles

- **Design is tuning of a TCL script**
  - And fixing problems
- **Avoid ‘stupid’ mistakes**
  - Rigorous testing
- **Need Correct-by -construction**
  - By backing off constraints
- **Less C, more A**



run.tcl

Design data

Run tool flow

Analyze results

Timing report

Summary: it’s the flow, not the algorithm!

- **A few EDA *analysis* tools may parallelize OK:**
  - SPICE, DRC
- **Synthesis tool flows parallelize poorly**
  - Nature of the algorithms and flows, data size
  - Customers not willing to pay quality hit
- **Overall flow speedup saturates at 3x-4x**

- **We’ll figure it out somehow**
  - Parallelism is only a part of the solution…