# Restrictive Design Rules and Their Impact on 22 nm Design and Physical Verification

David Abercrombie, Mentor Graphics Corporation
david_abercrombie@mentor.com / (503) 685-1446

Praveen Elakkumanan, IBM
pelakkum@us.ibm.com / (845) 892-0735

Lars Liebmann, IBM
lliebman@us.ibm.com / (845) 892-2254

## Abstract

Process variability is posing considerable challenge to the capability of lithography and manufacturing techniques, and thus impacts both performance and yield of advanced node chips. To ensure the manufacturability and performance of chips at 22 nm, one approach the industry is considering is restrictive design—limiting the type and placement of features used in designs. Gridding of critical layers significantly reduces the total physical design space available and makes restrictive design possible. This paper will examine the basics of gridding, the requirements for restrictive gridded design, and the automated methods for accurate checking of Restrictive Design Rules (RDRs). Resolving the debug challenges associated with the implementation of checking restrictive design and grid rules will also be discussed.

## Keywords

Restrictive design, RDR, 22nm

## Authors' Biographies

David Abercrombie is the Advanced Physical Verification Methodology Program Manager at Mentor Graphics in Wilsonville, Oregon. He has 15 years of experience driving design for manufacturing and yield enhancement programs in semiconductor manufacturing. David received his BSEE from Clemson University, and his MSEE from North Carolina State University.

Praveen Elakkumanan received his B.E (Honors) degree in Electrical and Electronics Engineering from Birla Institute of Technology & Science (BITS), Pilani, India, and his M.S and Ph.D degrees in Computer Science and Engineering from University at Buffalo (SUNY). His research interests are on process variability, SRAM design, design for manufacturability, soft errors and interconnect issues in high performance and low power VLSI systems. He received the best poster award at the IBM Austin Conference on Energy Efficient Design (ACEED), 2005. He is a member of the IEEE, ACM, Sigma Xi Honors Society and INFORMS, and serves in the technical program committee of ISQED, ICCAD, VLSI Design. He has given tutorials and invited talks at ISCAS, ICCAD, DAC, VLSI Design, SUNY, and RIT, and has authored several papers and patents. He worked as a software engineer for two years, and as a supplemental employee at IBM Austin Research Lab. He is currently with IBM Semiconductor Research & Development Center, working on advanced DFM & physical design methodologies for future technologies.

Lars Liebmann, Distinguished Engineer, Design for Manufacturing, Semiconductor Research and Development Center, joined IBM in 1991 after receiving B.S. and M.S. degrees in Nuclear Engineering and a Ph.D. in Engineering Physics from Rensselaer Polytechnic Institute. Dr. Liebmann came into Design for Manufacturing with a background in advanced lithography. His work on layout-intensive resolution enhancement technology, such as the design of sub-resolution assist features and the automatic generation of alternating phase shifted mask layouts, led to his pioneering work on restricted design rules (RDR) as a practical means of implementing DFM. Dr. Liebmann is currently charged with the strategic integration and execution of all aspects of DFM in IBM's Semiconductor Research and Development Center. Dr Liebmann holds over 40 patents and has published over 40 papers. He was a founding program committee member of SPIE's "Design and Process Integration for Microelectronic Manufacturing" conference, which he chaired in 2004 and 2005,

## Introduction

As every design and manufacturing engineer knows, the lower k1 falls, the more difficult it becomes to accurately print the desired shapes on the wafer. At the start of 45 nm, traditional lithography techniques were not sufficient to print difficult 2D structures. Immersion lithography improved the k1 factor and restored stability all the way though 32 nm. However, at 22 nm, it may become very difficult to print shapes accurately even with immersion lithography, and without immersion, impossible. Due to the absence of lithography tool improvements, printing any random 2D shape at 22 nm with the current sources of light and lithography techniques may both be infeasible and unnecessary for design.

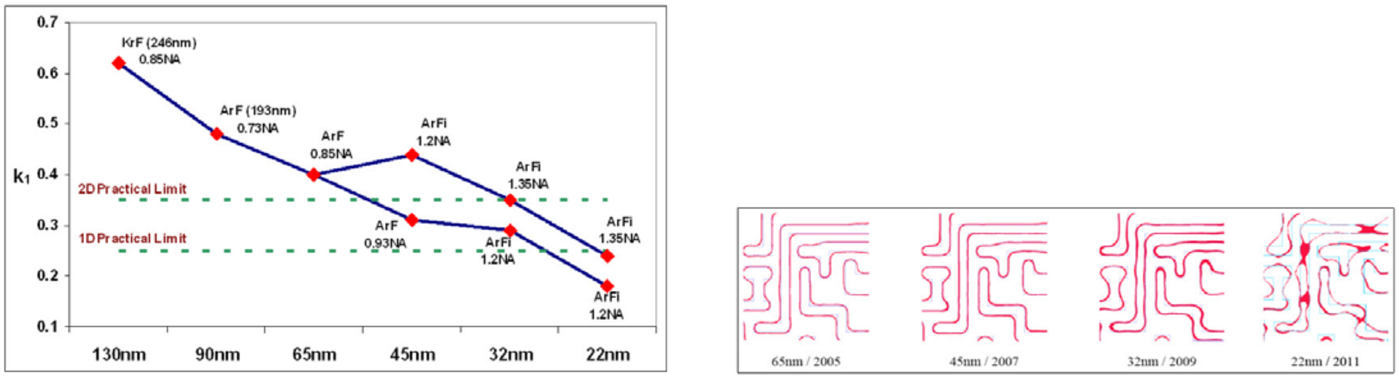


*Figure 1: The limits of lithography techniques and the changes in printed results across process nodes*

This ever-increasing challenge of attempting to print an accurate representation of what the designer is trying to implement has led, over time, to an increasing number of design rules and operations. The number of rules has increased in an attempt to define and limit the situations in which it gets very difficult to print accurately. Also, the number of operations that must be performed to implement the rules in a rule deck are increasing faster than the rule count itself, suggesting that even existing rules are becoming more and more complex to determine and apply. A fundamentally different approach could potentially provide more optimized technology and design solutions.

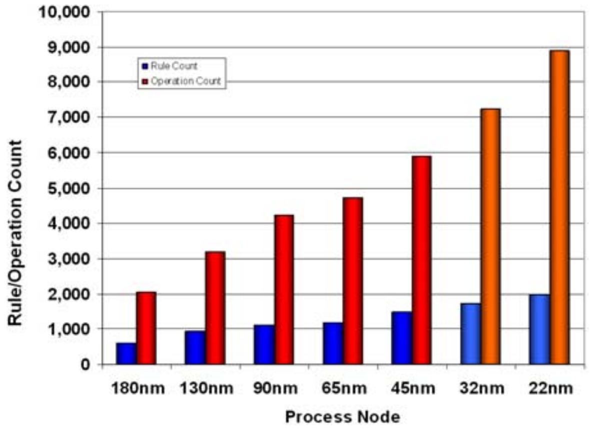


*Figure 2: Both design rule count and rule operations are growing, reflecting the complexity of smaller designs*

Rather than trying to define everything the designer cannot do, one rendering of design rules could attempt to define what the designer can do (prescriptive design rules), and make the lithography problem simpler to solve. A much less radical approach is the use of restricted design rules (RDRs), which may lead to much more regular layout, generally manifested as pitch and grid restrictions. Physical verification will need to evolve to better manage checking of restricted design constructs. In this paper, we describe in detail the implementation of practical restrictions in design and explore not only methods for accurate checking of pitch and grid, but also means for resolving the debug challenges associated with restricted design implementation.

## *Restricted Design*

Today's designs allow maximum flexibility to the designer. Designs need not be very regular—designers are free to use a wide assortment of configurations and shapes and routing types, and to be very creative in the actual design layout, as long as they meet some very simple, basic design rules. However, at 22 nm, increasing variability in the performance of the device necessitates us to reduce the available physical design space. For example, lithographic rounding of both the active and the contact in a source or drain connection can reduce the alignment marginality, creating the potential for a resistive contact. In a gate construct, horizontal bends in the field poly near the gate can induce an inherent systematic variation in the L-effective on the corner of the gate. With misalignment, that variation can be quite dramatic, causing the device in this transistor to have more variation in terms of drive current, leakage, etc. Similarly, because of a horizontal-to-vertical transition in the active layer, this curvature can cause variation in the W-effective, affecting the drive strength, among other aspects. With alignment variability, this effect will vary dramatically as well. These variations in layout, which affect the printability (e.g., inability to print accurate rectangles), create the yield and performance problems in the design.

The basic concept behind restricted layout is to limit what the designer is allowed to do. Designers want a wide assortment of possible features or shapes or constructs that they are allowed to use. Conversely, the fab would like a very limited set of what the designer is allowed to use—ideally, everything would be perfectly regular and repeatable, making designs much more simple to process and much more robust against manufacturing variability. The restricted design compromise is to have some assortment of allowable features or constructs, but a much smaller list and a more controlled list than what has been allowed in the past. With restrictive design, future designs will be much more regular than current designs, and therefore actually manufacturable, but will still provide the same robustness that we have today.

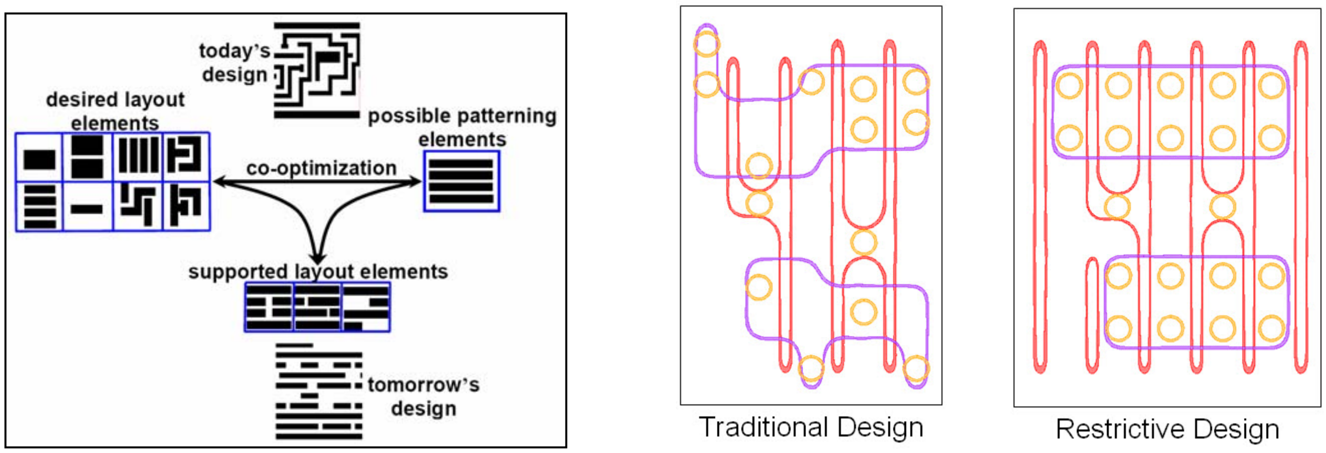


*Figure 3: Restrictive design will combine selected design elements to produce more regular and controlled layouts*

In the traditional design shown in Figure 3, there is regularity of a sort, although there is a wide assortment of features and constructs being used. The gates are on a regular pitch and are, for the most, vertical, although there are horizontal bends within them. There is a fair amount of variability in the metal routing and the active layout.

If the same design is laid out in a more regular style, the performance variability can be decreased considerably. In the restrictive design, there is much more regularity in the design shapes. Ideally, the gates would not only be vertical on a given pitch, but there would be no “wrong way” (horizontal) routing in the polysilicon whatsoever. All the active areas need to be perfect rectangles, and all the metal lines need to be unidirectional going only horizontally or vertically (no jogs and no bends). However, implementing all these restrictions may significantly constrain the design options and may impact density. For an effective technology enablement, there should be a middle ground where designability and design options are not very restricted, but the physical design space is more regular. This is attained through careful gridding of the layers, as described in the next section.

## ***Basics of Gridding***

A layout is gridded when the vertices of all drawn shapes within a given area are restricted in placement to coarse grid points. All the shapes in a gridded layout are drawn as in a conventional design, using rectangles, paths and polygons. Although the design is still hierarchical, all the design shapes are checked in two stages. In the first stage, each shape is checked for size legality (e.g., widths) and placement (gridding). In the second stage, additional rules that apply between geometries (spacing, overlap, etc.) are checked. Checking in the second stage is invalid if the layout objects do not pass the checking in the first stage. All grid violations have representations (e.g., graphical markers) similar to conventional checking.

There are three basic types of layout objects (or shapes) in any physical design:

1. Line Objects (polysilicon, metals)
2. Point Objects (contacts, vias)
3. Block Objects (diffusions, implants)

In a gridded design, the vertices of all layout objects will lay on a coarse grid called the Layout Base Unit (LBU) grid. The anchors of all shapes—centerlines of line objects, centers of point objects, and edges of block objects—lie on a coarser grid called the placement grid. Placement grids are both layer-specific and orientation-specific. For example, a horizontal metal line will have a different placement grid as compared to a vertical metal line or a horizontal polysilicon line.

The LBU grid may be some arbitrary number for a given technology. The contacted polysilicon pitch (CPP), wire pitches and placement grids must be multiples of the LBU grid. The coarser the LBU, the lower the number of design configurations. Reducing the number of design configurations is better for manufacturability, but it may negatively impact design density. The key to a successful gridded design is to effectively determine the values of the different grids by performing careful design-technology tradeoff analysis.

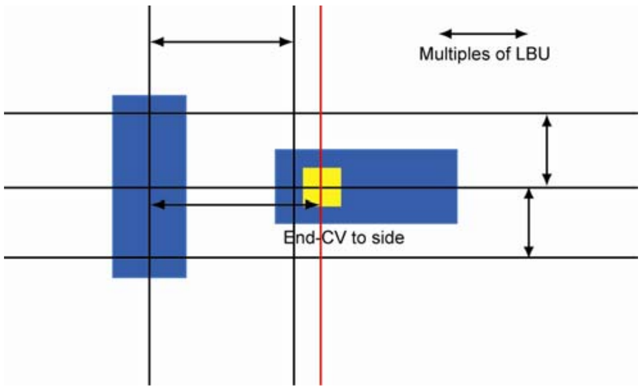


*Figure 4: The LBU grid may be arbitrary, but placement grids and metal/polysilicon widths must be multiples of the designated LBU grid.*

Each layer in the design will have a set of X and Y grids that are legal. For example:

- PC X-grid—centerlines 0, CPP, 2*CPP, 3*CPP, etc.
- M1 X-grid—centerlines 0, CPP/2, CPP, 3/2*CPP, etc.

However, placement periodicity is only a part of grid definition. Each layer will also have a set of valid offsets, or sub-grids. The general form for grid definition is as follows:

$$\text{<level>:<D>} = O_1[O_2, O_3 \ldots] + PG*n$$

where:

- <level>:<D> are the level and orientation (direction) for the grid

- N is an integer (DRC code checks to determine if n is an integer or not)
- $O_1$ is the normalized offset with respect to a global origin, and $O_2$, $O_3$… are optional additional offsets (sub-grids)
- Values are specified in the design rule manual for all Os and PGs for all layers in both X and Y orientations

In the grid example shown in Figure 5, the top and bottom edges of diffusion have different offsets (CPP±x), but the same placement grid (CPP). In the equation, x represents half contact size. Contact on active has a single offset of CPP/2, and a placement grid of CPP.

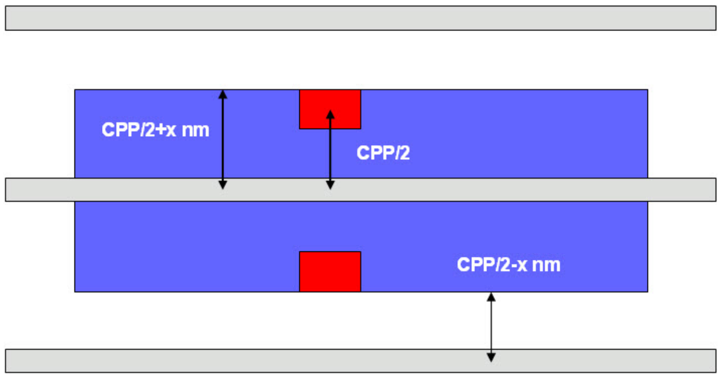


*Figure 5: Grid example showing different grid offsets*

The diffusion grid equation is

$$(-CPP/2-x, CPP/2+x) + CPP*n$$

where global origin is in the center of a polysilicon.

In the RDR configuration shown in Figure 6, restrictive design leaves some edges only roughly specified, with retargeting setting the exact edge positions. The valid minimum for a restrictive design rule is determined by looking at process restrictions (PR1/2/3) and, as necessary, examining each in the specific context. Typically, two or more PRs may define a single RDR (many to one relationship). The balance between competing limitations is used to set the retargeting. In this example, the three effects are:

1. Necking of the vertical Metal1
2. Metal1 tip-to-line shorting
3. Metal1 covering Via1 opens (overlap)

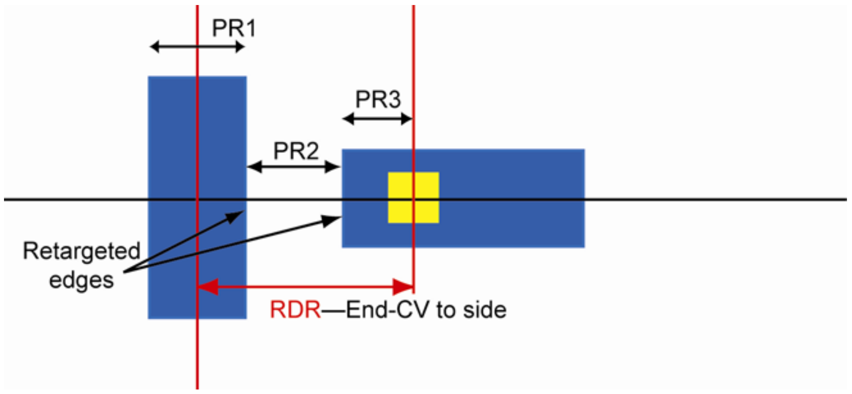


*Figure 6: Retargeting of edge positions in restrictive design*

All of the above grid definitions must be captured in the design rules and be supported by design entry tools. Because these grids are imaginary, design entry tools should provide options for turning on/off the grids on a per level basis. This capability will make manual design entry in a gridded system much more user-friendly. However, automated checking tools must also have a way to deal with grids that are imaginary. The challenges and implementation details of checking restrictive designs are described in the following sections.

## Checking Implementation

Assume some amount or some form of restrictive design is inevitable. What does that mean for physical verification? What has to change in physical verification tools to support this type of layout?

A primary requirement of restricted design is the concept of checking for features "on grid." If regularity is essential, one way to ensure it is to establish a grid and make sure that all features align on that grid. The required check is to ensure that all contacts are centered on intersections of this grid. However, as discussed earlier, this grid exists only in the virtual sense. The ability to establish this virtual imaginary grid and check feature alignment with this grid is one of the capabilities physical verification tools will need to be able to do.

A second requirement is an efficient and accurate way to check pitch. Pitch is similar to grid, in that features should be located at regular intervals (spacings) from each other, whether horizontally, vertically, or in both directions. The difference between pitch and grid is that pitch is a relative measure of a feature vs. its neighbors, rather than absolute grid spacing assigned to the whole layout. Pitch is relative because it is a measure of optical influence, meaning that features within a given distance affect the printability of their neighbors. For example, in checking a gate for pitch, the designer must look at other gates within the specified optical influence distance and make sure the gate is some equal increment of a pitch away from these gates. If the nearest neighbor gate is beyond the optical influence distance, then the gate does not need to be “on pitch” with that other gate. This capability to recognize and measure relative distances also needs to be supported effectively in the physical verification toolset.

Another requirement on the horizon is the ability to do pattern matching – specifying constructs that are (or are not) allowed, and then finding occurrences of those constructs in the layout. This is another approach to checking for restricted design that we see coming in physical verification. However, this paper will focus on the requirements for grid and pitch checking, as these two requirements are well understood and can be implemented today.

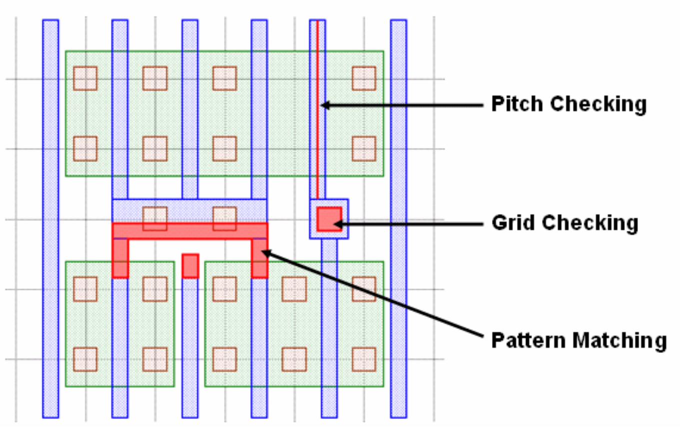


*Figure 7: The three requirements of restrictive design rule checking.*

## a) Grid Checking

As previously mentioned, a grid is a hard-coded, defined grid for an entire chip or a region that consists of a set of invisible lines defined in both X and Y directions. All grids are imaginary constructs—there is nothing on the layout showing the designer the grid. The distance intervals don’t have to be the same for X and Y, and they don’t have to be the same for all layers. Designers can establish one grid for contacts, another for gates, and yet another for metal lines. In all likelihood, these grids probably relate to each other—they’re defined in such a way that if a feature is “on grid” at contact, that point will align itself to the grid of the poly, such that the contacts will land properly, but that is all worked out by the design rule implementer.

Setting up a checking grid begins with defining its origin. Physical verification tools need some flexibility in how the grid origin is defined. For instance, grid origins can be defined by the extent of the chip, so the grid is established at one point for the whole chip and applied from that point across the whole layout. Grid origins could also be defined for separate marker layers independently, so that grids are established for regions. In these cases, the grid in one region may not line up with the grid in another region. Similar to the LBU grids and placement grids, the grid origins for different regions are defined during the technology development.

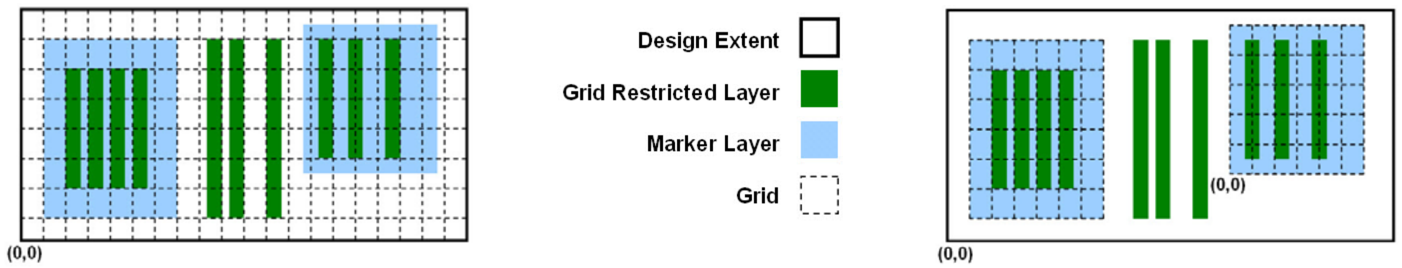


*Figure 8: Grid origins can be defined in relation to the design extent (left) or to shapes in a specified layer (right)*

The next step is to define what determines grid alignment. Physical verification tools must provide easy and convenient ways to derive a wide array of "features of interest" and then measure them against the grid. For instance, the following feature characteristics might be used to define grid alignment:

- Polygon centers
- Polygon endpoints (e.g., every inside or outside corner of the polygon must be aligned to grid)
- Edge endpoints (e.g., endpoints on one edge of a polygon must be aligned to grid)
- Edge centers (e.g., centerpoints of an edge of a polygon must be aligned to grid)
- Centerline edge endpoints (e.g., points on the centerline of a polygon must be aligned to grid)

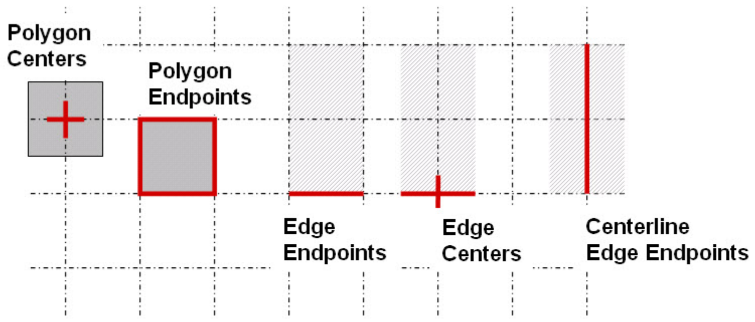


*Figure 9: "Features of interest" used to establish grid alignment*

A more advanced requirement that has been developed is the concept of "offsets," in which some primary grid is established, but some features (e.g., edges of ends of lines) that don't necessarily need to be aligned to the primary grid must instead be aligned to some offset of that grid. These offset alignments occur because the primary grid relates to where the vias land. Metal lines that connect these vias need to line up to the vias on the primary grid, but they need to enclose the via on the line ends by a certain amount, so the line ends must extend past the via by at least that amount. The line end cannot end right on the via grid, but the extension must be some specific distance. In addition to specifying the offset to the grid for the alignment, the physical verification tools must check whether the feature needs to be aligned to the offset on the outer side or inner side of the wire.

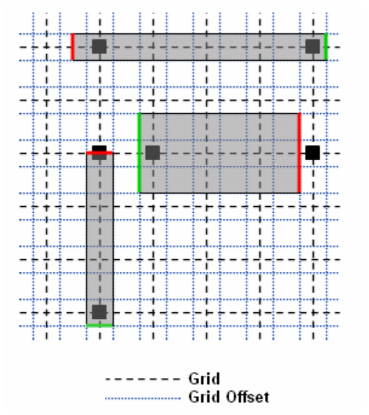
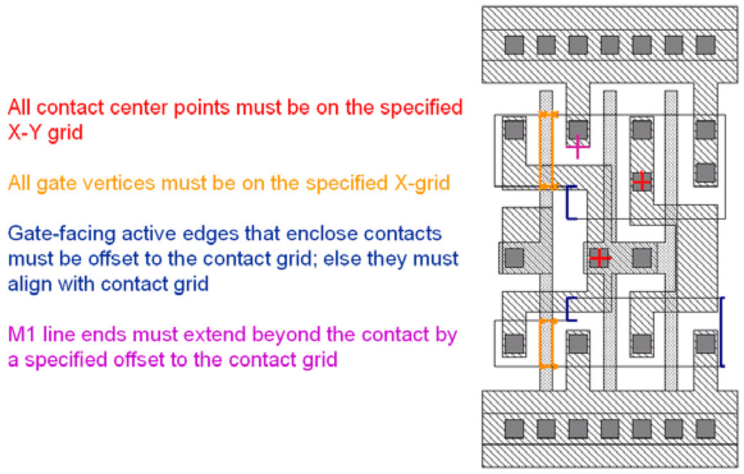


*Figure 10: Checking for grid offset alignment (green indicates a compliant offset, while red shows violations)*

Examples of grid checks include:

- All contact center points must be on the specified X-Y grid (misalignment on one or both axes is an error)
- All gate vertices must be on the specified X grid
- Gate-facing active edges that enclose contacts must be offset to the contact grid; else they must align with the contact grid (if they're not enclosing a contact)
- M1 line ends must extend beyond the contact by a specified offset to the contact grid



All contact center points must be on the specified X-Y grid
All gate vertices must be on the specified X-grid
Gate-facing active edges that enclose contacts must be offset to the contact grid; else they must align with contact grid
M1 line ends must extend beyond the contact by a specified offset to the contact grid

*Figure 11: Grid checking examples*

Determining how to fix grid errors during checking debug is extremely difficult. Not only is the grid invisible, but given all the ways to define a grid, it can be very challenging to mathematically calculate where the grid would be lying. There are so many factors to consider when an error occurs (origin points, offsets, X and Y measurements, etc.) that it's very difficult for the designer to figure out how to fix the error. The designer has no way to know where that grid actually lives. The physical verification tools must be able to not only find the grid misalignment errors and mark them, but also supply correction hints that provide a visual guide to the nearest allowed grid point location(s) for that feature.

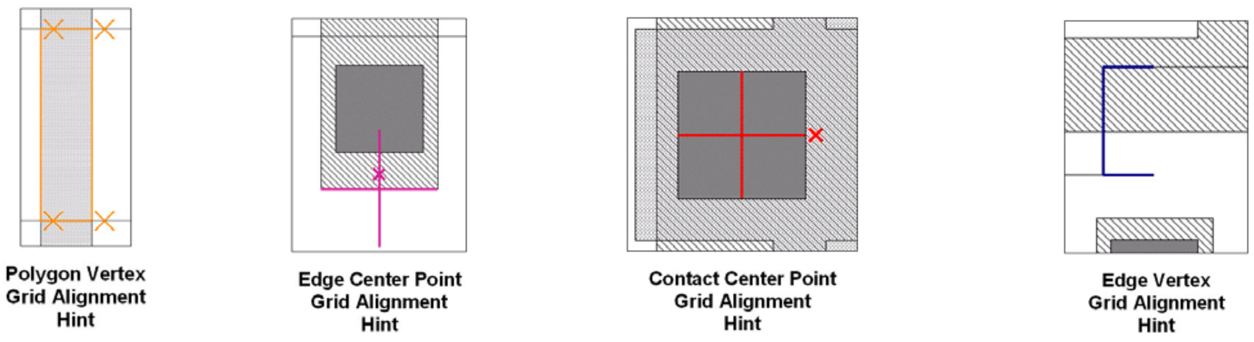

Polygon Vertex Grid Alignment Hint
Edge Center Point Grid Alignment Hint
Contact Center Point Grid Alignment Hint
Edge Vertex Grid Alignment Hint

*Figure 12: Examples of visual grid alignment “hints” to assist the designer during error correction*

An unusual example of grid checking is the condition that all cell pin rectangles must fully enclose at least one of the Metal2 routing grids. In the cell shown in Figure 13, certain polygons of Metal1 make up the landing pads for the pins that will be hooked up when this cell is placed into a design. Because the Metal2 routing is gridded, the Metal2 wires will only be allowed on certain grid routing channels at the next layer. The landing pads for these pins must fully enclose at least one of those routing channels (grid “square”), to ensure it is possible to put a via from Metal2 down to the Metal1 pin and get full coverage. Since this cell hasn’t been placed in a design yet, it doesn’t matter which routing channel it’s going to be in, but it must fully enclose at least one. The challenge is that the Metal2 grid is not accessible when looking at this cell—in fact, there is no Metal2 within the cell. New equation-based physical verification techniques enable these types of restrictions to be implemented as the grid coverage (or enclosure) can be mathematically calculated. In addition to calculating the coverage, equation-based checks can also calculate how much the edges of the pin need to move when there is a violation, which greatly improves error debug.

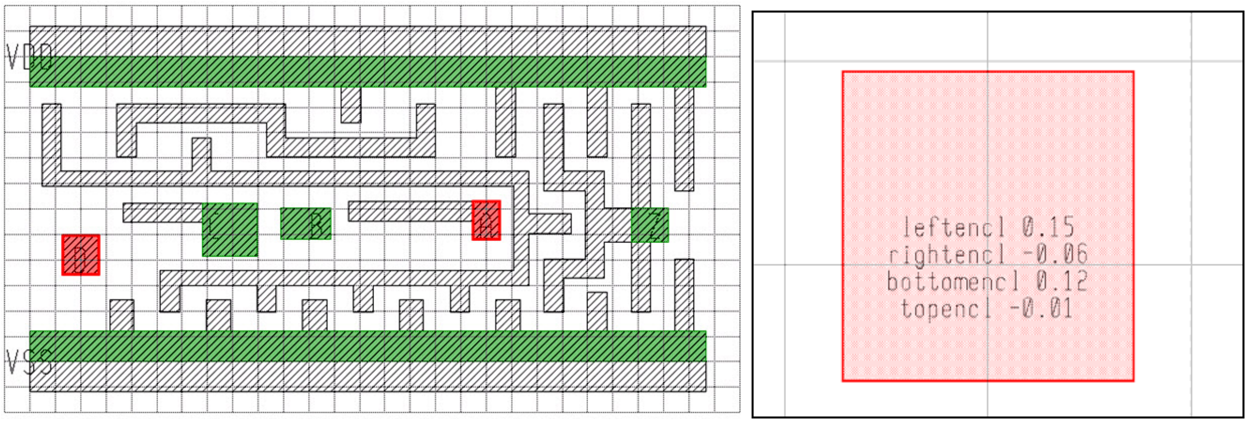


*Figure 13: Cell pin rectangles on Metal1 must fully enclose a Metal2 routing channel aligned to the Metal2 grid. Property values on error marker calculate how much positive or negative enclosure exists, so designers can determine how best to fix the violation.*

## b) Pitch Checking

What is meant by pitch, or, more specifically, relative pitch, in regard to performing a check? The simplest way to think of pitch is to add the space between two features to the width of one of the features, creating an allowable “pitch.”

Many people think about pitch as “centerline to centerline,” so by measuring from the center of one feature to the center of the next feature (which includes the space and half of each of the widths), the allowable pitch is established. To put it into mathematical terms:

$$\text{Pitch} = \text{space} + \text{width} = x$$

Pitch is relative because pitch only looks at a feature in relation to its neighbors. It differs from grid-based constraints in that beyond a certain distance, pitch can be ignored. For example, the pitch distance may be a measure of the optical radius (influence distance of the reflected light during the optical process). By allowing features beyond the influence distance to be “off grid” in relation to the features within the influence distance, it provides a less-regimented layout for the designer without affecting printability. For example, assume the optical radius is 10 times the pitch. When trying to print a given feature, every other feature within 10 pitches of that feature will have some influence on the ability to print that feature properly. Features beyond that optical radius of the stepper do not affect the given feature. Because everything within that pitch distance has a significant impact, those features need to be placed on some even number of pitches from the given feature so that regularity is established in the light/dark contrast of the stepper, ensuring the resolution will be high.

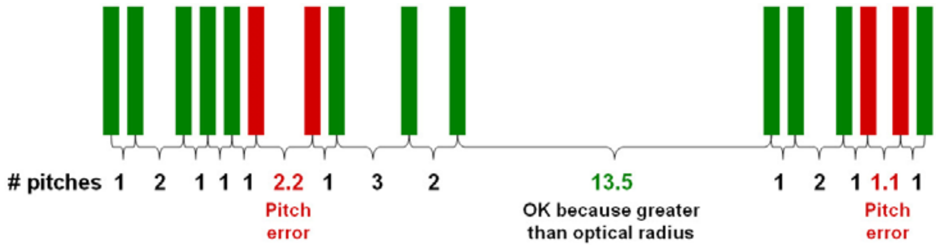


*Figure 14: Pitch spacing must be regular within the optical radius to ensure regularity in stepper operations*

Features are considered grouped if they are within an optical radius of each other. Features within a group are pitch-restricted only in relation to each other. Features in separate groups (more than an optical radius away) do not need to be constrained by pitch in relation to each other.

Sample pitch checks, as shown in Figure 15, might include:

- Metal routing wires must be on specified relative pitch within an optical radius
- All gates within the same active must be on specified relative pitch
- All contacts must be on a specified relative pitch

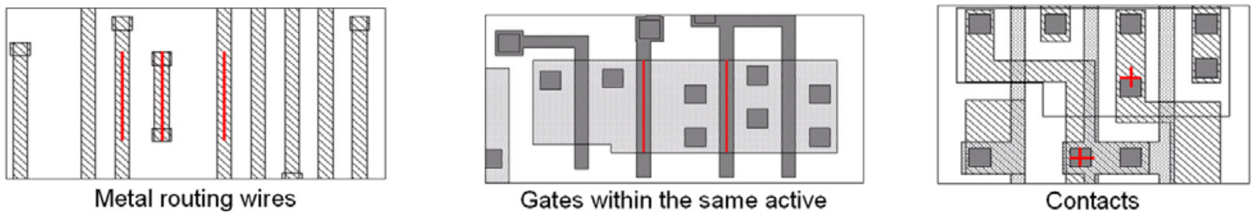


*Figure 15: Sample pitch checks*

Locating and debugging pitch errors can be extremely challenging. Relative pitch is not readily visible to the layout designer. If a pitch error occurs, the designer must know what the relative pitch is, and determine how far a feature must be moved to be on pitch. There are no markers on the layout to show pitch, because it is mathematically calculated “on the fly” relative to a feature’s neighbors, which makes pitch very difficult to visualize. When pitch errors occur, designers have traditionally used measuring tools to physically measure features to calculate the pitch, then divided the measurements by the pitch to try to locate the pitch failure.

Not only does a physical verification tool have to be able to calculate pitch and perform a wide variety of pitch checks, but it also must be able to calculate ways to fix the errors and provide that information to the designer. The physical verification tool should provide correction “hints” to the designer that include all the necessary information, such as:

- The centerline to centerline measurement of two features(eliminates physical measuring by designer)
- The percentage of the calculated pitch that the first measurement represents (anything other than a whole number identifies a pitch error)
- The percentage that the feature is “off pitch” (this allows the designer to sort and prioritize pitch errors)
- Calculated distance feature must be moved to be on pitch (eliminates manual correction calculations by designer)

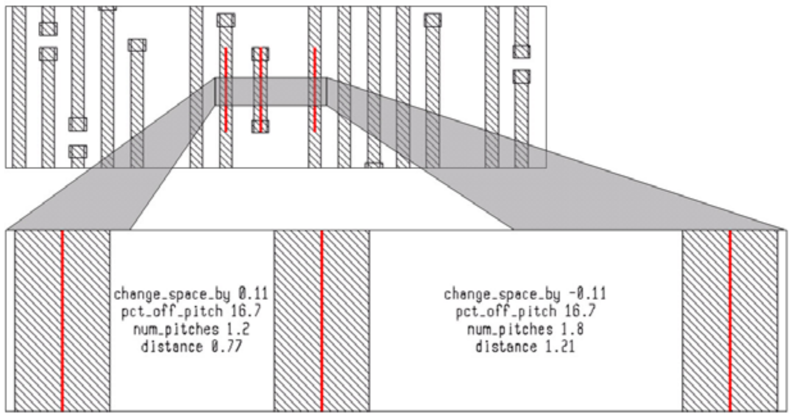


*Figure 16: Providing correction hints for pitch correction is an essential function of automated pitch checking*

Again, new equation-based physical verification techniques allow not only easy mathematical calculation of pitch, but also the ability to calculate how much the centerline of the wire must be moved to fix the violation. Figure 16 shows several equation-based properties that were calculated for the two errors to assist the designer in understanding the violation and how to correct it.

## *Summary*

This paper provides a introductory description of what restrictive design is all about, how it might impact your job as a designer or physical verification engineer, and how new capabilities in physical verification tools can help you both check and debug these types of issues.

It is clear that new techniques and design approaches will be needed at 22 nm to ensure continued production. Because pitch and grid are critical to effective restrictive design, but rely heavily on mathematical calculations for accurate implementation, it is essential that the physical verification tools not only be able to perform the design rule checks associated with pitch and grid, but that they also supply the designer with the information needed to debug and correct any errors associated with these checks. Without automated debug assistance, it would be nearly impossible to produce DRC-clean restrictive designs in a timely and efficient manner.

It is still unclear to what extent restrictive design practices will be enforced, but most industry experts agree that some form of this approach will be necessary. In many ways, this approach actually reduces complexity in regard to the number of rules and ease of manufacturability. The trade-offs are a change in design methodology, as well as an increase in the abstraction of a design rule in a mathematical sense. It is this abstraction from simple one-dimensional measurements to multi-dimensional mathematical relationships that is dramatically impacting the traditional capabilities of design rule checking tools. Thanks to new innovations in physical verification capabilities such as equation-based checks and advanced grid definition, restrictive design can be implemented today, ensuring manufacturable designs at 22 nm. In addition, advances in error correction hinting, including mathematical property calculations and nearest grid point markers, enable the designer to meet these new requirements without sacrificing time-to-market goals.