ITOP: Integrating Timing Optimization within Placement [*]

Natarajan Viswanathan[†], Gi-Joon Nam[†], Jarrod A. Roy[†], Zhuo Li[†],
Charles J. Alpert[†], Shyam Ramji[‡], Chris Chu[§]

[†] IBM Austin Research Laboratory, 11501 Burnet Road, Austin, TX 78758
[‡] IBM Systems & Technology Group, Hopewell Junction, NY 12533
{ nviswan, gnam, jaroy, lizhuo, alpert, ramji }@us.ibm.com
[§] Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011
cnchu@iastate.edu

ABSTRACT

Timing-driven placement is a critical step in nanometer-scale physical synthesis. To improve design timing on a global scale, net-weight based global timing-driven placement is a commonly used technique. This paper shows that such an approach can improve timing, but often degrades wire length and routability. Another problem with existing timing-driven placers is inconsistencies in the definition of timing closure. Approaches using linear programming are forced to make assumptions about the timing models that simplify the problem. To truly do timing-driven placement, the placer must be able to make queries to a real timing analyzer with incremental capabilities. This paper describes an incremental timing-driven placer called ITOP. Using accurate timing from an industrial static timer, ITOP integrates incremental timing closure optimizations like buffering and repowering within placement to improve design timing without degrading wire length and routability.

Experimental results on a set of optimized industrial circuit netlists show that ITOP significantly outperforms conventional net-weight based timing-driven placement. In particular, on average, it obtains an improvement of over 47.45%, 9.88% and 5% in the worst slack, total negative slack and wire length as compared to the conventional flow.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuits, Design Aids**]: Placement and routing

General Terms

Algorithms, Design

Keywords

Physical Synthesis, Placement, Timing Optimization

1. INTRODUCTION

Timing closure is one of the primary objectives of a physical synthesis tool. In this respect, timing-driven placement is a critical step during physical synthesis, as its quality significantly impacts the ability of the physical synthesis tool or designer to achieve timing closure. Existing timing-driven placement techniques can be broadly classified into two categories: (a) global techniques [5–8, 10, 11, 16–18, 23], and (b) incremental techniques [2–4, 12–14, 22].

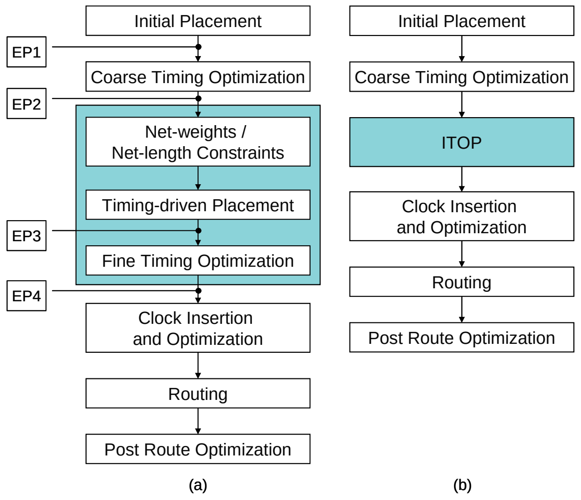


Figure 1: Physical synthesis flow: (a) using global timing-driven placement, (b) incorporating ITOP.

1.1 Global Timing-driven Placement

Global techniques place the entire circuit netlist, ignoring any previously obtained module locations. They typically use a net-based approach to minimize the wire length of the nets on the critical paths ("critical nets"). The rationale being that optimizing critical net lengths would implicitly minimize critical path lengths, leading to better critical path delay. Net-based techniques transform the net criticalities into "net specifications" using net-weights [5,10,11,17,18,23], or net-length constraints [6, 8, 16].

To generate net specifications, they follow a physical synthesis flow similar to Figure 1(a). First, a wire length driven global placement is performed to obtain module locations. Then, coarse timing optimization is performed to bring the design to a reasonable electrical and timing state. Based on a timing analysis, net specifications are then generated to reflect the timing criticality of the nets. These specifications guide the subsequent timing-driven placement. The key drawbacks of global timing-driven placers are:

[*]This work was partially supported by IBM Faculty Award and NSF under grant CCF-0540998.

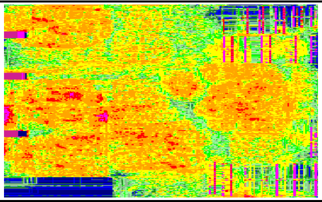
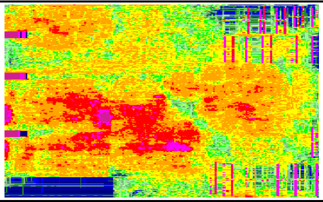
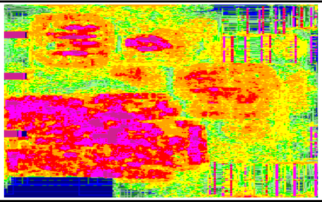
Stage	EP1	EP2	EP3
Congestion			
Wire Length	140.29	146.94	160.10

Figure 2: Wire length ($\times e6$) and routing congestion during the physical synthesis flow of Figure 1(a). EP1: After initial placement, EP2: After coarse timing optimization, EP3: After timing-driven placement. (Regions colored pink and purple have more than 100% routing resource usage - indicating unroutable regions.)

- Timing-driven placement minimizes critical net wire length at the expense of the non-critical nets. Inferior net specifications can result in a substantial increase in the wire length during global placement. This can also cause severe routing congestion. This is depicted in Figure 2, which shows a significant increase in the wire length and congestion after timing-driven placement.
- They do not interact with timing optimization transforms like buffering, gate sizing, etc., during global placement. Purely minimizing weighted wire length generally degrades timing after placement. This is shown in Table 1, where timing-driven placement (EP3) significantly degrades the timing obtained after coarse timing optimization (EP2). This needs to be recovered by fine timing optimization (EP4). In addition, a close interaction with optimization can potentially buffer a net or resize a gate instead of the corresponding net being over-optimized during placement. This can lead to significant savings in wire length.
- Net specifications are just an indirect measure of the actual timing constraints. It is difficult to come up with a good set of net specifications to effectively optimize design timing during placement.
- Typically, once generated, net specifications are kept constant for the duration of global placement. This introduces an additional level of inaccuracy, as placement changes will invalidate the timing upon which the specifications were initially generated.
- Dynamic updation of net specifications during placement have been proposed (e.g., [5,6,18]). But, to maintain placement efficiency, they are updated using inaccurate timing models and illegal module locations. This can also cause oscillations during placement resulting in issues with convergence.

Stage	Worst Slack (ns)	Num. Negative Paths
EP1	-1606.35	73557
EP2	-0.89	7219
EP3	-275.91	63743
EP4	-0.46	1451

Table 1: Design timing at various stages in the physical synthesis flow shown in Figure 1(a).

1.2 Incremental Timing-driven Placement

Alternately, incremental techniques place a subset of the circuit netlist. They mostly use a path-based approach, wherein they try to directly optimize the timing critical paths in the design. Although many flavors exist, a majority of them use the approach of linear programming [9]. The key drawbacks of incremental timing-driven placers are:

- To perform module movement, they rely on inaccurate or crude models for gate delay, interconnect delay, etc., that do not capture the complex timing environment of nanometer-scale design.
- They typically rely on computationally intensive mathematical programming techniques. As a result, they cannot be used during the early stages of physical synthesis, where a large number of paths need to be simultaneously optimized during placement.
- As with global placers, they generally do not interact with timing optimization during placement. Although techniques to incorporate optimization within placement have been proposed (e.g., [2]), they still rely on inaccurate and simple delay models.
- They ignore module overlap while solving the mathematical program, and rely on a subsequent legalization step to enforce placement legality. This can lead to a degradation in design timing during legalization.
- The legalization issue is particularly magnified in modern designs with numerous placement blockages. Incremental placers do not explicitly model and account for placement blockages during critical path optimization.
- They also do not honor placement density constraints, required to provide space for optimization and routing.

1.3 This Work

This work is motivated by the following observations: (a) a robust, high-quality timing closure flow requires close coupling between placement and timing optimization, and (b) both steps should rely on accurate timing from a static timing analyzer. It describes an incremental timing-driven placer called ITOP, that effectively integrates timing optimization and static timing analysis into placement. ITOP has been developed to improve design timing during the early stages of physical synthesis (i.e., have a global impact on design timing). Hence, it is embedded within a physical synthesis flow as shown in Figure 1(b).

The key features of ITOP are:

- A simple, yet effective netlist transformation technique to model the critical paths in the design, for them to be effectively optimized during placement.
- An efficient incremental path smoothing algorithm to simultaneously optimize a large number of critical paths.
- Explicit modeling and handling of placement blockages during critical path optimization.
- A tight integration of incremental timing optimization and static timing analysis within placement.

- An iterative flow incorporating periodic congestion mitigation, wire length recovery and global timing optimization, to improve design timing without degrading wire length and routability.

The rest of this paper is organized as follows: Section 2 provides an overview of ITOP. Sections 3–5 describe the individual components of the algorithm. Section 6 gives the detailed algorithm for ITOP. Experimental results on industrial 65nm and 45nm designs are reported in Section 7. The paper concludes with some observations in Section 8.

2. OVERVIEW OF ITOP

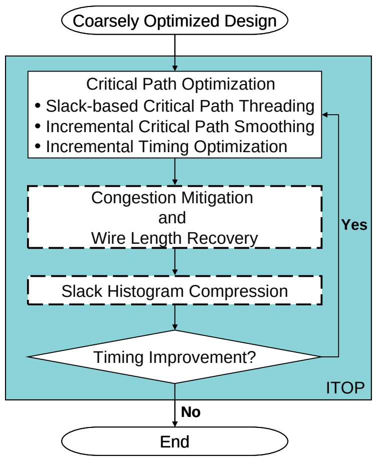

Coarsely Optimized Design
Critical Path Optimization
• Slack-based Critical Path Threading
• Incremental Critical Path Smoothing
• Incremental Timing Optimization
Congestion Mitigation and Wire Length Recovery
Slack Histogram Compression
Timing Improvement?
Yes
No
ITOP
End

Figure 3: High-level flow for ITOP.

Figure 3 gives the high-level flow for ITOP. The input to ITOP is a placed and coarsely optimized design. ITOP then uses an iterative approach to improve design timing in an incremental manner. The key stages during ITOP are:

1. Critical Path Optimization: This stage performs placement and timing optimization on the timing critical paths in the design. It comprises of three steps: (a) Slack-based Critical Path Threading – to identify, and model the critical paths, (b) Critical Path Smoothing – to re-place the modules on the critical paths, and (c) Incremental Timing Optimization – to perform quick optimization (e.g., buffering and gate sizing) (Sec 3).
2. Congestion Mitigation and Wire Length Recovery: Periodically, the non-critical modules are moved to improve placement density. In addition, detailed placement transforms are run to recover wire length (Sec 4).
3. Slack Histogram Compression: In addition, a periodic global timing optimization is performed to compress the entire slack histogram (Sec 5).

3. CRITICAL PATH OPTIMIZATION

This stage performs incremental path smoothing and timing optimization on the timing critical paths in the design. This section describes the associated steps.

3.1 Slack-based Critical Path Threading

To meet timing requirements, designers often set an overall *slack_threshold* for the design (usually a small positive value). In practice, all paths that have a slack below the *slack_threshold* ("negative slack") are considered critical. As

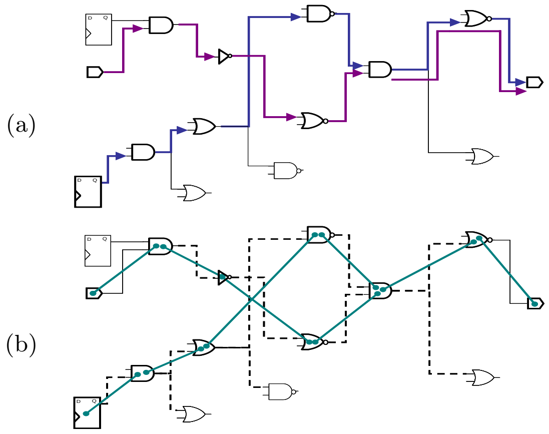


Figure 4: (a) A sample netlist (bold arrows represent the critical paths), (b) Critical Path Threading (solid lines represent path-threading attractions and dotted lines represent weighted output nets).

ITOP uses an iterative flow, only a subset of the negative paths, having the most negative slack values, are considered to be critical during each iteration. These paths are identified using the timing report from an industrial static timer.

For each critical path, the modules on the path are linked or "threaded" using additional two-pin nets that are assigned a high net-weight. The high net-weight aids the placement step to effectively minimize the two-pin net lengths. For each path, minimizing the two-pin net lengths minimizes the total path length. Within ITOP, these two-pin nets are termed as "path-threading attractions" and their weight is set to 10$\times$ the default net-weight for a non-critical net. In addition, multiple attractions between any two modules are coalesced and only a single attraction is added between them. Apart from the path-threading attractions, the output net for each movable module on the critical path is also assigned a weight of 5$\times$ the default net-weight. This aids in the timing convergence of the overall flow by also moving the modules that are adjacent to the critical path in its fanout cone.

For example, Figure 4(a) shows a sample netlist with two critical paths. Figure 4(b) shows the path-threading attractions (solid lines linking the modules on the critical paths), and the weighted output nets (shown by the dotted lines).

To set the attraction and net weights, ITOP does not use complex schemes relying on path counting [11] or slack and delay sensitivity [12, 17, 23]. Instead, it uses the aforementioned simple technique. As the modules are not moved by a large distance during placement, this simple net-weighting scheme is adequate to improve the overall design timing. This is validated by the experimental results in Section 7.

3.2 Incremental Critical Path Smoothing

The objectives of critical path smoothing are: (a) minimize critical path lengths by straightening the paths, (b) distribute the modules evenly along their respective paths.

To facilitate the subsequent discussion, we define:

- Critical net: A net with a weight higher than the default net-weight. This includes path-threading attractions and weighted output nets (Section 3.1).
- Critical module: A module that belongs to a critical net. This includes latches/flip-flops, which are typically the end-points of the path(s).

During this step, the locations of the non-critical modules remain unchanged. It is only the critical modules that are moved to optimize the critical paths. To perform module movement, initially, a regular bin grid is constructed over the placement region and the source bin for all the critical modules is determined. For each critical module, eight *movement scores* are computed that correspond to tentatively moving the module to its eight neighboring bins. To calculate the score, it is assumed that a module is moving from its current location in a source bin to the same relative location in the neighboring bin. The main objectives during placement are to minimize the path lengths and evenly distribute the critical modules along the paths. Hence, the score for each move is the reduction in the *weighted quadratic* wire length of all the nets connected to the module. If all the scores are negative, the module is not moved. Otherwise, it is moved to the neighboring bin with the highest positive score. The above steps are performed in an iterative manner on all the critical modules, until there is no improvement in the total critical net wire length.

To have a tight coupling between placement and timing optimization, the movement of the critical modules is restricted to a local region. This is ensured by setting a maximum displacement constraint on the modules during each ITOP iteration. This constraint is relaxed only if the modules need to cross over blockages in the placement image. This is explained in Section 3.2.1.

Critical path smoothing is followed by a legalization step. To ensure that the critical modules are not perturbed by a large distance, legalization follows a two-step approach:

1. First, the non-critical modules are ignored and the critical modules are legalized accounting for the placement blockages. They are then fixed and transformed into placement blockages themselves.
2. Next, the non-critical modules are legalized in the presence of all the placement blockages.

3.2.1 Tunneling to Handle Placement Blockages

To the best of our knowledge, none of the existing incremental timing-driven placers explicitly model and account for placement blockages during critical path optimization. They ignore the blockages while optimizing the critical paths, and rely on a subsequent "timing-unaware" legalization step to enforce placement legality. Modern mixed-size designs contain thousands of blockages in the placement image. Ignoring them during critical path optimization can lead to significant overlaps between the blockages and the critical modules. This can in turn lead to severe degradation in the design timing during legalization.

To consider blockages, ITOP uses the concept of tunneling during critical path smoothing. To move the critical modules, the iterative placement algorithm uses a bin-based approach, and evaluates a *movement score* in a local neighborhood of bins. If a neighboring bin overlaps with a blockage, then instead of landing on top of the blockage, it is assumed that the critical module tunnels through it in the general direction of the move. As a result, the *movement score* is evaluated in the closest bin(s) adjacent to the blockage. During tunneling, the maximum displacement constraint on the "tunneled module" is ignored. This ensures that additional candidate locations can be considered for score evaluation.

Tunneling is illustrated in Figure 5 which shows a placement blockage (dark box) and four critical modules (light boxes). One of the critical modules is placed to the right of the blockage and needs to move across to optimize the path. From Figure 5, the lower-left neighboring bin of the module overlaps with the blockage. Using tunneling, the *movement score* is evaluated in the closest bins adjacent to the left and bottom boundaries of the blockage. Since this module has three neighboring bins that overlap with the blockage, the bins with the hatched lines are the ones actually used for score evaluation to move the module.

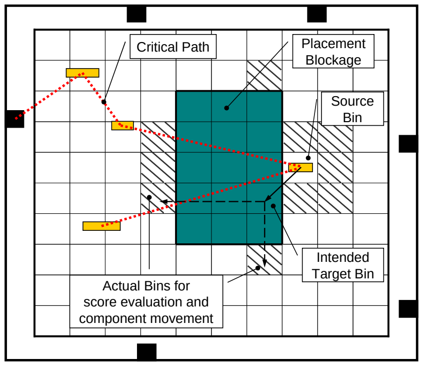


Figure 5: Tunneling through a placement blockage during critical path smoothing.

Please note, the maximum number of bins that need to be considered due to tunneling is limited to twelve[1]. This is comparable to the number of bins being considered without tunneling (eight). In addition, determining the bins adjacent to a blockage is quite efficient. Hence, tunneling has negligible impact on the runtime of the placement algorithm.

Tunneling has two key advantages:

- It ensures that there is no overlap between the critical modules and placement blockages. Hence, the modules are not significantly perturbed during legalization. This preserves the timing obtained by critical path smoothing.
- Local search techniques can terminate if the critical modules fall into the "alleys" between large blockages. Tunneling moves the modules out of these alleys and helps the placer to further optimize the critical paths.

3.3 Incremental Timing Optimization

After local movement of the modules during critical path smoothing, a timing analysis is performed, followed by incremental timing optimization to further optimize the critical paths. Buffering and gate sizing are the most common and powerful optimizations during the early stages of physical synthesis. Hence, incremental timing optimization employs these transforms to improve design timing. Please note, other optimization transforms, such as multi-threshold vt tuning, wire sizing, layer assignment, etc., can also be easily embedded within ITOP. Since the goal of ITOP is to gradually improve design timing, as before, only the top-most critical paths are considered for buffering and gate sizing during each iteration. To obtain high quality of results, state-of-the art buffering and resizing algorithms, as outlined in [1] are used. In addition, to avoid any inaccuracy and the resulting degradation in the quality of results, all the optimization transforms rely on accurate timing from an industrial static timing analyzer.

[1]In case a module is surrounded by blockages on all sides.

4. CONGESTION MITIGATION AND WIRE LENGTH RECOVERY

Since critical path smoothing only optimizes weighted wire length, it can pack modules in a local region leading to placement congestion. The effects of placement congestion are two-fold: (a) there might not be enough space for the optimization step thereby degrading its performance, (b) it can cause severe routing congestion. Hence, to alleviate placement congestion, a congestion mitigation step is performed at regular intervals in the iterative flow.

The goal of congestion mitigation is to maintain the density profile of the incoming placement to ITOP. The intuition being that preserving the incoming placement distribution will yield a final placement with similar routing congestion. To have an accurate view of the local placement distribution and limit module movement, congestion mitigation attempts to satisfy a "bin density target" as opposed to a "global density target" for the entire design.

To determine the bin density target, a regular bin-grid is imposed over the placement image, and the density of each bin is determined from the incoming placement. The density of a bin is defined as the ratio of the total movable area to the total free-space within the bin. The bin density target is then a function of the bin density and the global density target. Let,

- G_{DT}: The global density target for the design.
- d_b: Incoming density of a bin in the bin-grid.
- δ: Overfill factor (a fixed, additional density allowed within bins for which $d_b \leq G_{DT}$).
- d_{max}: Maximum incoming bin density above which no overfill (additional area) is allowed.

Then, the density target for bin b (DT_b), is given by the following piecewise linear function:

$$DT_b = \begin{cases} (1+\delta)G_{DT} & d_b \leq G_{DT} \\ (1 - \frac{\delta G_{DT}}{d_{max} - G_{DT}})d_b + \frac{\delta G_{DT} d_{max}}{d_{max} - G_{DT}} & G_{DT} < d_b < d_{max} \\ d_b & d_b \geq d_{max} \end{cases}$$

The intuition behind the above function is as follows:

- For bins with an incoming density below d_{max}, the density target is higher than the incoming density. This lets the bins to get marginally overfilled during ITOP to improve timing. These bins are divided into two categories: (a) Bins with $d_b \leq G_{DT}$ are quite sparse. Hence, the additional module area within them can be higher than other bins. (b) Bins with $G_{DT} < d_b < d_{max}$, the additional area allowed is decreased with an increase in the incoming bin density.
- Bins with an incoming density equal and above d_{max} are highly congested to begin with. To have space for optimization and prevent routing issues, no overfilling is allowed within them during ITOP.

To spread the modules from over-congested bins with minimal impact to timing, an enhanced version of local refinement [20, 21] is used. The enhancements being: (a) during each iteration, a bin-blocking mechanism is used, preventing module movement from any bin with density less than DT_b, (b) the modules in an over-congested bin are moved out in the decreasing order of their *movement scores.*

In addition to increasing the placement congestion, movement of the critical modules can increase the design wire length. Hence, after congestion mitigation, detailed placement transforms like module swapping, flipping, etc., in the spirit of [15] are performed to recover wire length.

5. SLACK HISTOGRAM COMPRESSION

At regular intervals in the iterative flow a global timing optimization step is performed, targeting the overall slack histogram. This step involves buffering and repowering a larger set of paths in the design. The reasons for performing periodic slack histogram compression are as follows:

- Movement of the modules on the critical paths over multiple iterations can accumulate and impact the timing on a large number of off-critical paths. Periodically increasing the scope of optimization can recover any degradation in the timing on the off-critical paths.
- Since timing optimization is performed on a larger set of paths, slack histogram compression also improves the convergence and efficiency of the overall algorithm.

6. THE ITOP ALGORITHM

Finally, Algorithm 1 gives the overall ITOP algorithm. From Algorithm 1, the values for the key ITOP parameters used in practice are:

- Maximum displacement for critical modules ($D1$): 1% of the chip diagonal.
- Maximum displacement for non-critical modules ($D2$): 5× circuit row height.
- Transition iteration (*itop_transition_iteration*): 20.
- Overfill factor (δ): 0.25 (Allow 25% overfilling of bins below the global density target).
- Maximum bin density upto which overfill is allowed (d_{max}): 0.90.

Algorithm 1 The ITOP algorithm

```
1: Phase 0: Initial Setup
2:     $D1 \leftarrow max\_displacement\_for\_critical\_modules$
3:     $D2 \leftarrow max\_displacement\_for\_non\_critical\_modules$
4:     impose an $M \times N$ bin-grid ($B$) over the placement region
5:     determine the placement density ($d_b$) for each bin $b \in B$
6:     determine the density target ($DT_b$) for each bin $b \in B$
7: end
8: $iteration \leftarrow 1$
9: repeat
10:     Phase 1: Critical Path Optimization
11:         perform static timing analysis
12:         identify the critical paths
13:         perform slack-based critical path threading
14:         repeat
15:             move critical modules to minimize critical path lengths
                (subject to displacement constraint $D1$)
16:         until (no improvement in total critical net wire length)
17:         legalize and fix the critical modules
18:         legalize the non-critical modules
19:         perform static timing analysis
20:         perform incremental buffer insertion and gate sizing
21:     end
22:     Phase 2: Cong. Mitigation, Wire Length Recovery
23:         if ($iteration$ % $itop\_transition\_iteration == 0$) then
24:             move non-critical modules to satisfy $DT_b$
                (subject to displacement constraint $D2$)
25:             legalize the non-critical modules
26:             perform detailed placement to recover wire length
                (subject to displacement constraint $D2$)
27:         end if
28:     end
29:     Phase 3: Slack Histogram Compression
30:         if ($iteration$ % $itop\_transition\_iteration == 0$) then
31:             perform static timing analysis
32:             perform slack histogram compression
33:         end if
34:     end
35:     $iteration \leftarrow iteration + 1$
36: until (no timing improvement)
```

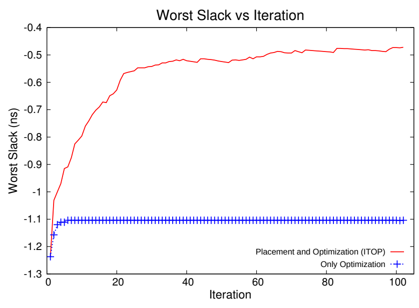
7. EXPERIMENTAL RESULTS

ITOP is implemented within the industrial physical synthesis tool PDS [1, 19]. This section presents experimental results on a set of high performance $65nm$ and $45nm$ industrial designs, with sizes ranging from $77K$ to $1034K$ modules. All reported runtimes are on a $2.93GHz$ Intel Xeon CPU running Linux. Finally, all timing numbers are generated on legalized placements using the EinsTimer static timing analyzer. The results are divided into two parts: (a) Sections 7.1–7.3 demonstrate the impact of the key ITOP components, and (b) Sections 7.4–7.5 compare ITOP with two representative flows embedded within PDS.

7.1 Effect of Placement During ITOP

Figure 6 shows the design timing during the iterative flow. The Figure of Merit (FOM) is the sum of all slacks below a *slack_threshold*. If the *slack_threshold* is zero, FOM is equivalent to the total negative slack. In Figure 6, the red (solid) line depicts the default ITOP flow (placement and optimization). The blue (dotted) line depicts the case when placement is skipped during each iteration. From Figure 6, the timing improvement during ITOP is not because of just running multiple iterations of timing optimization. Incremental placement is essential to improve the overall timing, and prevent optimization from getting stuck in a local minima.

(a)



(b)

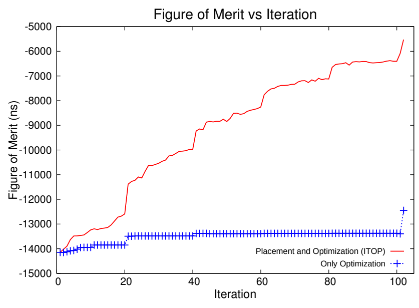


Figure 6: Design timing during the iterative flow. The red (solid) line depicts the default ITOP flow. The blue (dotted) line depicts the case when placement is skipped during each iteration.

7.2 Effect of Tunneling During Placement

Table 2 shows the impact of tunneling on the final design timing. It compares two runs: (a) No Tunneling: ITOP without tunneling, (b) With Tunneling: ITOP with tunneling during critical path smoothing. From Table 2, tunneling aids in significantly improving the worst slack and FOM.

7.3 Effect of Slack Histogram Compression

Table 3 shows the impact of running periodic slack histogram compression during the iterative flow. It compares two runs: (a) No Compression: ITOP without periodic slack histogram compression, (b) With Compression: ITOP with slack histogram compression at each transition point. From Table 3, running periodic slack histogram compression obtains upto 35% improvement in the FOM and 21% improvement in the number of negative paths at the end of ITOP.

Design	Worst Slack (ns)			Figure of Merit (ns)		
	No Tunneling	With Tunneling	%Improv	No Tunneling	With Tunneling	%Improv
ckt_5	-0.44	-0.31	29.55	-2902	-2636	9.17
ckt_6	-1.65	-1.07	35.15	-7083	-6298	11.08
ckt_9	-0.32	-0.06	81.25	-116	-48	58.62

Table 2: Effect of tunneling during placement.

Design	Figure of Merit (ns)			Number of Negative Paths		
	No Compression	With Compression	%Improv	No Compression	With Compression	%Improv
ckt_4	-762	-645	15.35	7658	7413	3.20
ckt_8	-359	-245	31.75	3100	2768	10.71
ckt_9	-74	-48	35.14	1266	994	21.48

Table 3: Effect of periodic slack histogram compression during the iterative flow.

7.4 Experimental Flows

To test its effectiveness, ITOP is compared with two representative flows shown in Figure 7. These are:

- NO-TDP: Physical synthesis flow that performs a single wire length driven global placement followed by coarse and fine timing optimization.
- TDP: Physical synthesis flow that augments the NO-TDP flow with a net-weight based timing-driven placement step. This is similar to the flow of Figure 1(a).

Please note, in both these flows, the fine timing optimization step also includes a timing-driven incremental placement algorithm in the spirit of [13, 14].

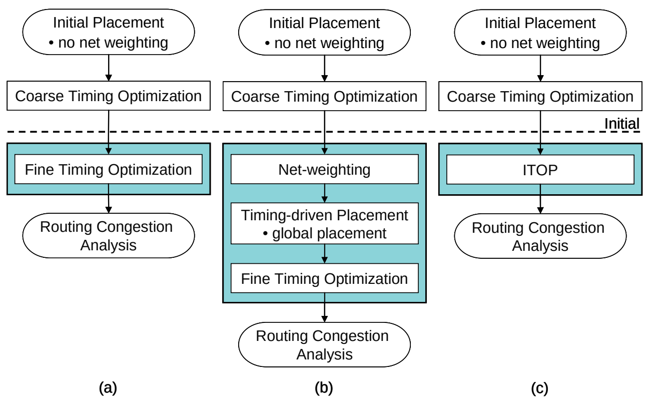


Figure 7: Experimental Flows. (a) NO-TDP: No timing-driven placement (b) TDP: Net-weighting and global timing-driven placement (c) ITOP. (Note: Detailed placement is run within fine timing optimization for the NO-TDP and TDP flows).

The following observations (validated by the results presented in Section 7.5) are first made regarding the NO-TDP and TDP flows:

- Since the NO-TDP flow performs only a single wire length driven global placement (without net-weighting), it should have better wire length and routing congestion compared to the TDP flow.

- On the other hand, the TDP flow should have better design timing, since it includes an additional net-weight based timing-driven placement step.

In this respect, the goal of ITOP is to obtain wire length and routing congestion comparable to the NO-TDP flow and design timing better than the TDP flow.

7.5 Results on Industrial Designs

Tables 4 and 5 compare the flows of Figure 7 on the following metrics: (a) the final worst slack, FOM and number of negative paths, (b) total Steiner wire length, (c) global routing congestion, (d) runtime of the flows. To perform global placement, all flows use the *RQL* [20] global placer. To perform net-weighting, the TDP flow uses the sensitivity guided net-weighting scheme [17]. In both tables, the values given along the *Initial* row are the numbers at the end of the coarse timing optimization step. In addition, for Table 5 the improvement obtained by the respective flows is evaluated w.r.t. the values given along the *Initial* row.
Design Timing: From Table 5, on average, **ITOP obtains the best timing among the three flows**. In terms of worst slack, ITOP obtains an average improvement of 56.2% and 47.45% above the NO-TDP and TDP flows. In terms of FOM, it obtains an average improvement of 38.91% and 9.88% above the NO-TDP and TDP flows. Finally, it reduces the number of negative paths by 30.93% and 10.69% over the NO-TDP and TDP flows respectively. Looking at individual results from Table 4, ITOP obtains the best result in terms of worst slack on 12/12 designs and best timing FOM on 9/12 designs.
Design Wire Length: From Table 5, on average, **ITOP obtains an average wire length improvement of 5% compared to the TDP flow,** and is only 3% worse compared to the NO-TDP flow.
Global Routing Congestion: At the end of the three flows an industrial global router is invoked to perform global routing congestion analysis. The metric used to measure the routing congestion is the number of nets above 100% congestion. This is defined as the number of nets assigned to a global routing edge which is using 100% or more of its routing resources. A lower value for this metric implies better routability. From Table 5, **the global routing congestion for ITOP is significantly better than the TDP flow and comparable to the NO-TDP flow**.
Runtime: Finally, from Table 4 (column eight), **the runtime for ITOP is comparable to the TDP flow for most of the designs.**

8. CONCLUSIONS

Timing-driven placement is a critical step in nanometer-scale physical synthesis. To improve design timing on a global scale, net-weighting followed by global timing-driven placement happens to be the most popular approach. This paper demonstrates that such an approach can improve timing, but significantly degrade wire length and routability. Alternately, it describes an incremental timing-driven placement algorithm called ITOP. Using accurate timing from an industrial static timer, ITOP integrates timing optimization within placement to improve design timing without degrading wire length and routability. Experimental results on high performance industrial designs show that ITOP significantly outperforms conventional timing-driven placement in terms of design timing and wire length, with negligible impact to design routability.

We are confident that there is room for further improvement both in terms of quality and runtime. For example, performing other optimization transforms like multi-threshold vt tuning, wire sizing, etc., during the iterative flow, has the potential to further improve the results and the overall runtime of the flow.

9. REFERENCES

[1] C. J. Alpert, S. Karandikar, Z. Li, G.-J. Nam, S. T. Quay, H. Ren, C. N. Sze, P. G. Villarrubia, and M. Yildiz. Techniques for fast physical synthesis. *Proc. IEEE*, 95(3):573–599, 2007.

[2] W. Chen, C.-T. Hsieh, and M. Pedram. Simultaneous gate sizing and placement. *TCAD*, 19(2):206–214, 2000.

[3] W. Choi and K. Bazargan. Incremental placement for timing optimization. In *Proc. ICCAD*, pp. 463–466, 2003.

[4] A. Chowdhary, K. Rajagopal, S. Venkatesan, T. Cao, V. Tiourin, Y. Perasuram, and B. Halpin. How accurately can we model timing in a placement engine? In *Proc. DAC*, pp. 801–806, 2005.

[5] H. Eisenmann and F. Johannes. Generic global placement and floorplanning. In *Proc. DAC*, pp. 269–274, 1998.

[6] B. Halpin, C. R. Chen, and N. Sehgal. Timing driven placement using physical net constraints. In *Proc. DAC*, pp. 780–783, 2001.

[7] W. Hou, X. Hong, W. Wu, and Y. Cai. A path-based timing-driven quadratic placement algorithm. In *Proc. ASPDAC*, pp. 745–748, 2003.

[8] S. Hur, T. Cao, K. Rajagopal, Y. Perasuram, A. Chowdhary, V. Tiourin, and B. Halpin. Force directed Mongrel with physical net constraints. In *Proc. DAC*, pp. 214–219, 2003.

[9] M. A. B. Jackson and E. S. Kuh. Performance-driven placement of cell based ICs. In *Proc. DAC*, pp. 370–375, 1989.

[10] A. B. Kahng and Q. Wang. An analytical placer for mixed-size placement and timing-driven placement. In *Proc. ICCAD*, pp. 565–572, 2004.

[11] T. Kong. A novel net weighting algorithm for timing-driven placement. In *Proc. ICCAD*, pp. 172–176, 2002.

[12] T. Luo, D. Newmark, and D. Z. Pan. A new LP based incremental timing driven placement for high performance designs. In *Proc. DAC*, pp. 1115–1120, 2006.

[13] M. D. Moffitt, D. A. Papa, Z. Li, and C. J. Alpert. Path smoothing via discrete optimization. In *Proc. DAC*, pp. 8–13, 2008.

[14] D. A. Papa, T. Luo, M. D. Moffitt, C. N. Sze, Z. Li, G.-J. Nam, C. J. Alpert, and I. L. Markov. RUMBLE: An incremental timing-driven physical-synthesis optimization algorithm. *TCAD*, 27(12):2156–2168, 2008.

[15] M. Pan, N. Viswanathan, and C. Chu. An efficient and effective detailed placement algorithm. In *Proc. ICCAD*, pp. 48–55, 2005.

[16] K. Rajagopal, T. Shaked, Y. Perasuram, T. Cao, A. Chowdhary, and B. Halpin. Timing driven force directed placement with physical net constraints. In *Proc. ISPD*, pp. 60–66, 2003.

[17] H. Ren, D. Z. Pan, and D. S. Kung. Sensitivity guided net weighting for placement-driven synthesis. *TCAD*, 24(5):711–721, 2005.

[18] B. M. Riess and G. G. Ettelt. SPEED: Fast and efficient timing driven placement. In *Proc. ISCS*, pp. 377–380, 1995.

[19] L. Trevillyan, D. Kung, R. Puri, L. N. Reddy and M. A. Kazda. An integrated environment for technology closure of deep-submicron IC designs. *IDTC*, 21(1):14–22, 2004.

[20] N. Viswanathan, G.-J. Nam, C. J. Alpert, P. Villarubia, H. Ren, and C. Chu. RQL: Global placement via relaxed quadratic spreading and linearization. In *Proc. DAC*, pp. 453–458, 2007.

[21] N. Viswanathan, M. Pan, and C. Chu. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In *Proc. ASPDAC*, pp. 135–140, 2007.

[22] Q. B. Wang, J. Lillis, and S. Sanyal. An LP-based methodology for improved timing-driven placement. In *Proc. ASPDAC*, pp. 18–21, 2005.

[23] Z. Xiu and R. A. Rutenbar. Timing-driven placement by grid-warping. In *Proc. DAC*, pp. 585–590, 2005.

Design	Flow	W_Slk (ns)	FOM (ns)	No. of Neg Paths	Steiner WL (xe6)	Routing Cong. #Nets $\geq$ 100%	Runtime (sec)
ckt_1	Initial	-2.62	-1750	2590	94.70	80	–
Objs: 77K	No-TDP	-2.54	-1492	1973	94.16	77	1023
Nets: 61K	TDP	-3.30	**-1223**	2193	101.23	81	2538
	ITOP	**-1.01**	**-1235**	1918	94.92	80	2758
ckt_2	Initial	-0.43	-169	1164	16.97	3053	–
Objs: 102K	No-TDP	-0.15	-34	370	15.96	2243	674
Nets: 104K	TDP	-0.17	-59	633	17.99	4776	1526
	ITOP	**0.07**	**-2**	210	16.56	2438	1449
ckt_3	Initial	-0.23	-89	693	28.81	132	–
Objs: 142K	No-TDP	-0.17	-50	415	27.79	152	831
Nets: 145K	TDP	-0.12	-31	476	27.68	0	1931
	ITOP	**-0.01**	**-13**	259	27.89	117	1435
ckt_4	Initial	-0.64	-2036	11041	47.21	262	–
Objs: 171K	No-TDP	-0.51	-1469	9548	46.86	218	1046
Nets: 176K	TDP	-0.35	**-581**	6563	52.77	4756	2227
	ITOP	**-0.27**	-645	7413	47.25	339	3524
ckt_5	Initial	-1.25	-6244	25881	118.09	158	–
Objs: 260K	No-TDP	-1.01	-5273	23801	115.98	158	1686
Nets: 269K	TDP	-0.83	-3076	18203	125.04	486	3148
	ITOP	**-0.31**	**-2636**	20971	117.37	158	4154
ckt_6	Initial	-2.03	-8256	12085	117.14	3837	–
Objs: 298K	No-TDP	-1.71	-6566	11190	115.97	3796	1416
Nets: 313K	TDP	-2.47	-6407	12114	143.74	31197	3558
	ITOP	**-1.07**	**-6298**	11652	128.61	6406	4273
ckt_7	Initial	-1.24	-14147	39635	185.27	246	–
Objs: 433K	No-TDP	-1.16	-11811	36981	182.46	169	2765
Nets: 441K	TDP	-0.75	-7083	32658	195.84	2481	5835
	ITOP	**-0.47**	**-5507**	34124	185.99	324	6779
ckt_8	Initial	-0.98	-1795	6377	154.05	6608	–
Objs: 451K	No-TDP	-0.92	-1149	4305	147.46	6565	3631
Nets: 465K	TDP	-0.56	-449	2905	161.68	52486	10612
	ITOP	**-0.16**	**-245**	2768	147.62	6595	8426
ckt_9	Initial	-0.90	-1537	7219	132.63	919	–
Objs: 476K	No-TDP	-0.68	-683	4008	128.15	917	2587
Nets: 490K	TDP	-0.46	-163	1451	127.89	1964	6227
	ITOP	**-0.06**	**-48**	994	129.80	991	8785
ckt_10	Initial	-0.61	-111	615	142.69	706	–
Objs: 554K	No-TDP	-0.46	-104	1186	135.51	769	3024
Nets: 562K	TDP	-0.40	-42	664	135.51	356	9583
	ITOP	**0.06**	**-2**	149	135.39	616	4100
ckt_11	Initial	-1.56	-96857	149269	341.62	1112	–
Objs: 951K	No-TDP	-1.26	-77375	148770	327.84	7313	9896
Nets: 961K	TDP	-1.02	**-19557**	83897	347.01	169232	33673
	ITOP	**-0.60**	-20616	98880	341.61	17039	36162
ckt_12	Initial	-2.19	-10987	36252	290.72	10864	–
Objs: 1034K	No-TDP	-1.90	-9088	42916	282.11	8951	6964
Nets: 1056K	TDP	-2.00	**-6462**	26548	297.69	36415	18965
	ITOP	**-1.01**	-6996	31605	295.98	13839	20386

Table 4: Design timing, wire length, routing congestion and runtime comparison between the NO-TDP, TDP and ITOP flows shown in Figure 7. (NOTE: For all designs, the values given along the "Initial" row are the results at the end of the coarse timing optimization step as shown in Figure 7.)

Flow	W_Slk Improv(%)	FOM Improv(%)	Neg Paths Improv(%)	Steiner WL (Ratio)	Routing Cong. (Increase)
Initial	0.00	0.00	0.00	1.00	0
No-TDP	20.30	29.51	11.16	0.97	279
TDP	29.05	58.54	31.40	1.05	23021
ITOP	76.50	68.42	42.09	1.00	1747

Table 5: Average results over all designs. For each flow, (1) Improv: Improvement of the flow over the "Initial" results, (2) Ratio: Final wire length of the flow / Initial Wire length, (3) Increase: Increase in the number of nets $\geq$ 100% compared to the "Initial" results.