Recommended Rules NOT Recommended

Lou Scheffer

Origin of recommended rules

- Originally, design rules were the contract between designer and fab
  - Follow the rules and the chip can be built
  - Does not cover parametric yield
- But modern rules are not really binary and are set aggressively
- A small chip might have OK yield with everything set to minimum
- A large chip may have poor yield
- Mathematically, the true min spacing (for example) is a function of chip size and desired yield

Origins

- But rules that vary with chip size have practical problems
  - Chip size may not be known (IP for example)
  - No infrastructure for this (DRC)
- Hence ‘recommended rules’
  - If you follow this rule your yield will be better
  - How much better, rule does not say

Recommended rules and optimizations

- Typical recommended rules
  - Double vias where possible
  - Add extra spacing where possible
  - Add more extension around vias where possible
- Unfortunately, these do not work well at all
- If you have enough room so all recommendations can be followed might be OK.
  - But this is seldom the case

Choice of recommended rules

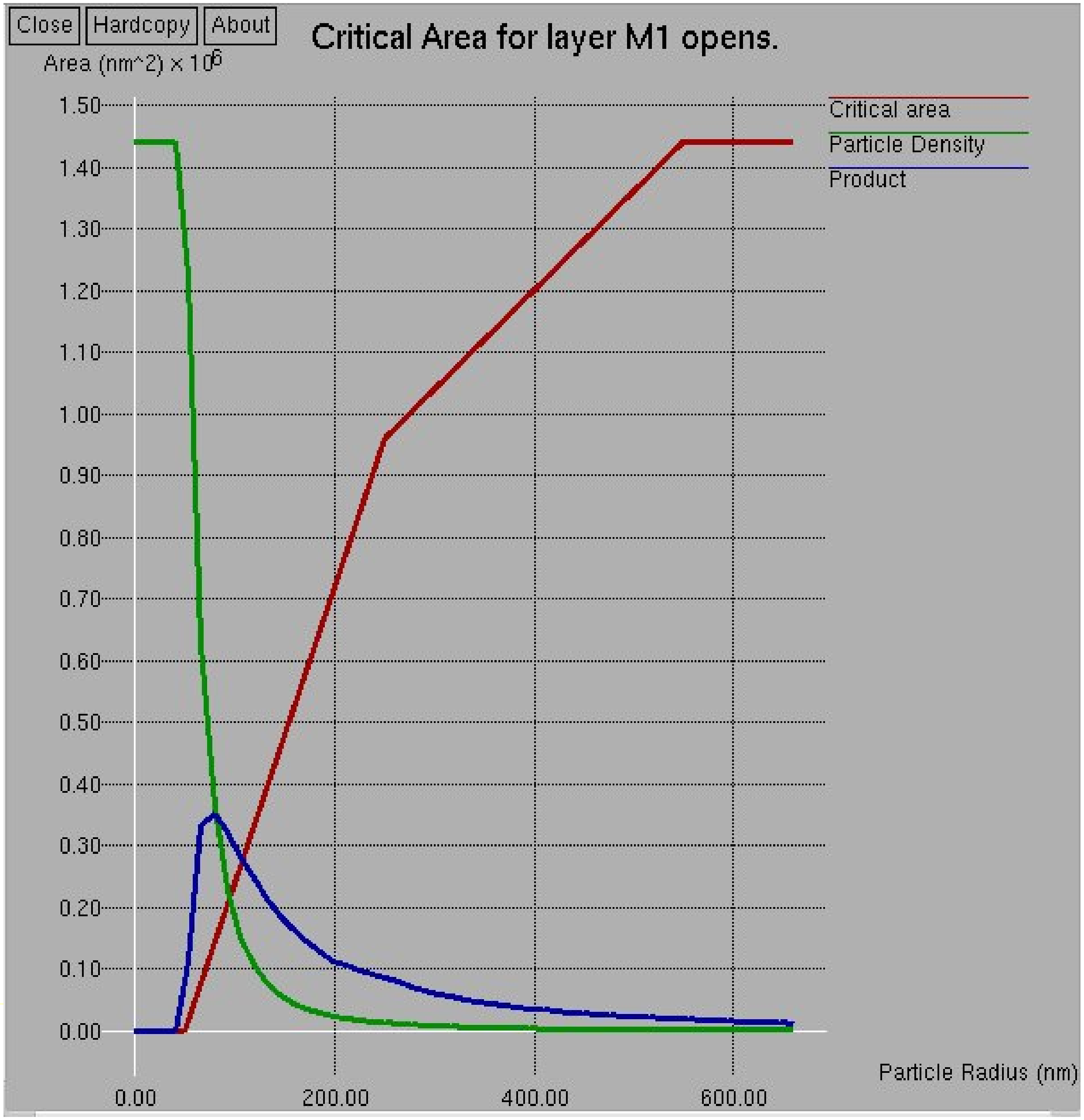
- Historically, smallest chips are best, so no designer will make their chip bigger just to follow recommended rules
- Therefore designer must normally choose which recommended rules to follow.
- But the rules give no guidance for this

A simple experiment

- In some cases the real data exists
  - Quantitative data on single vs double via
  - Defect density for opens/shorts of different wire spacings and widths
- Using this data, we can see how well recommended rules work

More accurate approach to min spacing

- The “correct” answer is to compute critical area curves, mpy by particle density, integrate over all particle sizes

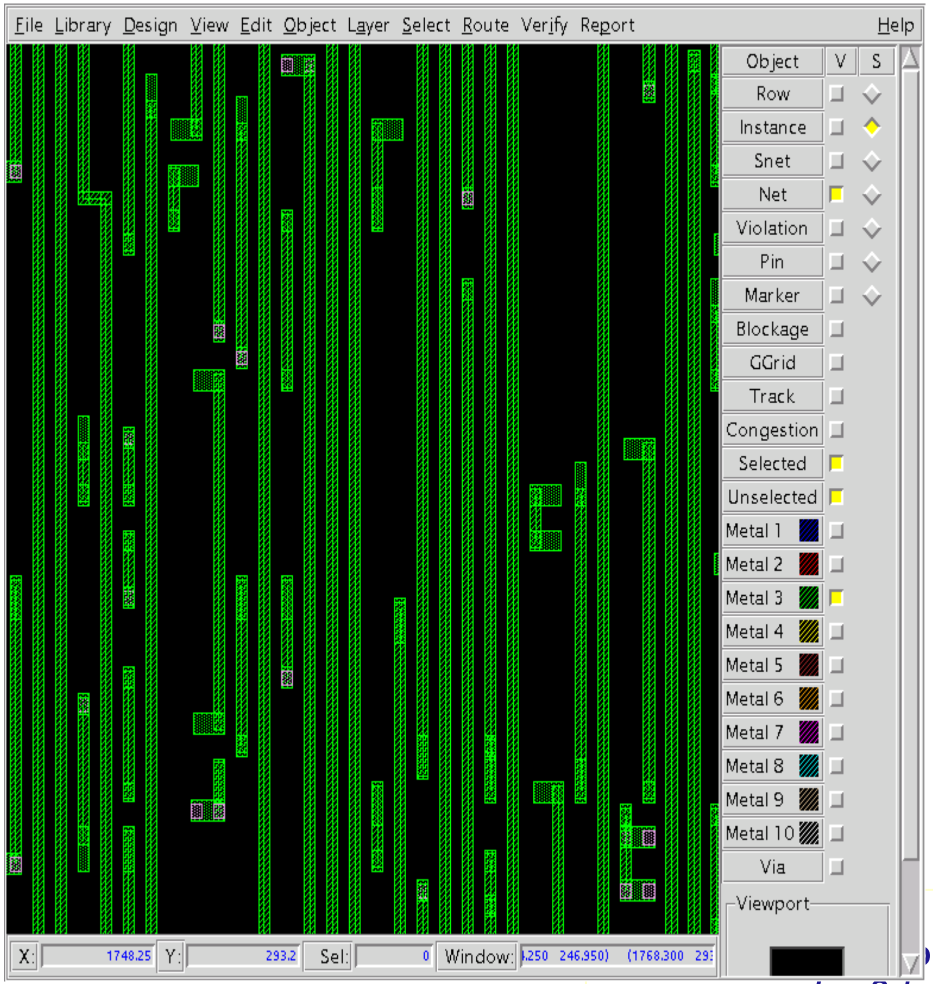


Practical example – cramming double vias

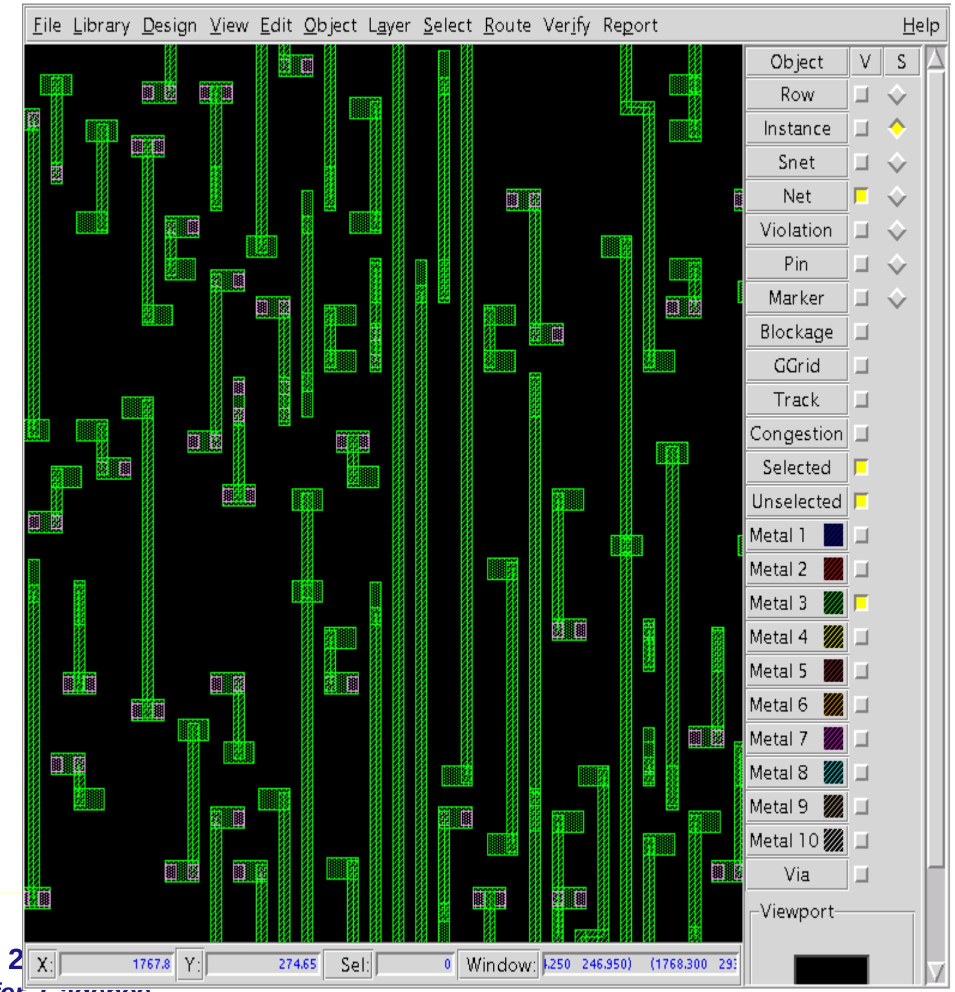
- Hypothesis – "double via" rule is more important than the "avoid min spacing" rule
- Experiment – use a router with different settings for inserting double vias
  - + Do where it's easy, post process to double vias
  - ++ Move lines to double vias, topology unchanged
  - +++ Do during routing, can change topology
- Even if you only consider vias and particles, best results require tradeoffs

Routing Yield Optimization

- Concurrent: Swap double-cut vias for single-cut vias while routing
  - dynamically check local congestion
  - change routing topology to allow more swapping
  - Effects on timing, SI and yield can be handled concurrently and during later ECO steps

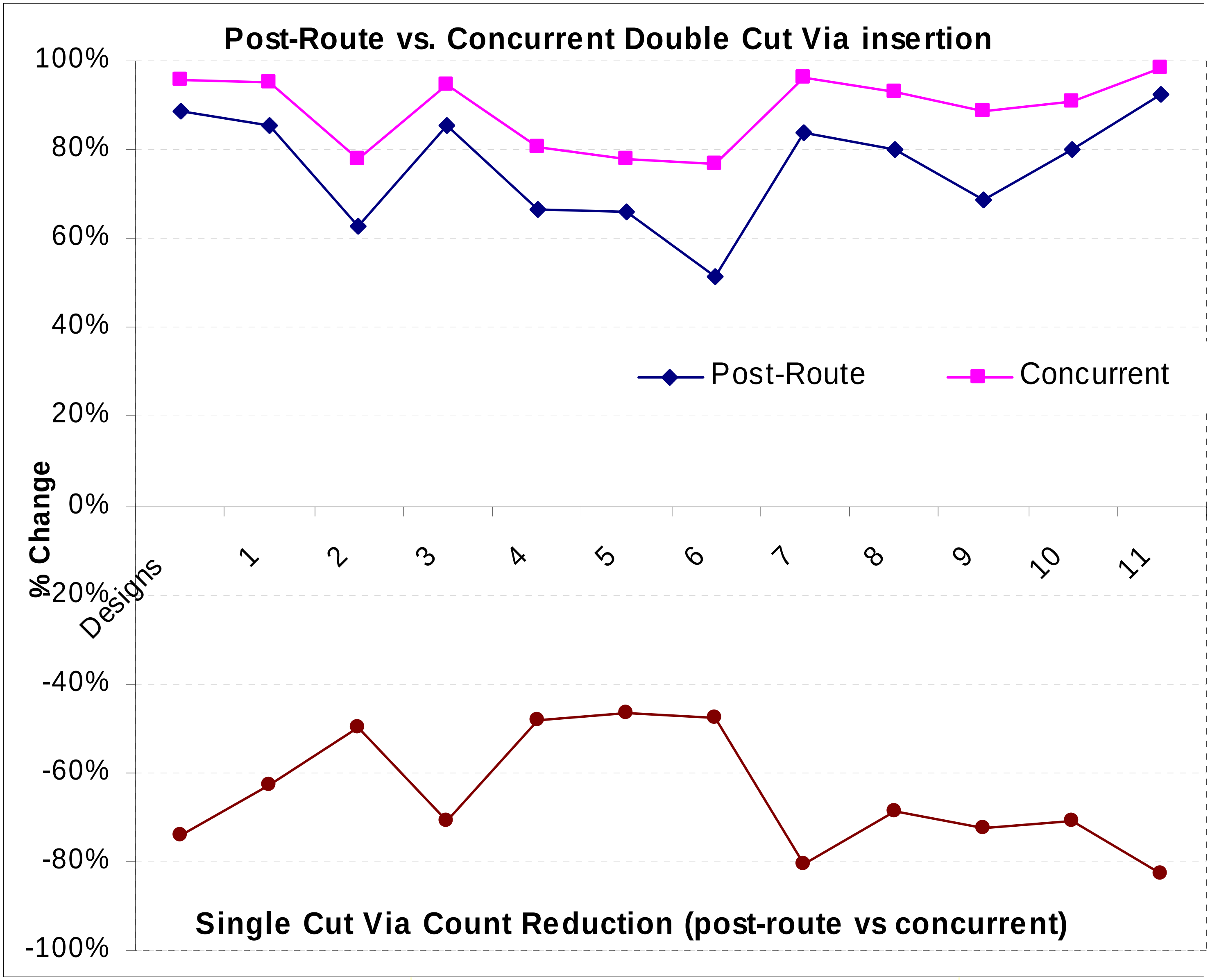


X: 1748.25 Y: 293.2 Sel: 0 Window: .250 246.950) (1768.300 29:



X: 1767.8 Y: 274.65 Sel: 0 Window: .250 246.950) (1768.300 29:

More aggressive router setting, more double vias



But max doubling does not optimize yield!

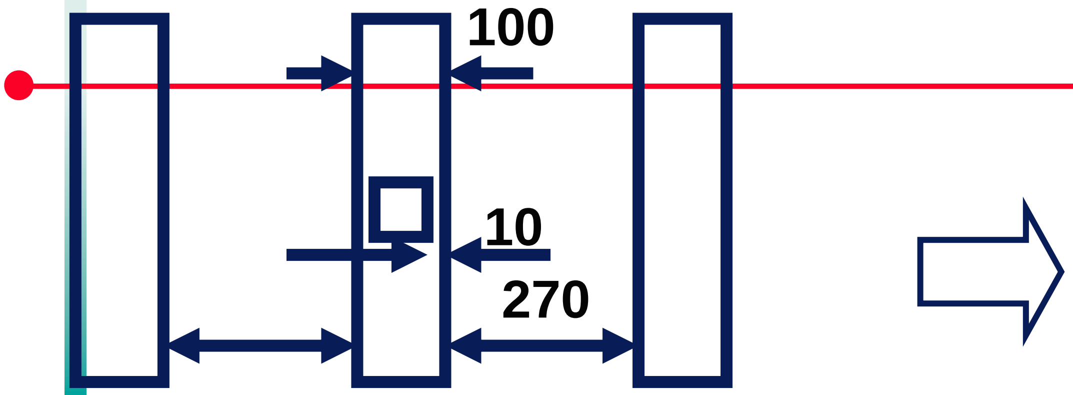
Design Name	#Layers	#Nets	No Routing Opt	Wire-Spread + Via Reduction + Post-Route Via Opt	Wire-Spread + Via Reduction + Aggressive Concurrent Via Opt	Yield % Change
B	90nm / 7	235k	**92.51**	92.99	**93.17**	+0.67
S	130nm/ 7	363k	**83.81**	**84.72**	84.13	+0.91
T	90nm / 8	552k	**71.46**	**73.45**	72.87	+1.99
N	90nm / 7	1.1M	**81.89**	82.43	**82.56**	+0.67
A	90nm / 8	1.21M	**82.13**	84.73	**84.89**	+2.76

Next, litho data may be needed

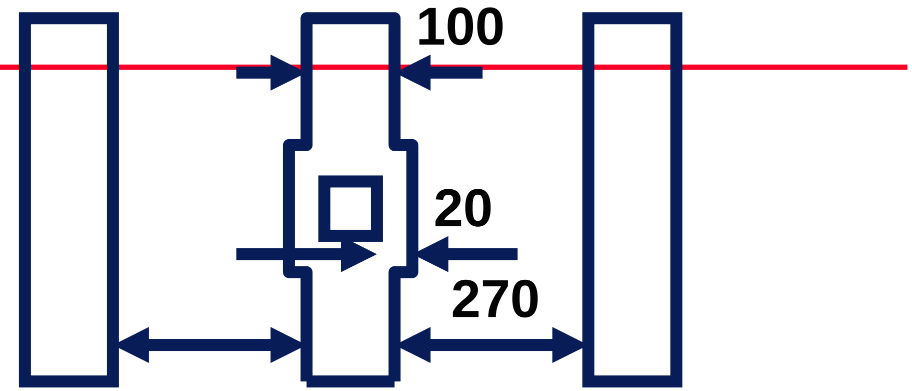
- Router must be able to trade off
  - Particle yield
  - Litho yield
  - Via yield
- All must be expressed in comparable units
- Note that litho yield is very hard to express as recommended rules

Next slides shows why this is needed

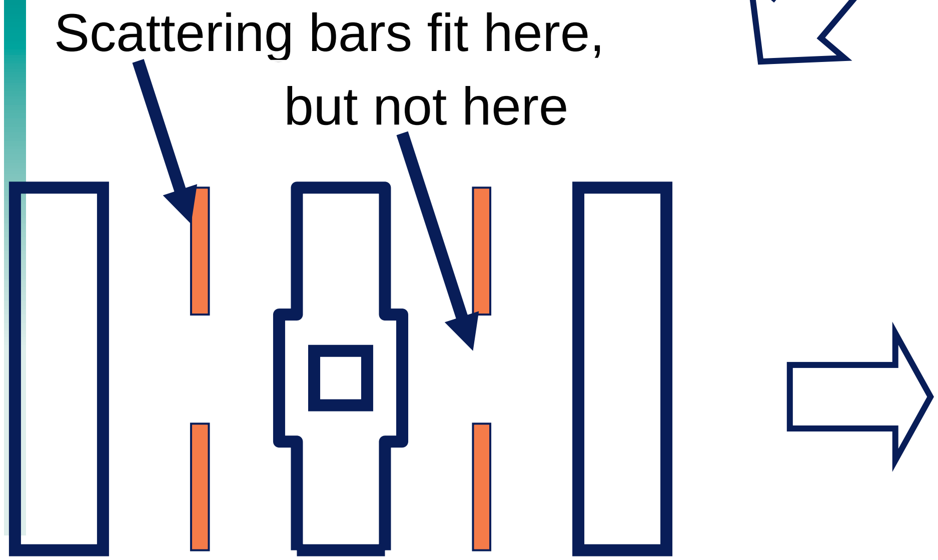
Example of tradeoffs in a simple case



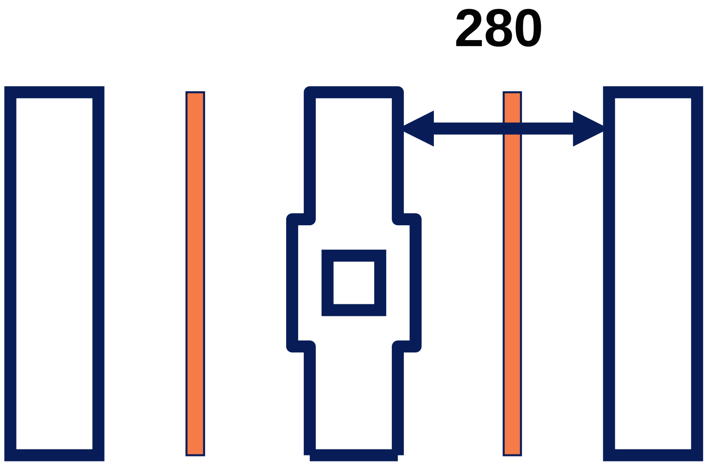
(a) Original layout. Dimensions in nm



(b) Attempted improvement



(c) Why ‘improvement’ has a worse process window



(c) Is this better? Depends on the numbers – via vs. lithography vs. increased size

Few recommended rules for width

- Exact models for point defects vary
  - 1/x^3, exponential, binomial
- But all models fall off very fast with radius, so
- Bigger width/spacing is always better
- Traditionally, shorts a bigger problem than opens
- And, extra spacing helps timing
- Hence, traditional wire spreading spreads wires, does not increase width, needs no hard data

But they will come, and they will conflict

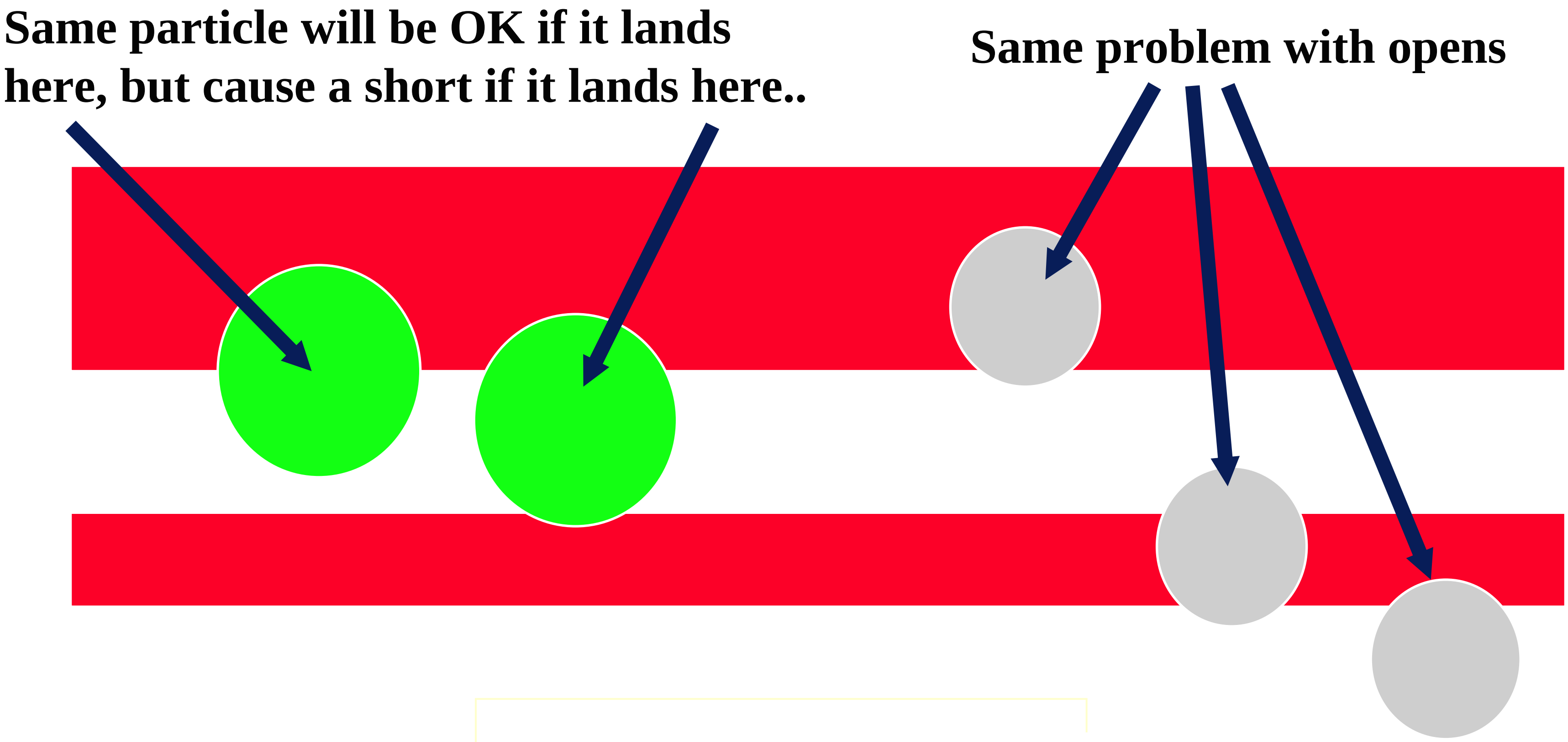
- With copper process, opens also becoming important
  - Use extra space for wire widening, too
  - Double vias want space too
  - And extra enclosure is recommended
  - And forbidden pitches might need this space
- Now 3-4 different recommended rules could apply to every spot in a routed design!

So what should be done?

- Replace recommended rules on width/spacing with defect density data
- Replace recommended rules on via doubling with a via manufacturability model
- Add data on lithography, which is hard to do with recommended rules

Point defects (usually particle contamination)

- Based on the idea of ‘critical area’
- Find odds of a short/open based on particle size distribution and geometry

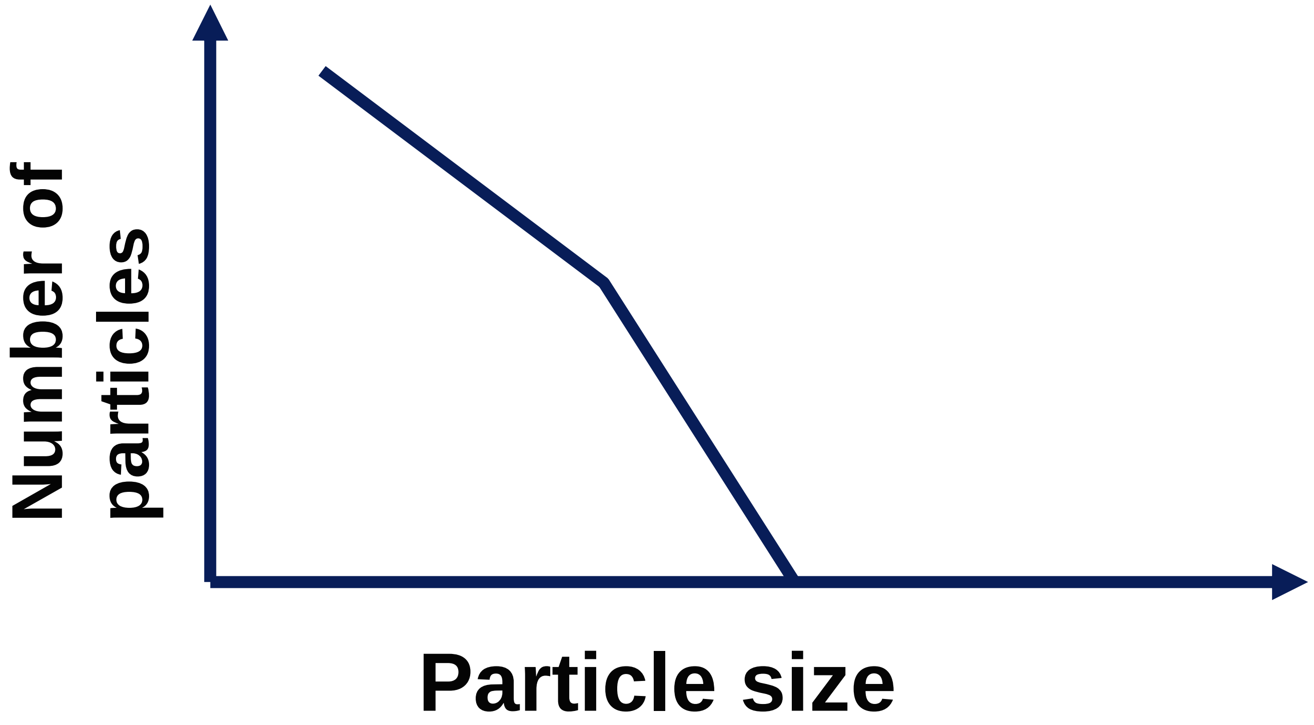


Same particle will be OK if it lands here, but cause a short if it lands here..

Same problem with opens

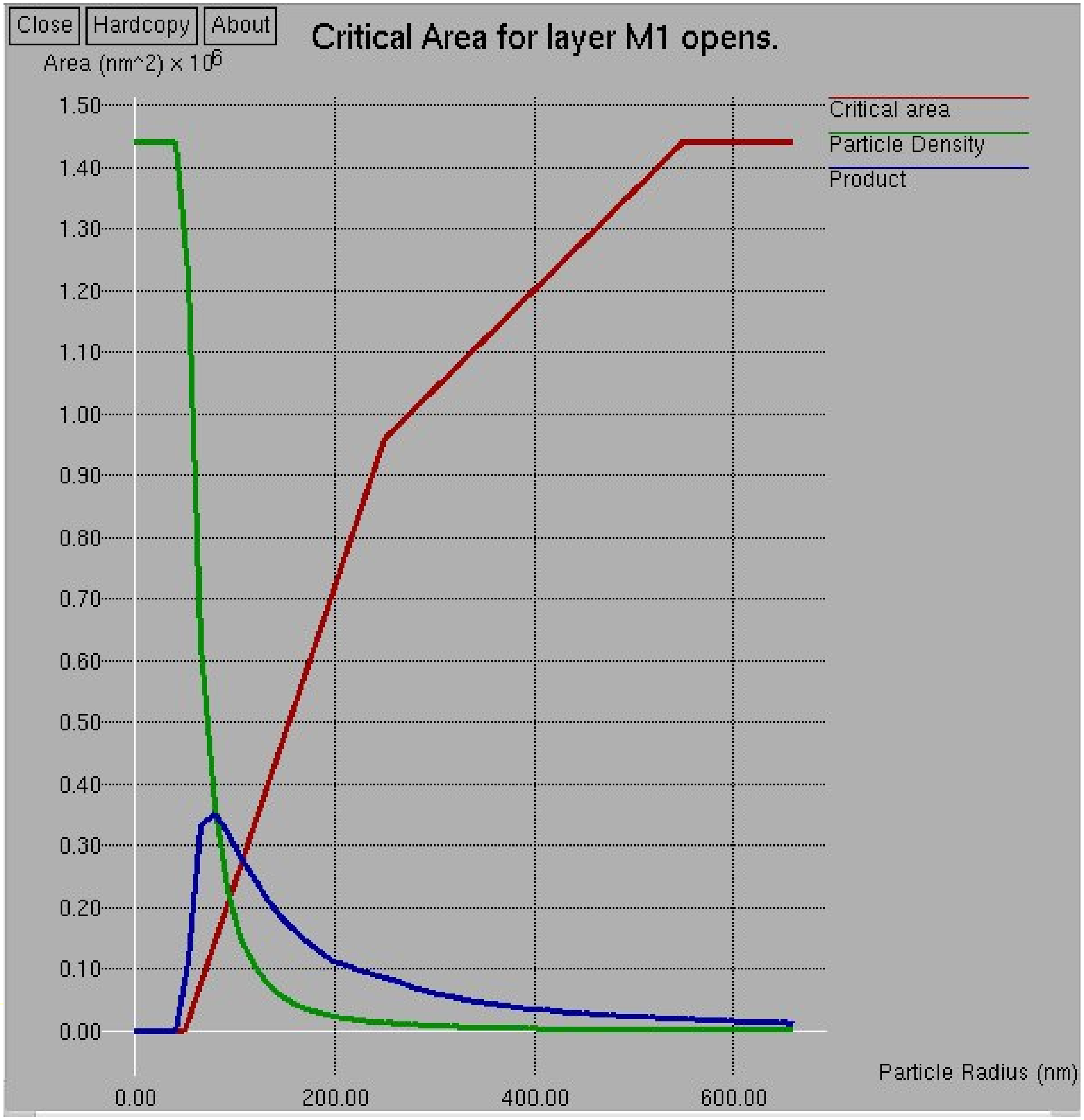
So we tell the router the particle density

- For each layer, and for opens and shorts independently.
  - Piecewise linear on a log scale



Then do critical area calculation

- The “correct” answer is to compute critical area curves, mpy by particle density, integrate over all particle sizes

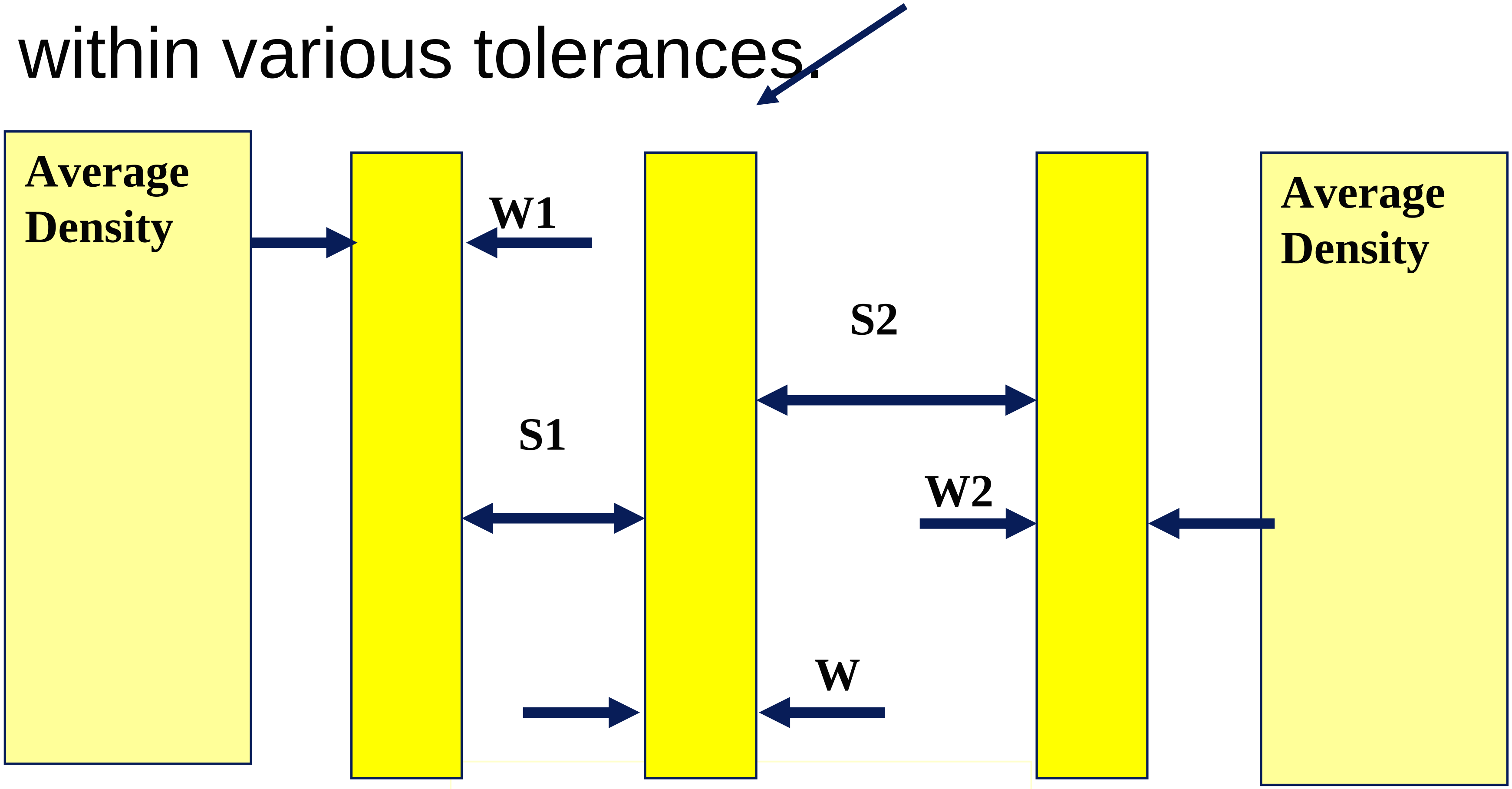


How we might model vias

- First, assume a 3 (or more) cut via is so much better than a 1 or 2 cut via that extra cuts don’t matter
  - All vias classified as 1-cut, 2-cut, many (>= 3) cut.
- A common customer model
  - <u>1 cut vias</u> have a cost that depends on overlap
  - <u>2 cut vias</u> have a smaller and constant cost
  - <u>3 or more cut vias</u> have no cost
- Extension – cost depends on distance to nearest other via (PDF is advocating this)

Representation of process window for router

- Assume long parallel lines.
- Read a table f(w1,s1,w,s2,w2). This has the litho windows under which the wire width is within various tolerances.



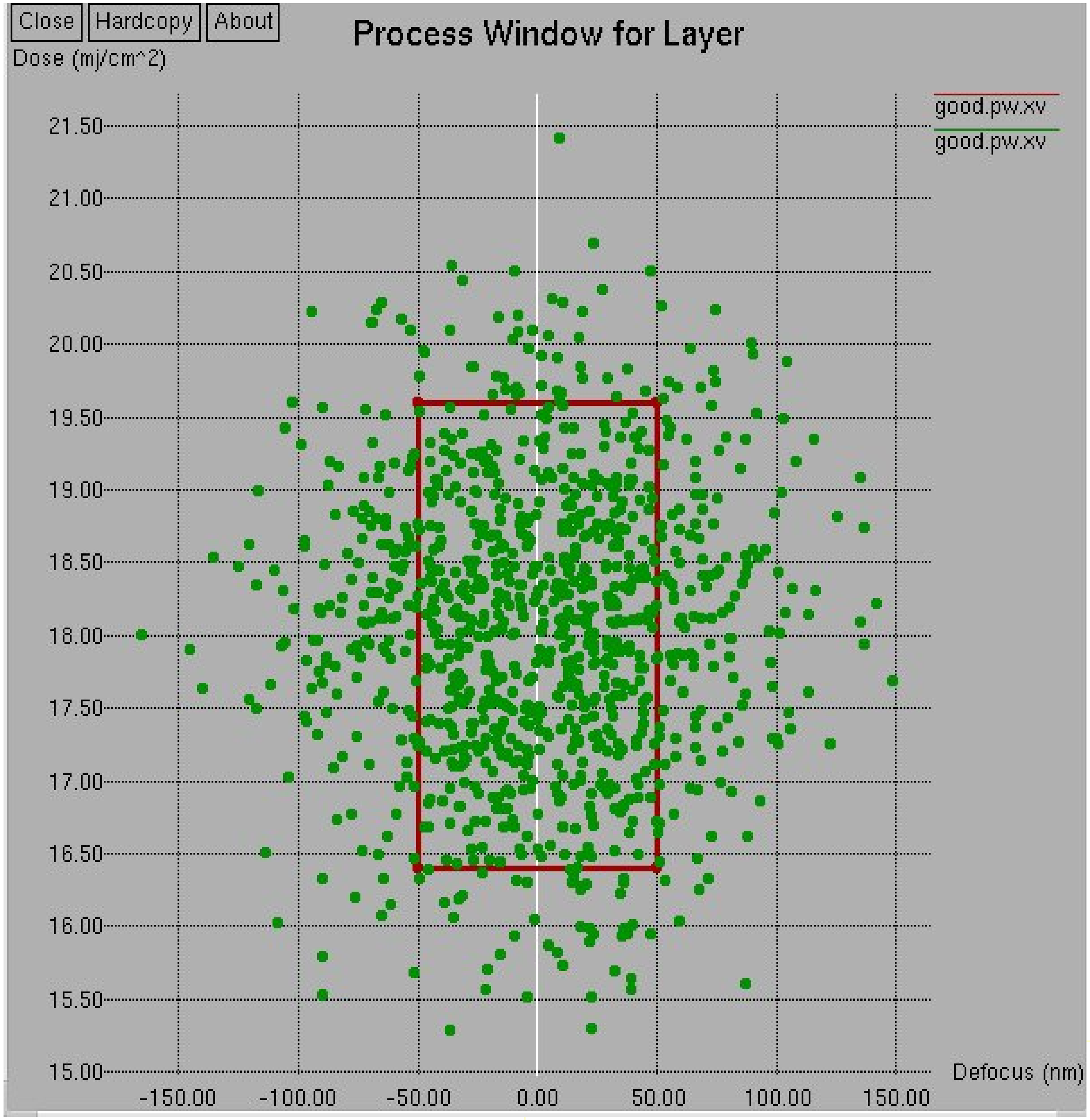
How can we derive/store the litho data?

- Build representative widths/spacings. For designs made by routers, typically about 2000 cases are needed, each very small.
- For each case, do OPC as you will in production.
- Now, for a variety of tolerances,
  - Simulate the lithography and find process window
- The process window becomes a table entry.
  - Examine design, compute process window for a layer
  - Use Monte Carlo over variables to compute failure probability

Big problem - reporting

- The users of a design tool are ***NOT*** process experts
- One advantage of recommended rules is that they are already in their language
  - Either a rule is met, or it is not
  - Error markers are integrated into layout editors
- If we replace by a calculation we must explain
  - Width, spacing, multiple vias, and enclosures are more or less intuitive
  - Objective: “good chips per wafer”, not min size
  - But litho is not intuitive

Lithography issues



- We compute process windows internally
- But telling designers what to fix, and how to fix it, is hard
- Gridded routers are easier here

Lou Scheffer, Cadence

Reporting lithography

- This is especially hard to explain since there is **NO MENTION** in the design rules
- We’ve looked in detail at the rules from 2 large 65nm manufacturers
- Just minimum spacings and widths
- Litho considerations are not even in the “recommended” rules!
- Designer says “show me why this is bad”, and we cannot point to anything in the rules

Conclusions

- Recommended rules work poorly
  - All cost area, conflict with each other, no way to choose
- Situation will get worse
  - Opens will create more conflicting rules
  - Litho is not easily expressible by recommended rules
- Need real numerical data instead
  - Particle densities, via fail rates, litho info
- Need to improve reporting to match

The End