Performance-Driven OPC for Mask Cost Reduction

Puneet Gupta†, Andrew B. Kahng†, Dennis Sylvester‡ and Jie Yang‡
‡ EECS Department, University of Michigan at Ann Arbor
† ECE Department, University of California at San Diego
(puneet, abk@ucsd.edu), (dennis, jiey@eecs.umich.edu)

ABSTRACT

With continued aggressive process scaling in the subwavelength lithographic regime, resolution enhancement techniques (RETs) such as optical proximity correction (OPC) are an integral part of the design to mask flow. OPC adds complex features to the layout, resulting in mask data volume explosion and increased mask costs. Traditionally the mask flow has suffered from a lack of design information, such that all features (whether critical or non-critical) are treated alike by RET insertion. A recent work [1] proposes to exploit design information (timing slacks) to reduce OPC data volume, but has a number of impractical aspects. In this paper, we propose an *implementable* flow that drives model-based OPC explicitly by timing constraints, with the objective of reducing mask data volume and OPC runtime. We apply a mathematical programming based slack budgeting algorithm to determine edge placement error (EPE) tolerance budgets for all polysilicon gate geometries. These tolerances are then enforced by a commercial OPC tool to achieve up to 24% MEBES data volume and 41% OPC runtime reductions on a suite of six testcases implemented in Artisan TSMC 0.13um libraries.

1. Introduction

Continued technology scaling in the subwavelength lithography regime results in printed features that are substantially smaller than the optical wavelength used to pattern them. For instance, modern 130nm CMOS processes use 248nm exposure tools, and the industry roadmap through the 45nm technology node will use 193nm (immersion) lithography. The International Technology Roadmap for Semiconductors (ITRS) [2] identifies aggressive microprocessor (MPU) gate lengths and highly controllable gate CD control as two critical issues for the continuation of Moore's Law cost and integration trajectories. To meet ITRS requirements (see Table 1), resolution enhancement techniques (RETs) such as optical proximity correction (OPC) and phase shift masks (PSM) are applied to an increasing number of mask layers and with increasing aggressiveness. The recent steep increase in mask costs and lithographic complexity due to these RET approaches has had a harmful impact on design starts and project risk across the semiconductor industry.

OPC and Mask Cost. In this work we focus on OPC, which is a major contributor to mask costs as well as design turnaround time (TAT). More than a 5X increase in data volume and several days of CPU runtime are common side effects of OPC insertion in current designs [3]. With respect to the cost breakdown shown in Figure 1, OPC affects mask data preparation (MDP), defect inspection (and implicitly defect repair), and the mask-writing process itself. Today, variable-shaped electron beam mask writers, in combination

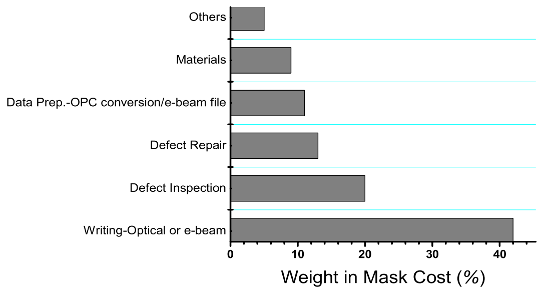


Figure 1: Relative contributions of various components of mask cost [6].

with vector scanning[1], comprise the dominant approach to high-speed mask writing. In the standard mask data preparation flow, the input GDSII layout data is converted into the mask writer format by *fracturing* into rectangles or trapezoids of different dimensions. With OPC applied during mask data preparation, the number of line edges increases by 4-8X over a non-OPC layout, driving up the resulting GDSII file size as well as fractured data (e.g., MEBES format) volume [4]. Mask writers are hence slowed by the software for e-beam data fracturing and transfer, as well as by the extremely large file sizes involved. Moreover, increases in the fractured layout data volume[2] lead to disproportionate, superlinear increases in mask writing and inspection time. Compounding these woes is the fact that the total cost to produce low-volume parts is now dominated by mask costs [5] since masks costs cannot be amortized over a large number of shipped products. There is a clear need to reduce the negative implications of OPC on total design cost while maintaining the printability improvements provided by this crucial RET step.

Design Function in the Design-Manufacturing Interface.
A primary failing of current approaches to the design-manufacturing interface is in lack of communication across disciplines and/or tool sets. For example, it is well documented that mask writers do not differentiate among shapes being patterned - given this, gates in critical paths are given the same priority as pieces of a company logo and errors in either of these shapes will cause mask inspection tools to reject a mask. In this light, we observe that OPC has tra-

[1]Compared to traditional raster scanning, vector scanning allows features to be scaled up or down in size while maintaining sharpness, but the write cost is proportional to feature complexity: the mask pattern must be decomposed into a set of disjoint "shots" or "flashes", each of which takes roughly constant (unit) time.

[2]E.g., according to the 2003 ITRS [2], the maximum single-layer MEBES file size increases from 216GB in 90nm to 729GB in 65nm.

Year	2004	2007	2010
Technology Node	90nm	65nm	45nm
MPU gate length	37nm	25nm	18nm
MPU Gate CD 3σ	3.3nm	2.2nm	1.6nm
ASIC gate length	53nm	32nm	22nm
ASIC CD 3σ	4.7nm	2.9nm	2.0nm

Table 1: The ITRS requirement of gate dimension variation control is becoming more stringent as the technology scales.

ditionally been treated as a purely geometric exercise wherein the OPC insertion tool tries to match every edge as best as it can. As we show in our work, and has been observed in [1], such "over-correction" leads to higher mask costs and larger runtimes. A first approach to driving RET explicitly by performance considerations was proposed at DAC-2003 by Gupta et al. [1]. Their work proposes selective OPC based on an assumption of several available levels of correction. By a clever mapping of assumed statistical timing libraries and OPC costs to the classical gate sizing problem, a commercial synthesis tool was coerced into functioning as a sizer to determine minimum levels of correction for each cell instance under timing and parametric yield constraints. However, we note that OPC causes a pattern dependent variation and in this case modeling variation by Gaussian random variables, as in [1], is inaccurate. Further, the work of [1] does not quantify levels of correction and hence is not validated (in fact, cannot be validated) within any current industrial design-MDP flow.

A Performance-Driven OPC Methodology. In this work, we propose a performance-driven OPC methodology that is demonstrated to be highly implementable within the limitations of current industrial design flows. Contributions of our work include the following.

- *Quantified CD error tolerance.* We propose a mathematical programming based budgeting algorithm that outputs edge placement error tolerances (in nm) for layout features.
- *Integration within a commercial MDP flow.* We describe a practical flow implementable with commercial tools and validate the minimum cost of correction methodology.
- *Reduction of OPC overhead.* We measure OPC overhead in terms of additional MEBES features as well as runtime of the OPC insertion tool and show substantial improvements in both.

Our paper is organized as follows. In Section 2, we review the "MinCorr" (minimum cost of correction) methodology [1], and discuss roadblocks to its adoption within the context of the modern design methodology. Section 3 proposes a practical approach to the cost of correction problem that overcomes difficulties in the original MinCorr methodology. Section 4 then describes several key implementation details, our experimental setup, and experimental results. Section 5 concludes with directions for future work.

2. Review of the MinCorr Methodology

In this section we review the *MinCorr* methodology presented in [1]. Recall that current OPC techniques are unaware of design intent, so that the entire layout is corrected uniformly with the same effort. On the other hand, features in the layout which are not timing-critical might reasonably be expected to tolerate a larger degree of variation. This leads to the key idea behind MinCorr: if various *levels of correction* are available to trade off OPC cost for L_{eff} uncertainty, then selected gates can receive less aggressive OPC provided that the resulting increase in their timing

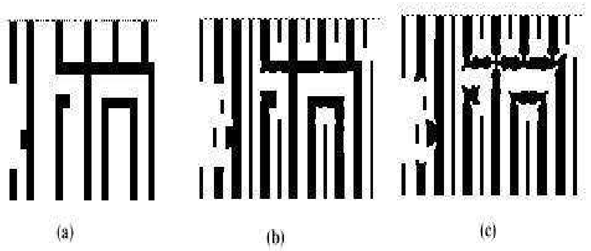


Figure 2: Example from [1] showing an assumed three levels of OPC: (a) No OPC; (b) Medium OPC; (c) Aggressive OPC.

uncertainties does not harm the overall circuit performance. More formally, the *Minimum Cost of Correction (MinCorr)* problem is: Given a range of allowable corrections for each feature in the layout as well as the cost and CD deviation associated with each level of correction, find the level of correction for each feature such that prescribed circuit performance is attained with minimum total correction cost.

The authors of [1] show equivalence between the MinCorr problem and the traditional gate-sizing problem, enabling the use of off-the-shelf synthesis tools to solve the MinCorr problem. The key analogy - and assumption - is that there are discrete allowed "sizes" in the MinCorr problem that correspond to allowed levels of OPC aggressiveness (see Figure 2). Furthermore, for each instance in the design there is a cost and delay penalty associated with every level of correction. The mapping between traditional gate-sizing and the MinCorr problems is reproduced in Table 2. Using a flow that involves construction of yield-aware libraries for each level of correction, and a commercial synthesis tool which then "resizes" the design for minimum cost of correction, [1] reports up to a 70% reduction in figure count.

Our investigations have identified several key areas for improvement over the approach of [1], as follows.

- OPC corrects the layout for pattern-dependent through-pitch CD variation. Such variations are predictable, for example, by lithography simulations. Treating these variations as random - as is done in [1] - is inaccurate.
- No distinction is made between field-poly and gate-poly features in the MinCorr approach. Field poly features do not impact performance and hence any delay-constrained MinCorr approach should not change the correction of field-poly. Moreover, quality metrics of field-poly are different from those of gate-poly (e.g., contact coverage [3]). Therefore, the cost savings given in [1] may represent substantial over-estimates.
- Commercial OPC tools are driven by *edge placement errors* (EPEs), rather than critical dimensions (CDs). Thus, the levels of correction assumed in [1] need to be quantified so that they can be enforced using existing OPC insertion tools.
- The figure count numbers (which are proxies for mask cost implications) for the cells are projected numbers, and are not obtained by model-based OPC. Indeed, the entire MinCorr flow as presented in [1] has not, to our knowledge, been validated with any actual OPC insertion tool.

3. A Practical Methodology: EPEMinCorr

We now describe EPEMinCorr, a practical implementation of the MinCorr methodology that accounts for the shortcomings in [1] noted above. We can summarize the key contribution of EPEMinCorr as: *We devise a flow to pass design constraints on to the OPC insertion tool in a form that it can understand.*

Gate Sizing		MinCorr
Area	$\equiv$	Cost of Correction
Nominal delay	$\equiv$	Delay $\mu + k\sigma$
Cycle Time	$\equiv$	Selling point delay
Die Area	$\equiv$	Total Cost of OPC

Table 2: Correspondence established in [1] between the traditional gate sizing problem and the minimum cost of correction (to achieve a prescribed selling point delay with given yield) problem.

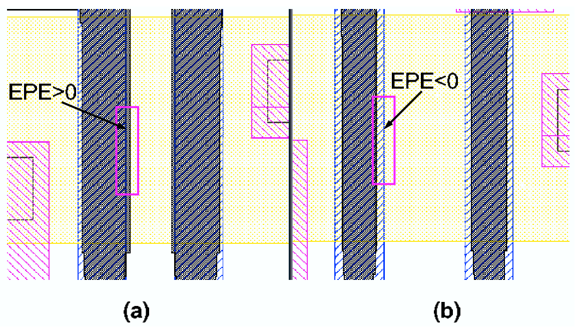


Figure 3: The signed edge placement error (EPE).

As previously mentioned, OPC insertion tools are driven by *edge placement error* (EPE) *tolerances*. Typical model-based OPC techniques break up edges into *edge-fragments* that are then iteratively shifted outward or inward (with respect to the feature boundary) based on simulation results, until the estimated wafer image of each edge-fragment falls within the specified EPE tolerance. EPE (and hence EPE tolerance) is typically signed, with negative EPE corresponding to a decrease in CD (i.e., moving the edge inward with respect to the feature boundary). An example of a layout fragment and its EPE is shown in Figure 3. Mask data volume is heavily dependent on the assigned EPE tolerance that the OPC insertion tool is asked to achieve. For example, Figure 4 shows the change in MEBES file size for cell with applied OPC as the EPE tolerance is varied. In this particular example, loosened EPE tolerances can reduce data volume by roughly 20% relative to tight control levels.

Since model-based OPC corrects for pattern-dependent CD variation, which is systematic and predictable, we assert that OPC actually determines *nominal timing*, rather than parametric yield as assumed in the work of [1]. This allows us to base our OPC insertion methodology on traditional corner-case timing analysis tools instead of (currently non-existent from a commercial standpoint) statistical timing analysis tools. Our methodology adopts a slack budgeting based approach - as opposed to a sizing based approach as in [1] - to determine EPE tolerance values for every feature in the design. For simplicity, our description and experiments reported here are restricted in two ways: (1) we apply selective EPE tolerances in OPC to only gate-poly features, and (2) every gate feature in a given cell instance is assumed to have the same EPE tolerance (the approach may be made more fine-grained using the same techniques that we describe). Figure 5 shows our EPEMinCorr flow. The quality of results generated by the flow are measured as MEBES data volume of fractured post-OPC insertion layout shapes as well as OPC insertion tool runtime, which can be prohibitive when run at the full-chip level. In the remainder of this section, we describe details of the major steps of the Figure 5 EPEMinCorr flow.

3.1 Slack Budgeting

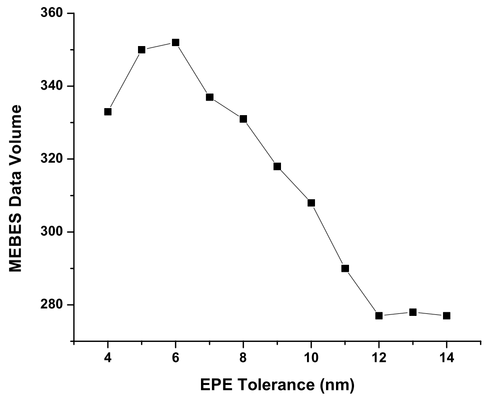


Figure 4: Mask data volume (kB) vs EPE tolerance for a NAND3X4 cell in TSMC 130nm technology.

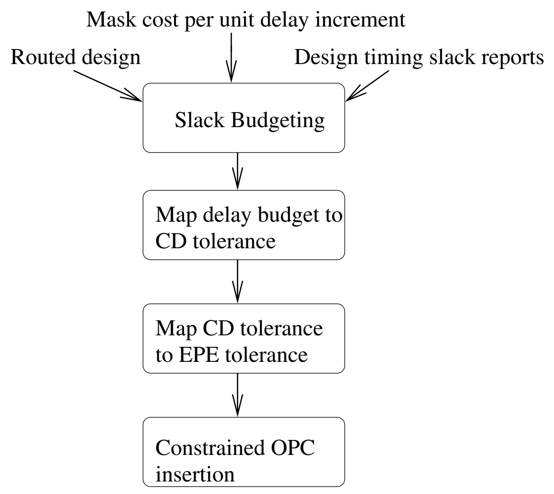

Mask cost per unit delay increment
Routed design
Design timing slack reports
Slack Budgeting
Map delay budget to CD tolerance
Map CD tolerance to EPE tolerance
Constrained OPC insertion

Figure 5: The EPEMinCorr flow to find quantified edge placement error tolerances for layout features and drive OPC with them.

The slack budgeting problem seeks to distribute slack at the primary inputs of combinational logic (i.e., sequential cell outputs) to various nodes in the design. One of the earliest and simplest approaches, the zero-slack algorithm (ZSA) [7], iteratively finds the minimum-slack timing path and distributes its slack equally among the nodes in the path. The MISA algorithm for slack budgeting proposed in [8] distributes slack iteratively to an independent set of nodes. As with ZSA, the objective is to maximize the total added incremental delay budget on timing arcs. A weighted version of MISA is also proposed in [8].

We observe:

- Neither MISA variant is guaranteed to provide optimal solutions.
- ZSA is much faster than MISA, and a weighted version of ZSA can also be formulated.
- While [9] formulates the budgeting problem as a convex programming problem, full-chip MISA or mathematical programming is, as far as we can determine, too CPU-intensive for inclusion in a practical flow.

We propose to approximate full-chip mathematical programming by iteratively solving a sequence of linear programs (LPs). In each iteration, slack is budgeted among the top k available paths. Once a budget is obtained for a node, this budget is retained as an upper bound for subsequent iterations. The process is repeated until all nodes have been assigned a slack budget or path slack is sufficiently large. The basic LP has the following form:

$$\text{Maximize } \sum_{i=1}^{n} C_i s_i \tag{1}$$

$$\sum_{j \in P_k} s_j \leq S_k \; \forall \, k \in \text{Current path list}$$

$$s_j \leq s_j^f \; \forall \, j \in F$$

where C_i denotes the correction cost decrease per unit delay increase for cell i, and s_i is the slack allocated to cell i. The notation P_k is used to denote the k^{th} most critical path, and S_k is the slack of this path. Finally, F denotes the set of nodes with slacks fixed from previous iterations. An example sequence of LPs might be obtained by allowing k to take on the range from 1 to 100 in the first iteration, 101 to 200 in the second iteration, and so on.

We observe that when a budgeting formulation is adopted in place of a sizing formulation, the method of accounting for changes in next-stage input pin capacitance becomes an open question. To be conservative, we generate timing reports with pin input capacitances that correspond to the loosest tolerance (i.e., largest pin capacitance) but gate delays corresponding to the tightest achievable tolerance. C_i is obtained via a pre-built look-up table (similar to .lib format) containing the increase in data volume, mapped against delay change.

Our budgeting procedure yields positive delay budgets leading to positive EPE tolerances. Since EPE tolerance is a signed quantity (e.g., in Mentor Calibre, a common OPC insertion tool), negative EPE tolerances (corresponding to reduced gate length and faster delay) can also be obtained in a similar way based on hold-time or leakage power constraints. However, in this paper we assume equal positive and negative EPE tolerances since we deal with purely combinational benchmarks and focus on timing rather than power.

3.2 Calculation of CD Tolerances

To map delay budgets found from the above linear programming based formulation to CD tolerances, we require characterization of a standard-cell library with varying gate lengths. Using such an augmented library, along with input slew and load capacitance values for every cell instance, we can map delay budgets to the corresponding gate lengths. For example, if a particular instance with specified load and input slew rate has a delay budget of 100ps, then we can select the longest gate length implementation of this gate type that meets this delay. This largest allowable CD will lead to a more easily manufactured gate with less RET effort. Subtracting these budgeted gate lengths from nominal gate lengths yields the CD tolerance for every cell in the design.

3.3 Calculation of EPE Tolerances

The next step in our flow maps CD tolerances to signed EPE tolerances. Again, obtaining EPE tolerances is crucial since this is the parameter which OPC insertion tools understand and can exploit. As noted above, in this work we assume positive and negative EPE tolerance to be the same. Since CD is determined by two edges, the worst-case CD tolerance is twice the EPE tolerance.

In most lithography processes, gates shrink along their entire width such that the printed gate length is always smaller than the drawn gate length, except at the corners of the critical gate feature. OPC typically biases the gate length such that corrected gate length is *larger* than the designer-drawn gate length. Thus, model-based OPC shifts edges *outward*, i.e., in the "positive" direction, until it meets the EPE tolerance specification. If the step size of each edge move is small enough, the EPE along the gate width will always be negative (since we are approaching the larger nominal gate length value starting from the smaller printed gate length value). As a result, actual printed gate length will almost always be smaller than the drawn gate length, leading to leakier but faster devices.

Test Case	Source	Cell Count
c432	ISCAS85	337
c5315	ISCAS85	2093
c6288	ISCAS85	4523
c7552	ISCAS85	2775
alu128	Opencores	12403
r4_sova	Industry	34288

Table 3: Benchmark details.

To achieve a more unbiased deviation from nominal, we exploit the behavior of the OPC tool by applying simple pre-biasing of gate features in an attempt to achieve EPE tolerances that are equal to CD tolerance. Specifically, we pre-bias each gate feature by its intended EPE tolerance. For instance, for a drawn gate length of 130nm and EPE tolerance of 10nm, the printed CD would typically lie between 110nm and 130nm (each edge shifts by 10nm inward). If the gate length is biased by 10nm so that the OPC tool views 140nm as the target CD, the printed CD would lie between 120nm and 140nm, which amounts to a $\pm$10nm CD tolerance. In this way, pre-biasing achieves CD tolerances equal to the EPE tolerance. An example of the average CD for a specific gate-poly with and without pre-biasing is shown in Figure 6. It is clear that pre-biasing achieves its goal of attaining average CDs that are very close to the target CD (130nm in our case). Another point illustrated in Figure 6 is that the variation in CD (measured as the standard deviation of CD taken across all edge-fragments) grows as the EPE tolerance is relaxed. This is shown more clearly in Section 4.

3.4 Constrained OPC

We enforce the obtained EPE tolerances within a commercial OPC insertion flow. We use *Calibre* [10] as the OPC insertion tool; details of constraining the tool are described in the next section.

4. Experimental Setup and Results

In this section we describe our experiments and the results obtained in order to validate the EPEMinCorr methodology.

4.1 Test Cases

We use several combinational benchmarks drawn from ISCAS85 suite of benchmarks, Opencores [11] and industrial signal processing benchmarks[3]. These benchmark circuits are synthesized, placed and routed in a restricted TSMC 0.13 μm library containing a total of 32 cell macros with cell types of BUF, INV, NAND2, NAND3, NAND4, NOR2, NOR3, and NOR4. The test case characteristics are given in Table 3.

4.2 Library Characterization

We assume a total of EPE tolerance levels ranging from $\pm$4nm to $\pm$14nm. Corresponding to each EPE tolerance, the worst case gate length is $130nm + EPE\_Tolerance$. We map cell delays to EPE tolerance levels by creating multiple .lib files for each of the 10 worst case gate lengths using circuit simulation. For simplicity, we neglect the dependence of delay on input slew in our analysis but this could easily be added to the framework.

Expected mask cost for each cell type is extracted as a function of EPE tolerance. We run model-based OPC using Calibre on indi-

[3]Thank Prof. B. Nikolic with the Dept. of EECS, University of California at Berkeley, for providing the benchmark.

vidual cells followed by fracturing to obtain MEBES data volume numbers for each (cell, tolerance) pair. Though the exact corrections applied to a cell will depend somewhat on its placement environment, standalone OPC is fairly representative of data volume changes with changing EPE tolerance. Finally, we calculate the sensitivity of mask cost to delay change under the assumption that cost reduction is a linear function of delay increase. This assumption is based on linearity between gate delay and CD as well as the rough linearity shown in Figure 4 between data volume and EPE tolerance. We then build a .lib-like look-up table of correction cost sensitivities (with respect to the tightest EPE tolerance of 4nm). When slack is distributed to various nodes, we extract the load capacitances that are used to identify entries in the sensitivity table. Cost change is most sensitive to delay changes when the load capacitance is small (this typically indicates a small driver and subsequently small amount of data volume) and the sensitivity numbers are on the order or 10^2 to 10^4 MEBES features per ns delay change.

4.3 EPEMinCorr with Calibre

Our OPC flow involves assist-feature insertion followed by model-based OPC. The EPE tolerance is assigned to each gate by the *tagging* command within Calibre. We first separate the entire poly layer into gate poly and field poly components. The field-poly tolerance is taken to be $\pm$14nm while gate-poly tolerance ranges from $\pm$4nm to $\pm$14nm. We tag the assigned EPE tolerance to cell names. In this way, we can track the EPE tolerance of each gate individually. We take 1nm as our step size[4] when applying OPC to obtain very precise correction levels. We set the iteration number to the minimum value beyond which adding mask cost and CD distribution show little sensitivity to OPCs, which is found experimentally. After model-based OPC is applied, we perform 'printimage' simulations in Calibre to obtain the expected as-printed wafer image of the layout. Average gate CD and its standard deviation are extracted from this wafer image. The corrected GDSII is fractured into MEBES using CalibreMDP. The total mask data volume is then determined based on the MEBES file sizes.

4.4 Results

We synthesize the benchmark circuits using *Synopsys Design Compiler*. Place and route is performed using *Cadence Silicon Ensemble*. *Synopsys Primetime* is used to output the slack report of the top 500 critical paths (not true for the biggest benchmark r4_sova where more paths are needed as discussed below) as well as the load capacitance for each driving pin. As noted above, STA is run with a modified 134nm (tightest EPE tolerance) library with pin capacitances corresponding to 144nm (loosest EPE tolerance) to remain conservative after slack budgeting. We use *Cplex v8.1* [12] as the mathematical programming solver to solve the budgeting linear program. Two types of benchmarks are involved in our experiments: (i) large designs with a "wall" of critical paths, e.g., r4_sova in Table 3; and (ii) circuits with fairly small sizes, e.g., benchmarks except r4_sova. For (ii), a single iteration is efficient to solve the budgeting problem; for (i) however, more iterations may be necessary because some paths which are potentially critical but are not reported due to the constraint of maximum number of critical paths may become top critical later on as they are not treated as optimization objects by the slack budgeting algorithm, resulting in performance degradation. One possible solution to this problem is to perform iterations to selectively include those paths that may cause performance degradation, as slack budgeting objects. Another simple but not as efficient option is to increase the constraint of maximum number of critical paths in the slack report. We deploy a hybrid way for r4_sova in our case, i.e., the constraint on the initial number of critical paths is increased from 500 to 10000, then in each iteration 5000 more paths that are potentially critical are included for slack budgeting. After 8 iterations the performance degradation due to the selective OPC is reduced to less than 1% (first iteration gives 4.3% performance degradation).

[4]Step size is the minimum perturbation to an edge that model-based OPC can make. Smaller step sizes lead to better correction accuracy at the cost of runtime.

The extracted CD variation for test case $c432$ after EPEMinCorr OPC is shown in Figure 7. The distributions show that Calibre is able to enforce assigned tolerances very consistently. A tighter CD distribution for critical gates is achieved while non-critical gates (which can tolerate a larger deviation from nominal) have a more relaxed (and hence less expensive to implement) gate length distribution. Table 4 compares the runtime and data volume results for EPEMinCorr OPC and traditional OPC. For relatively small circuits, a single iteration of the budgeting approach ensures that there is no timing degradation going from the traditional to the EPEMinCorr flow, and the budgeting runtimes are negligibly small ranging from 1s to 11s. For large designs especially those with a "wall" of critical paths, iterations may be required to avoid performance degradation and the sum of budgeting runtimes of each iteration may reach several hours (7 hours for r4_sova). The important result is the amount of mask cost reductions achieved whether measures as runtime of model-based OPC or fractured MEBES data volume. EPEMinCorr flow reduces MEBES data volume by 17%-24%. Such reductions directly translate to substantial mask-write time improvements. OPC runtimes are improved by 6%-41%. These percentage numbers translate to a huge absolute TAT savings. For instance, the EPEMinCorr flow saves 16.4 hours compared to the traditional OPC flow on a 34000 gate benchmark.

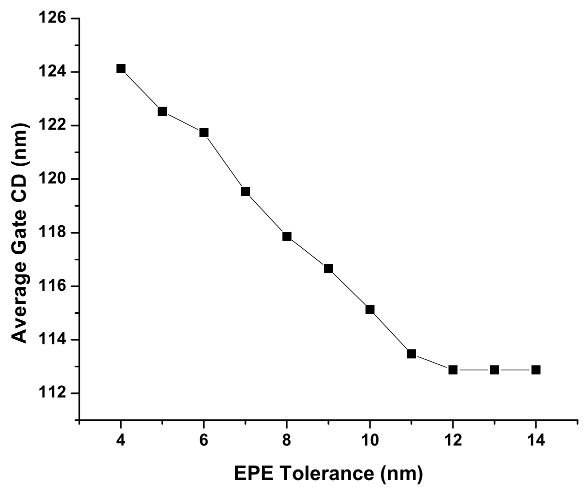
5. Conclusions and Future Work

This work aims to propose and implement a practical means of reducing masks costs and the computational complexity of OPC insertion through formalized performance-driven OPC assignment. In particular we focus on the use of edge placement errors to drive OPC insertion tools and leverage EPEs as the mechanism to direct these tools to correct only to the levels required to meet timing specifications. An iterative linear programming based approach is used to perform slack budgeting in an efficient manner. This formulation results in a specific slack budget for each gate which is then mapped to allowable critical dimensions in the standard cell. Finally EPEs are generated from the CD budget and tags are placed on gates to indicate to the OPC insertion tool the appropriate level of correction. Our results on several benchmarks ranging from 300 to 34000 cells show up to 24% reductions in MEBES data volume which is frequently used a metric for RET complexity. Furthermore, the runtime of the OPC insertion tool is reduced by up to 41% - this is critical since running OPC tools at the full-chip level is an extremely time-consuming step during the physical verification stage of IC design.

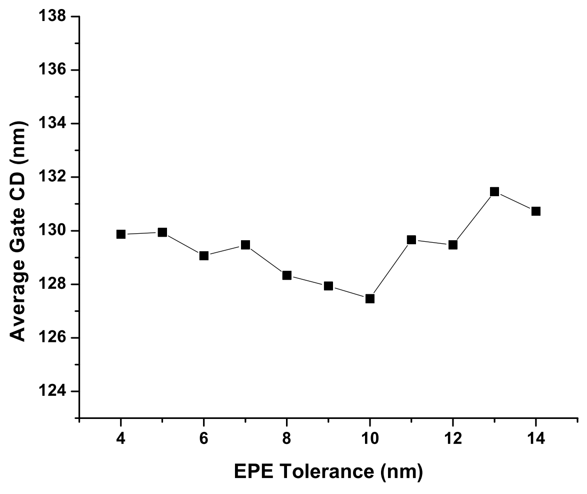
Our future work in this area will extend the above framework to consider sequential circuits (e.g., comprehending hold-time constraints) and leakage power constraints. This latter point is significant given that the spread observed in static power consumption in modern microprocessors can exceed an order of magnitude due to the exponential dependency of leakage on channel length [13]. In future technologies allowable CD tolerances may be set more by bounds on acceptable leakage power than by traditional delay uncertainty constraints. We also plan to extend the EPEMinCorr methodology to field-poly features.

Testcase	Traditional OPC Flow				EPEMinCorr Flow							
	CD Distribution		OPC	Delay	Budgeting	CD Dis tribution				OPC	Delay	Normalized
	All Gates (nm)		Runtime	(ns)	Runtime	All Gates (nm)		Critical Gates (nm)		Runtime	(ns)	MEBES
	mean	σ	(s)		(s)	mean	σ	mean	σ	(s)		Volume
r4_sova	126.3	2.07	142989	8.19	29648	131.9	5.00	130.0	1.75	83864	8.26	0.79
alu128	126.1	1.48	51516	3.28	11	131.5	4.93	130.8	2.04	33535	3.28	0.76
c7552	126.2	1.89	7149	1.59	4	132.0	4.77	130.1	1.99	5142	1.59	0.78
c6288	126.0	1.37	12830	5.21	9	131.4	4.45	129.7	1.27	9710	5.21	0.82
c5315	126.1	1.82	4539	1.94	3	131.7	4.70	129.7	1.89	4247	1.94	0.79
c432	126.8	1.57	1020	1.33	1	131.3	3.90	129.9	1.67	737	1.33	0.83

Table 4: Impact of EPEMinCorr optimization on Cost and CD. All runtimes are based on a 2.4GHz Xeon machine with 2GB memory running Linux.



(a) Unbiased OPC



(b) Pre-biased OPC

Figure 6: Comparison of average printed gate CD with and without pre-bias for the cell macro NAND3X4.

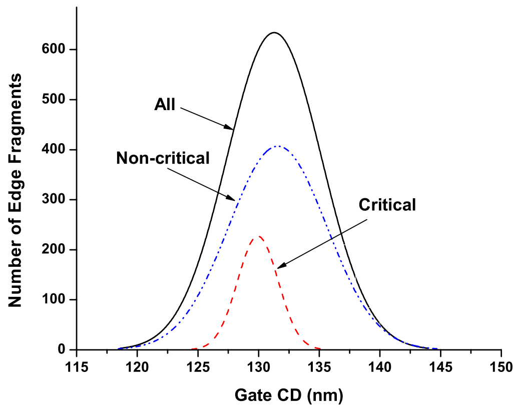


Figure 7: Gate CD distribution for c432. Gates with budgeted 4nm EPE tolerance are labeled critical gates while others are labeled as non-critical. The y-axis shows the number of fragments of gate edges with a given printed CD.

6. REFERENCES

[1] P. Gupta, A.B. Kahng, D. Sylvester and J. Yang, "A Cost-Driven Lithographic Correction Methodology Based on Off-the-Shelf Sizing Tools", *Proc. IEEE/ACM DAC*, June 2003, pp. 16-21.

[2] International Technology Roadmap for Semiconductors, 2003. http://public.itrs.net/

[3] P. Gupta, F.-L. Heng and M. Lavin, "Merits of Cellwise Model-Based OPC", *Proc. SPIE International Symposium on Microlithography*, 2004, pp. 182-189.

[4] S. Murphy, Dupont Photomask, *SEMATECH: Mask Supply Workshop*, 2001.

[5] M.L. Rieger, J.P. Mayhew and S. Panchapakesan, "Layout Design Methodologies for Sub-Wavelength Manufacturing", *Proceedings of Design Automation Conference*, 2001, pp. 85-92.

[6] *Optical Lithography Cost of Ownership - Final Report* , http://www.sematech.org/docubse/document/4014atr.pdf

[7] R. Nair, C.L. Berman, P.S. Hauge and E.J. Yoffa, "Generation of Performance Constraints for Layout", *IEEE Transactions on Computer Aided Design*, 8(8), 1989, pp. 860-874.

[8] C. Chen, E. Bozorgzadeh, A. Srivastava and M. Sarrafzadeh, "Budget Management with Applications", *Algorithmica*, 2002, pp. 261-275.

[9] E. Bozorgzadeh, S. Ghiasi, A. Takahashi and M. Sarrafzadeh, "Optimal Integer Delay Budgeting on Directed Acyclic Graphs", *Proc. IEEE/ACM DAC*, 2003, pp. 920-925.

[10] http://www.mentor.com

[11] http://www.opencores.org

[12] http://www.ilog.com

[13] P. Gupta, A.B. Kahng, P. Sharma and D. Sylvester, "Selective GateLength Biasing for Cost-Effective Runtime Leakage Control", *Proc. IEEE/ACM DAC*, June 2004, pp. 327-330.