New DFM Approach Abstracts AltPSM Lithography Requirements for sub-100nm IC Design Domains

Pradiptya Ghosh, Chung-shin Kang, Michael Sanie and David Pinto
Numerical Technologies, 70 West Plumeria Dr., San Jose, CA 95134

Abstract

Since the semiconductor industry hit the 0.18-micron generation, device feature sizes have become increasingly smaller than the wavelength of light used by available optical-lithography equipment. In this subwavelength arena, manufacturing requirements must be handled up front in the IC design stage—while changes can still be made—to enhance quality and yield. This paper defines the components needed to get clean alternating phase-shifting masks (altPSM) that ensure the manufacturability of subwavelength circuit designs. The authors present a new design for manufacturability (DFM) approach, creating an abstract set of rules that can be used to advantage in various IC CAD tool domains, especially for 100nm and below design rules. A new methodology and algorithm are presented that can quickly and easily integrate altPSM into existing and future tools earlier in the IC design flow. Finally, experimental results show how the methodology and algorithm is used to debug process-aware designs and make them altPSM-compliant.

1. Introduction

The shift to the subwavelength realm [1] has impacted semiconductor manufacturing and physical design at all levels, prompting both sides to explore DFM solutions that improve quality, predictability and yield. In subwavelength manufacturing, optical distortions and other process (etch, resist) effects deform the patterns, which are described in the design (GDSII) and printed onto a photomask. Line-width variations and other distortions significantly decrease IC performance. In the worst case, they could cause missing, incomplete or shorted structures that result in hard failure.

A 10% margin for proximity errors is now unacceptable, driving the need for post-GDSII design corrections. Optical Proximity Correction (OPC) and attenuated PSM help correct subwavelength distortions for control of critical dimensions (CD), and have become an integral part of the lithography process [2]. With advances in the process node, these traditional methods may soon reach their limit. For example, at the 90- and 65-nm nodes the minimum gate lengths will be 50nm and 35nm, and the poly-interconnect line widths will be 90nm and 65nm, respectively [1]. This will force manufacturers to utilize other more enabling approaches such as alternating phase-shift masks (altPSM).

In 1982, Levenson et al [3] proposed the concept of using the phase information of light to improve lithographic patterning properties. AltPSM is based on these principles, wherein destructive optical interference helps improve depth of focus and resolution in lithography. The technique involves etching selected areas of regular chrome-on-glass masks to selectively change the phase of light used to expose IC features on a silicon wafer.

Advanced ICs are now produced at 0.1-micron design rules, with transistor gate sizes of 70nm or just one-third of the actinic wavelength. Combined with OPC techniques, altPSM has presented the most effective solution to date for these process generations. Possibly its most important benefit is the improved statistical process control of the CD, which results from both the large depth-of-focus and the favorably low values (~0.5) of mask-error-enhancement-factor (MEEF), as has been experimentally demonstrated through pitch and even at defocus [4].

This paper focuses on altPSM, proposing a simplified view that will help designers to know whether their designs are altPSM-compliant. Furthermore, experimental results show that it is easy to implement altPSM-compliant IC CAD tools by making incremental changes in terms of the methodology and algorithms used in the design flow.

2. Need for a DFM approach

The subject of using OPC and altPSM has been covered in detail in the survey paper [5]. Most of the prior focus was centered on post-processing modifications, which works with minor geometric corrections like OPC. With altPSM, the phase-shifter geometry and topology must be planned earlier in the design phase.

Many examples indicate that subwavelength requirements must be handled earlier in the design process. One example is the phase conflict—an area where locations that are reserved for phase-shifters of opposite phase overlap, resulting in a misprinted feature. Resolving a phase conflict is a full-chip challenge, since addressing a phase conflict in one corner of the design might force conflicts elsewhere. Resolving conflicts requires making adjustments to design spacing and topology. These adjustments should be made during physical design—not during post-tapeout processing. Previous research has proposed doing this in the library flow [6].

0-7695-1881-8/03 $17.00

The DFM solution proposed in this paper defines the problem abstractly and applies specific rules and algorithms to various domains of IC development, such as verification and P&R. As indicated in earlier studies, the first step is to identify and correct only those geometries that can be made altPSM-compliant [7]. Checking whether the shifter can be added to the geometry using only two phases has also been studied [8].

Section 3 defines the concepts around altPSM and related terminologies. Section 4 formally discusses the problem. Section 5 outlines the methodology, the rules and algorithms. Section 6 presents studies on full chips and blocks with experimental results.

3. AltPSM terms and constraints

The altPSM technology proposed by Pati and Wang [9] enables clear regions of a mask to transmit light with a prescribed phase shift. Light entering the two transparent regions and the diffracted light cancel each other as they are out of phase (figure 1), improving image contrast.

These open regions are called *shifters*. One of the shifters has the same phase of light as other openings in the mask and is called *0 phase shifter*. The other opening (around a geometry) has a phase shift of 180 degrees and is called *180 phase shifter*. Any geometry that can benefit from an altPSM approach is called a *critical polygon* in this paper.

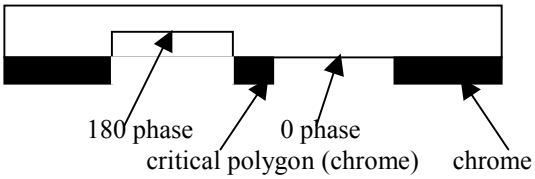

Figure 1: Alternating phase shifting mask.

Two positive constants (b < B) define a simplified relationship between printability and the distance between two clear regions (or between shifters) [5]. Now any two shifters that are separated by a distance less than B will share the same phase.

If phase assignment leads to a critical polygon having the same phase shifters on either side, then the two shifters are said to be in *phase conflict*.

Another factor to consider is the phase-assignment constraint (figure 2), whereby all shifters of a given layout that are conflicting are assigned the opposite phase.

The manufacturing process introduces further constraints. Thus, a design that does not meet the phase-assignment constraint cannot use altPSM technology. But the converse is not necessarily true, in that a design that does meet the phase-assignment constraints may also be unsuitable for altPSM technology.

This paper discusses the additional conditions that must be met for a design to qualify for the benefits of the altPSM approach.

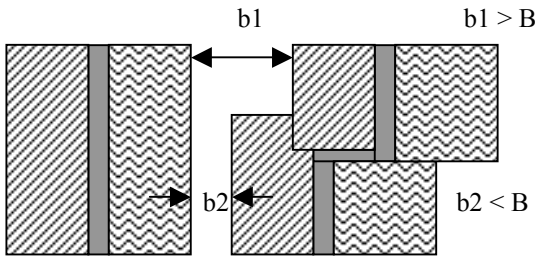

Figure 2: Phase-assignment constraint.

Another important term used is the *endcap*. It is located at the end of a polygon abutted to or along the critical polygon. Depending on the application type used (detailed in the following sections), it may appear in the form of a polygon or just an edge. It inherits all the placement rules from critical polygon and is treated as the extension of a critical polygon.

Finally, we call the rest of the polygon (excluding the critical and endcap polygon area) a *field* polygon. It is important to understand that both endcap and field polygons are *non-critical polygons* (i.e., they do not need altPSM). The main purpose of having two different types of non-critical features is to apply different rules to them, therefore improving CD control and/or preventing line-end shortening (endcap). Non-critical polygons also signify change in phases for a critical polygon and we often refer to them as the exit of a critical polygon. If we add shifters to the endcap or field polygons, we call them *endcap shifters* or *field shifters* respectively.

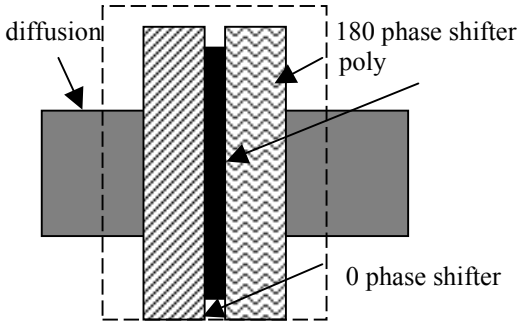

Figure 3: Full-feature altPSM.

When altPSM is applied to the entire polygon of a layer, it is called *full-feature altPSM* (figure 3). In other words, all geometries in that layer are considered critical. Moreover, it requires very complicated methods when selecting critical and non-critical polygons.

4. Problem definition

A design's compliance with altPSM technology can be divided into three main categories. The first check is to see if there is any critical polygon that--in isolation--cannot be made using altPSM. This is resolved in the research paper [7]. Next step is to check if the geometries are laid out in such a way that manufacturable shifters can be applied. This process helps to detect and remove any manufacturability and colorability problems that are local between edges and are directly in line of sight with each other. Finally we need to verify that the shifter created does not create any global colorability problems. Other papers have solved this problem [8] [10].

4.1. Manufacturability and local inter-geometry colorability problem

From an altPSM manufacturability perspective, there are three types of broad errors. They are shifter error, margin error, and endcap error. The inter-geometry colorability problem is a subset of the global colorability problem. It can be detected locally, which is easier than detecting the global colorability problem.

Shifter error: This error happens either between critical polygons or between critical and non-critical polygons (figure 4). This is a fatal spacing problem, since it states that within the given technology there is no way to insert shifters without violating the manufacturing spacing and size requirements.

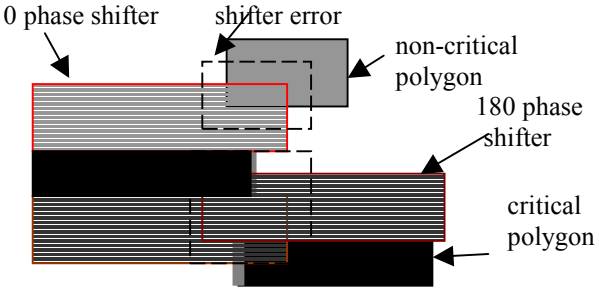

Figure 4: Shifter errors.

Endcap error: This kind of error occurs at line ends (endcap edge away from critical polygon) where a polygon is very close to an endcap (figure 5). This is important to consider when checking the quality of the critical polygon close to the line ends, but is not as crucial as shifter errors.

Margin error: Endcap and field shifters are often added to improve CD control near the junction of critical and non-critical features. Endcap shifters can also prevent line-end shortening. They may be reduced or dropped entirely if they cause manufacturing problems. These spacing violations are called margin errors (figure 6).

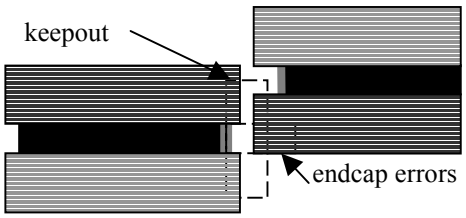

Figure 5: Endcap errors.

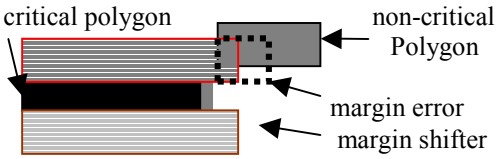

Figure 6: Margin errors.

4.2. Local inter-geometry edge color problem

These are coloring problems that occur in a space surrounded by edges. A good example is the inverse T (figure 7). Unlike the global coloring problem, it is local to a space and can be detected by simple DRC rules.

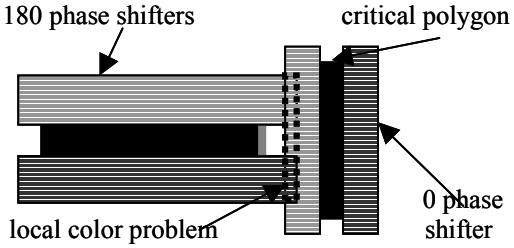

Figure 7: Local inter-geometry edge color problem.

5. Proposed solution and methodology

We propose a set of checks that can be applied to a given design layout or used as constraints for layout creation. If a design passes, it qualifies as altPSM-compliant.

The methodology is to detect each of the geometries independently (single geometry colorability problem). We propose that each one is qualified or changed such that each one meets the phase-assignment constraint individually and is also manufacturable [7]. Next, we look at all the spaces between the polygons in the design to make sure they can satisfy both manufacturing and process constraints. Finally, we look at the entire geometry and try to solve the potential shifter phase-assignment problem [4].

5.1. Manufacturability and local inter-geometry colorability solution

These problems can be converted into edge-spacing problems. The trick is to have the right set of edge types and then come up with the correct spacing requirements between the various edges. Section 5.2 discusses the various classes of edges while Section 5.4 classifies the various checks. Section 5.5 shows how each check maps into the various problems. Section 5.6 provides the algorithm used to apply the checks.

5.2. Edges classification

Figure 8 depicts the various classes of edges described in the following sections.

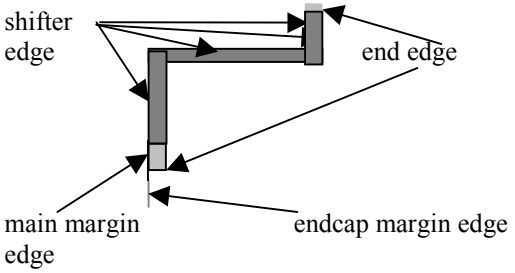

Figure 8: Edge classification.

endEdge: This is the edge of the endcap that is opposite to the edge abutting the critical polygon.

shifterEdge: This is the edge of the critical polygon where the shifter will be placed.

mainMarginEdge: This is the extension of the shifter edge. It may abut to a field or endcap polygon, in which case the shifter placed is called field or endcap main margin shifter respectively. It shares the same placement parameter—shifter width—as that of the main shifters.

endMarginEdge: This is the extension of the mainMarginEdge away from shifterEdge. These shifters are called endcap or field shifters. Unlike margin shifters, they have their own endcap or field shifter widths.

5.3. Region definition

Figure 9 shows the two "keep outs" that we care about. Based on these two clearance requirements, along with the minimum shifter width and endcap shifter width for a given process technology, we break the endEdge region into four regions as listed below:

LineEndRegion: This is the region directly above the endcap.

EndcapShifterRegion: This is the region around the endEdge where an endcap shifter would extend.

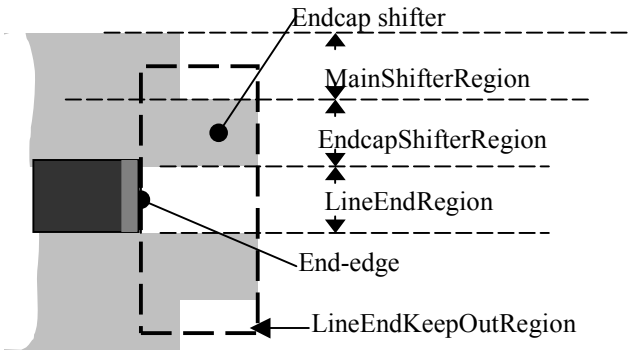

Figure 9: Various regions.

MainShifterRegion: This is the region around the endEdge where a minimum width shifter would extend beyond the endcapShifterRegion.

LineEndKeepOutRegion: This is the region around the endEdge (not endEdge) where no other shifter should be created.

5.4. Checks classification

Endcap to endcap checks: These are checks between the two endcaps. Different manufacturers specify special clearance requirements at the endcaps. There are four line-end to line-end checks. They are all separation checks, as specified in figure 10.

Check11. Detects distortion to both the line ends due to proximity.

Check12. Detects distortion in the other line end due to the *endcap shifter* from the original line end.

Check13. Detects distortion in the other line end due to the *shifter* from the original line end.

Check14. Mask manufacturer or the fabrication process could specify this clearance region.

Endcap to shifter edge checks: These checks help to avoid line-end distortion due to soft coloring error (interference due shifters created for other critical polygon). There are four checks possible (figure 11).

Check31. Detects distortion in the line end due to another non associated shifter.

Check32. Detects mask manufacturing problems due to minimum space violation between unbalanced shifter and endcap shifter.

Check33. Detects problem as check32 but between main shifter and unbalanced shifters.

Check34. Detects unbalanced shifters that distort geometry.

Line-end to non-critical polygon checks: These are checks to detect if there is a non-critical polygon near an endcap. Although a typical non-critical polygon is wider than a critical polygon, it may get distorted due to its proximity with the shifters around the critical polygon. Figure 12 proposes two checks to detect this.

Check21. Detects deformation of non-critical polygons due to endcap shifters.

Check22. Detects deformation of non-critical polygons due to main shifter.

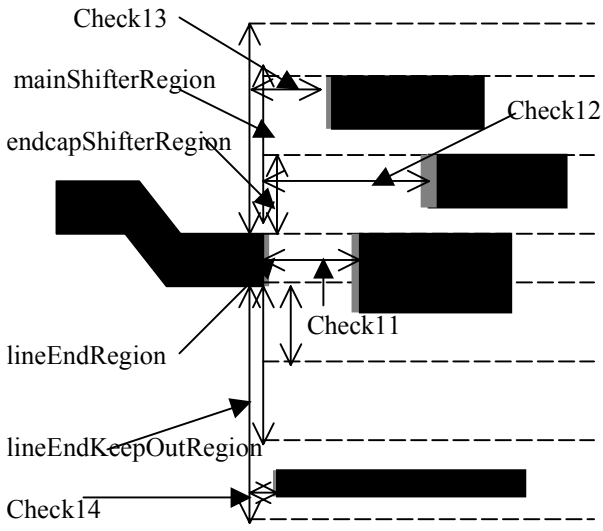

Figure 10: Endcap to endcap checks.

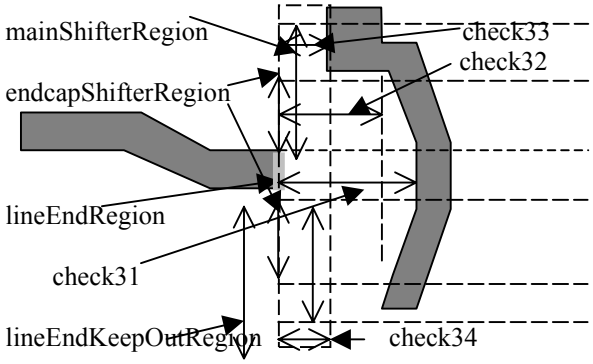

Figure 11: Endcap to shifter edge checks.

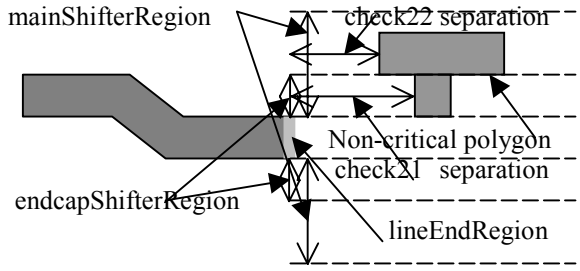

Figure12: Line end to non-critical polygon check.

Shifter edge checks: These two checks try to detect if any manufacturing violations may result when shifters are put on these edges.

Check41. This checks whether there is any shifter edge along the given shifter edge within the check41 separation (figure 13). A violation of this check means the space between the two shifter edges is too narrow to add another shifter and could lead to mask manufacturing problems.

Check51. This checks whether any non-critical polygon (figure 13) is too close to the shifterEdge.

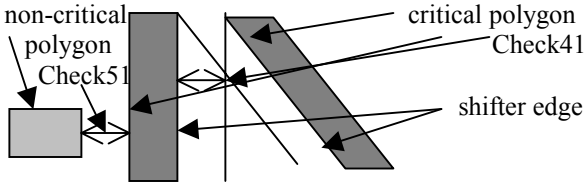

Figure 13: Shifter edge checks.

Margin area checks: The margin area as mentioned earlier is the region around the endEdge where shifters (both margin and endcap types) are added to avoid the distortion of the endEdge. We recommend two checks (figure 14) that try to detect any distortion to the non-critical geometries due to margin shifters. They also check to see whether there is enough space to insert the margin shifters.

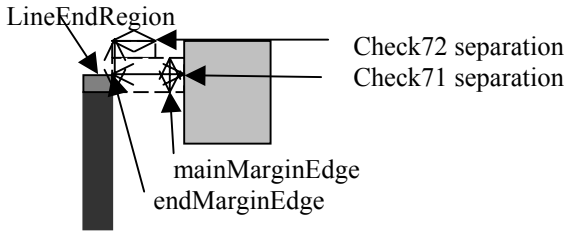

Figure 14: Margin area checks.

Check71. This tries to detect if there is any non-critical polygon within the check71 separation along the mainMarginEdge.

Check72. This tries to detect if there is a non-critical polygon within the check72 separation along the endMarginEdge.

5.5. Mapping checks to problems

Table 1 shows how each check maps to different problems. Each check has a primary dominant problem that causes it, and other secondary side effects.

Table 1: Mapping between error types and the cause of the problems (S/E=side-effect).

Error type	Endcap Errors	Shifter Errors	Margin errors	Local coloring errors
1.1	**Dominant**			
1.2			S/E	**Dominant**
1.3			S/E	**Dominant**
1.4	**Dominant**			
2.1			**Dominant**	
2.2			**Dominant**	
3.1	**Dominant**	S/E		
3.2	S/E	S/E	S/E	**Dominant**
3.3	S/E	S/E	S/E	**Dominant**
3.4	**Dominant**	S/E		
4.1		**Dominant**		
5.1		**Dominant**		
7.1		S/E	**Dominant**	
7.2		S/E	**Dominant**	

5.6. Algorithm

We recommend a simple algorithm for detecting the various error types. It is similar to those used to check design-rule violations.

```
  begin
  /* Get the checkboxes to be used for
detecting various errors */
  myCheckBoxCt = NTIPSgetCheckBoxes(true,
  checkBox);
  /*For each of the checks apply it on the
edgeToApply type of edges. This edge being
searched is edgeToSearch */
  foreach myCheckBox in checkBox
   do
     begin
       /* Get the edges of the type
     edgeToApply*/
       for myEdges of type myCheckBox->
       edgeToApply;
         do
           case(myCheckBox->checkType)
            edgeCheck : do
              myNewEdge is myEdge
   offEdgeCheck: do
  create myNewEdge that is offset by
  myCheckBox->parallelOffset and spans
  myCheckBox->parallelSpan on both ends of
  the myEdge.
   overEdgeCheck: do
create myNewEdge that spans myCheckBox->
parallelSpan beyond the two ends of myEdge.
   case end
   /* create the search box at an offset of
perpendicularOffset and
   extends perpendicularSpan */
   error = separation check to find
myCheckBox->edgeToSearch within
  myCheckBox-> perpendicularSpan but
greater than
  myCheckBox-> perpendicularOffset on
myNewEdges
  finalError[myCheckBox->type] = error
 end //for loop
end
```

Figure 15: Algorithm applying the various errors to detect manufacturability and DRC problems.

In the algorithm (figure 15), the check box has six basic parameters:

1. Edge to apply the check box (edgeToApply)
2. Edge to search in the check box (edgeToSearch)
3. Offset in the direction of the edgeToApply beyond the edgeToApply (parallelOffset)
4. Offset perpendicular to edgeToApply (perpendicularOffset)
5. The distance over which the check spans (parallelSpan)
6. The separation check itself (perpendicularSpan)

These check boxes are created based on the technology parameters used and correspond to the checks in Section 5.4. The algorithm is the driver program that uses these checks to apply them on a given layout.

In the case of an IC CAD P&R tool like a router for example (where the driver engine is a creation engine), each of the checkboxes would be converted into constraints. Thus in this situation, instead of simply detecting final errors, the driver engine would prevent these errors.

6. Experimental setup and results

We performed verification on real designs targeted at the 0.13- and 0.18-micron process nodes with 248DUV. We ran two test cases for full-feature altPSM. Our aim was to detect if there were any potential PSM design errors using psmLint, and to qualify them using Cadence's Assura PSM tool. We were also interested in other benefits like performance gain or in terms of helping users to quickly resolve the PSM design errors.

6.1. Setup

We used Numerical Technologies' PSM technology for design placement and phase assignment. We used the Assura environment (versions 2.0 and 3.0) as the geometry engine for the edge detection as well as for applying the checks to designs and creating error markers accordingly. We implemented the checks based on Numerical's process technology in C++. The algorithm was in SKILL (Assura extended language). For the global colorability check, we placed coarse shifters in the Assura environment and the graph color conflict detection was

done using Numerical's PSM engine (iN-PhaseTM). Runs were made on a single-processor Solaris7 Sun Ultra-4.

6.2. Results

For performance gain/PSM design error comparison, all these test cases were run through Assura PSM normally. We then ran the same tests through the psmLint flow and tweaked some of the technology parameters (within the process limits).

We also ran psmLint and Assura PSM and compared the PSM design errors found. Next, we estimated the runtimes to process the design, both with and without psmLint. One component that we could not quantify was the time that users could save as a result of sub-typing or categorization of errors.

We ran two test cases for full-feature altPSM (table 2); the first a RAM block and the second a control block from a fab vendor at 0.18-micron technology.

Table 2: Full-feature PSM results.

Test case (process node; polygon)		**Memory and control block (180nm; 558K)**	**Control block (180nm; 83K)**
Original PSM run time		12 minutes	20 minutes
psm-Lint run result	**Run time**	<80 sec	<60 sec
	Initial errors	~700 (~20 basic types)	369 (~10 basic types)
	Final errors	512 (1 basic type)	5 (1 basic type)
	Iterations	2	3
Final PSM run	**Run time**	6 min 25 sec.	512 min 8 sec.
	Final errors	512 (1 basic type)	5 (1 basic type)
Flow run-time	**Without psmLint**	16+ min (2* 12 min)	60+ min (3 * 20 min)
	With psmLint	<9min (2 * 80 sec + 6.5min)	~23 min (3* 60sec +20min)

7. Conclusion

We found that all the PSM design errors found by the Assura PSM design run were detected by the psmLint run. We observed that the Assura-psmLint run was considerably faster. By using the cause-effect information we could find a compatible technology or change the design quickly (based on the error markers) for several design styles and intents.

We also discovered that Assura PSM ran significantly faster when using a compatible technology. Hence, by using psmLint, a process engineer could quickly match the manufacturing technology parameters to a specific design style. Conversely, a designer can easily qualify a design for a given technology node.

8. Future work

Existing and next-generation CAD tools can incorporate these checks as part of the constraints in placement and routing to help designers create altPSM-compliant layouts that are manufacturable in the first pass. These rules are also important in that they allow floorplanning of regions that are targeted for different technologies.

Layout engineers could also use this work to understand the impact of process information earlier in the design domain. Thus, further enhancements to the layout editor and all other physical design tools using these rules could help communicate more information and improve design for manufacturability. Our work currently targets four broad categories of problems. We will further refine and expand on this work to include newer altPSM application approaches [10].

References

1. Semiconductor Industry Association, ITRS 1999 edition, Austin, TX: International SEMATECH, 1999.
2. C.H. Park, Y.H. Kim, H.J. Lee, J.T. Kong, S.H. Lee, A practical approach to control full chip level gate CD in DUV lithography, SPIE vol. 3334,1998.
3. M.D. Levenson, N.S. Vishwanathan, R.A, Simpson, Improving resolution in photolithography with phase shift masking, IEEE Trans. on Electron Devices, 29(11),1982,1828-1836.
4. G. Vandenberghe, F. Driessen, Paul van Adrichem, K. Ronse, J.Li and L. Karklin; Performance Optimization of the Double Exposure Alternating PSM for sub-100nm ICs, SPIE 2001 Proceedings, Vol. 4564.
5. A.B. Kahng, Y.C. Pati, Subwavelength Optical Lithography: Challenges and Impact on Physical Design, Proc. ACM Intn. Symposium on Physical Design, April 1999, pp 112-119.
6. M. Sanie, M. Cote, P. Hurat, V. Malhotra, Practical Application of Full-Feature Alternating Phase-Shifting Technology for a Phase-Aware Standard Cell Design Flow, Design Automation Conference 2001.
7. L. Liebmann, J.Lund, F. Heng, I. Gruar, Enabling Alternating Phase Shift Mask Design for a full Logic Gate Level: Design Rules and Design Rule Checking, Design Automation Conference 2001.
8. P. Berman, A.B. Kahng, D. Vidhani, H. Wang, A. Zelikovsky, Optical Phase Conflict Removal for Layout of Dark Field Alternating Phase Shift Masks, Proc. ACM International Symp. on Physical Design, April 1999, pp 121-126.
9 Y.T. Wang, Y.C. Pati, Phase Shifting Circuit Manufacture Method and Apparatus, U.S.Patent 5,858,580, 1999.
10. C. Pierrat, M. Cote, K. Patterson, New Alternating Phase-Shifting Mask Conversion Methodology Using Phase Conflict Resolution, SPIE 2002, Vol.4691.