16.3

Efficient Timing Closure Without Timing Driven Placement and Routing

Miodrag Vujkovic
Dept. of Electrical Eng.
University of Washington
Seattle, WA 98195
(206)-685-8678
miodrag@
ee.washington.edu

David Wadkins
Dept. of Electrical Eng.
University of Washington
Seattle, WA 98195
(206)-685-8678
d90mhz@
ee.washington.edu

Bill Swartz
InternetCAD.com, Inc.
10880 Cassandra Way
Dallas, TX 75228
972-613-6772
bills@
internetcad.com

Carl Sechen
Dept. of Electrical Eng.
University of Washington
Seattle, WA 98195
(206) 313-0180
sechen@
ee.washington.edu

ABSTRACT

We have developed a design flow from Verilog/VHDL to layout that mitigates the timing closure problem, while requiring no timing driven placement or routing tools. Timing issues are confined to the cell sizer, allowing the placement algorithm to focus solely on net lengths, resulting in superior layout densities and much lower power. The primary enablers to this new technology are: 1) gridded transistor sizing, 2) variable die routing that allows each net to be routed in the shortest possible length, 3) simultaneous cell placement, routing, gate sizing, and clock tree insertion, and 4) an effective incremental (ECO) placement that preserves net lengths from one iteration to the next. The variable die router is the key enabler. Superior placement and routing densities result from this new variable die approach. In addition, each net is routed at or near minimum length, and thus minor placement changes or cell size changes do not materially impact net lengths.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids – *placement and routing*, *layout.*

General Terms

Design, Performance, Algorithms.

Keywords

Digital design flow, gate sizing, placement and routing, timing closure.

1. INTRODUCTION

Circuit designers spend a large part of their time executing and optimizing logic-synthesis and physical-implementation (place-and-route) tasks. Timing closure refers to the application of electronic design automation (EDA) tools and design techniques to minimize the number of iterations required between these tasks to meet chip-timing specifications. The advent of deep-submicron (DSM) technologies shifted the design paradigm from a conventional logic-dominated to an interconnect-dominated design process [1]. Today, the timing closure problem is largely caused by the inability to predict interconnect loading with adequate precision prior to physical design [2]. Logic is optimized with respect to timing/area/power goals with assumed interconnect capacitances, but their actual values are not known until after layout phase. Statistical wire load models can be good predictors of the average wire load, but for the few nets on a critical path, the timing prediction can be very inaccurate resulting in a negative slack and thus require several re-iterations of synthesis and placement. In order to achieve a desired timing closure result, physical design has to be tightly integrated with other design process steps.

Different design flows have been proposed to address DSM timing closure challenges [3]. One of them iterates between gate-level synthesis and place-and-route phases. After the P&R phase, the netlist is back-annotated with the extracted net capacitances and synthesis is repeated. This can often result in a different netlist, with different placement and routing, and thus different timing, without any guarantee that the process will converge. Several placement-aware synthesis flows have been reported. They perform a partial physical design to compute more accurate loading estimates and use it for logic remapping in an iterative fashion. The integrated synthesis and placement flow reported in [4] restricts synthesis to only change power levels of the original cells during placement. However, placement alone may not be able to capture all of the interconnect effects, and some amount of routing detail is necessary. One of the most popular placement techniques required for timing closure is timing-driven placement. The success of timing-driven placement relies on the quality of the placement and the accuracy of the timing model. If the timing model does not capture design details such as routing topology, or capacitive coupling, the timing-driven effort may be ineffective [1]. Timing-driven placement may perturb placement at the expense of wire length and routability, causing speed degradation. The most common method for optimizing placement for timing is to employ net weights, but it is not possible to predict the net length obtained in response to a net weight. A natural choice for controlling delay is a net length constraint (NLC), limiting the maximum length of a net. Several detailed placement techniques to optimize for NLCs are presented in [5].

A design methodology emphasizing early timing closure for a high-frequency microprocessor design was proposed in [6]. Its main characteristics are logic partitioning on timing boundaries, predictable dataflow and control structures (PLAs), low skew and jitter clock distribution, deterministic placement of macros and a refinement method for chip integration. A semi-custom design

DAC'04, June 7–11, 2004, San Diego, California, USA.

methodology that incorporates automated transistor sizing has been reported in [7]. The basic building blocks used in this methodology are a set of parameterized static CMOS gates suitable for circuit tuning. Another key component of this flow is script-based cell generator designed to produce optimal layout for the parameterized set. A similar flow called the Power and Performance Optimization (PPO) flow was developed to achieve higher performance and to reduce power consumption in cell-based designs [8]. The PPO flow starts from an implemented design and optimizes the transistor sizes in each cell in the design to increase the performance and/or reduce the power dissipation. Also, in [9] a post-layout transistor sizing method downsizes devices inside a cell for power reduction without any modification of wiring. Their cells have fixed width and a standard cell height of 13 interconnect pitches, allowing for a wide range of transistor widths (0.9-6.2um for a 0.35um process). Many transistors will be downsized closer to the lower limit, resulting in a significant amount of wasted area.

What makes the timing closure problem particularly difficult for current electronic design automation (EDA) flows is the high variability in the loading of wires. We measured the capacitance per unit length for all nets for a wide variety of circuits in the 0.18um TSMC technology. We found that the capacitance per unit length varied from a high of about 0.7 fF/um to a low of about 0.02fF/um, a 35X variance! There is no way for a timing driven placement approach to be effective in meeting timing, especially given that an appreciable part of the delay is due to interconnect, and given that this portion is unpredictable. It's also apparent that meeting timing with a small cell library, with a very limited number of drive strengths and beta ratios, is either virtually impossible or extremely wasteful of power.

Timing Closure Design Flow

1) Iterative design synthesis using Synopsys DC
2) Add wire load model to synthesized netlists
3) Delay/power analysis (Pathmill/NanoSim)
4) "Optimized" curve extraction after synthesis
5) Select point(s) from the "optimized" curve (e.g., target delay, min PD point, min PDA point, or a set of points spanning the P-D curve)
6) Convert complex series-parallel cells to the set of basic functions in our optimized library
7) Initial Power/Delay optimization using AMPS
8) "Optimized" P-D curve extraction
9) Calculation of total cell width for each point on P-D plot
10) Choose desired points from one or more delay sub-ranges
11) Cell layout generation using *Cellgen* (actually: extract desired cells from library if previously generated, else generate
12) High quality (iTools) placement run for each point
13) iTools routing (novel variable die approach)
14) Accurate hierarchical parasitic (RC) extraction using Assura RCX from Cadence
15) Logical-effort based clock-tree sizing
16) Detailed skew measurements to the actual FF loads, including distributed RC parasitics
17) Delay/power/area measurements for each generated layout
18) If converged (usually need 1 or at most 2 iterations), stop; else do steps 19 and 20
19) New AMPS optimizations for one or more layout points
20) "Optimized" curve extraction over all optimizations
21) Re-execute steps 9-17 where step 12 is now an ECO-based placement run (iTools)

Figure 1

In this paper we present a fully automated design flow that minimizes power consumption for a user-specified level of performance, and that achieves timing closure without using timing-driven placement and routing. Our flow is performed is based on transistor-sizing, foundry-independent cell generation and a novel approach to placement and routing.

2. DESIGN FLOW

The basic steps in our design flow are shown in Figure 1. This flow is foundry independent. However, in this paper some steps are illustrated using the TSMC 0.18um technology, with native design rules, featuring a drawn channel length of 0.18um and a fanout-of-four inverter delay of 84 ps. Additional details regarding each of the design flow steps in Fig. 1 are presented in the remainder of this section.

2.1 Design synthesis – step 1

Design Compiler from Synopsys was used for synthesizing the RTL descriptions into optimized, technology-dependent, gate-level designs). Arithmetic blocks are generated using the DesignWare Foundation Library. Design Compiler (DC) was run iteratively to generate optimal points for various target delays (or throughput frequencies). DC starts from the minimum area point that usually corresponds to the lowest frequency range, and reduces the maximum delay in each subsequent iteration. Our experiments have shown that 20-30 iterations are needed to obtain a wide delay-power (area) range.

We use both DC Ultra and power optimization to produce an optimized synthesized netlist. One of the most important features of the DC Ultra flow is BOA (Behavioral Optimization of Arithmetic). BOA is particularly effective at optimizing a design with a relatively large number of arithmetic computations in the form of tree structures, especially with large operand bit-widths. Power optimization is performed using Power Compiler on a mapped design that already has been optimized for delay and area. Power optimization utilizes annotated switching activity; we use the switching activity from RTL simulation. Power optimization is performed during the second compilation, in incremental mode. During that compilation, timing and area constraints are still valid. The level of improvement in this phase depends heavily on cell library features, such as the range of beta ratios and cell drive strengths available.

Designs are mapped into a generic cell library consisting of 160 combinational cells. The various library functions are available in four different drive strengths. Inverters and buffers are available in two versions (symmetrical, for clocks, and asymmetrical), and also in nine drive strengths. No complex cells (*e.g.* AOI/OAI) have a transistor stack of more than two, based on previous research that showed this to be optimal [11]. Our research has also found it to be quite effective to include certain two-level series/parallel cells (as typically found in an Artisan library, *e.g*., AND2-4, OR2-3, NAND2B, NOR2B, OAI2BB1, AOI2BB1, *etc*.). We include a total of 20 flip-flops; five different flip-flop topologies in four different drive strengths. Our research has

found that it is sufficient to provide only positive edge-triggered flip-flops, with and without synchronous and asynchronous reset.

2.2 Wire load model – step 2

For the 0.18um technology we estimated the wire capacitance to be approximately 0.2fF/um. We used a basic linear fanout-length relationship for estimating wire lengths; based on past experience, we estimated the wire lengths to be 17um per fanout, corresponding to 3.4fF per fanout. After design synthesis these wire loads were added to each node. During transistor-level optimization, wire capacitances are kept fixed. Later on, after the initial place and route phase, the circuit optimizer will be provided with extracted wire loads.

2.3 Power-Delay Curve Generation – steps 3-5

For each synthesized design, we run static timing analysis using PathMill and dynamic power simulation using NanoSim to determine the worst-case delay and average power consumption *(step 3)*. The dynamic power simulation runtime is strongly dependent on circuit size and the number of test vectors. For smaller circuits, we used up to 512 test vectors; for larger circuits, the number of test vectors is reduced to 128 to decrease the computation time. For very large circuits, we replace dynamic power simulation with static power simulation, based on the switching activity from gate-level simulation. Although Design Compiler gives delay and power estimates at the end of compilation, we have found that transistor-level measurements are far more accurate.

Among all synthesized points, only the dominant points are extracted to define the "optimized" power versus delay curve *(step 4)*. An optimized power versus delay curve obtained after the synthesis step for benchmark *k2* is shown in Figure 2. After "optimized" curve extraction, we pick *(step 5)* one or more points that satisfy user-specified criteria (*e.g.*, minimum power-delay product, as shown here, minimum power-delay-area product, or a specific target delay or throughput). The chosen implementations are the starting point for transistor-level optimization.

2.4 Transistor-Level Optimization – Steps 6 - 10

The first step in transistor-level optimization is netlist conversion (*step 6*). While the inclusion of complex two-level cells (mentioned in Section 2.1) yields better synthesis results, we have found that the best transistor-level optimization results are obtained if these cells are first decomposed to basic (single-level)

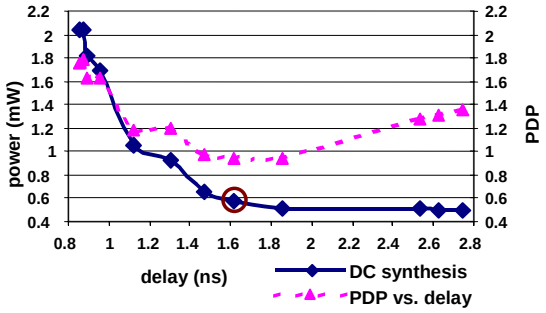

Figure 2. Iterative design synthesis – each point on the "optimal" curve represents one possible synthesized design. The circled point was chosen since it provides the minimum PDP (Power-Delay Product).

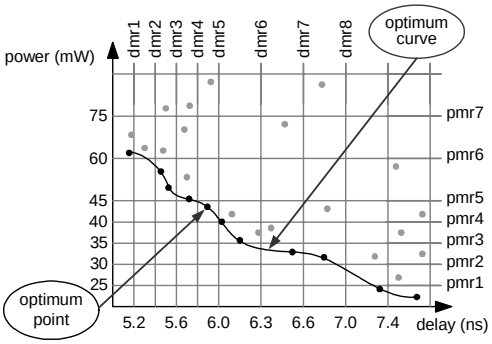

Figure 3. AMPS optimization process consists of target delay (dmr) and target power (pmr) optimization runs. Each run generates a set of points in power-delay space.

cells. In prior work, we showed that a library consisting of 11 basic functions yields the best power-delay performance for combinational benchmarks [11]. The 11 basic functions are: INV, NAND2-4, NOR2-3, AOI21, AOI22, OAI21, OAI22 and XOR. However, if a design includes arithmetic functions, better performance is obtained if full adder and multiplexer cells are also included in the base library. Of course, for sequential designs, flip-flop cells are also included in the library; however, thus far we have not found it beneficial to optimize the flip-flop cell sizes at the transistor level.

Our library contains both pass-transistor and static CMOS versions of the full adder, XOR cell, and two-to-one MUX. Our research has shown definitively that pass-transistor cells are (only) beneficial in the high-speed portion of the power versus delay curves, while static CMOS cells are considerably more beneficial in the low power regime (not surprisingly, since the static CMOS versions occupy considerably less area).

We use AMPS from Synopsys for transistor-level optimization within the three-dimensional optimization space of delay/area/power [9]. We provide AMPS with a circuit netlist in *spice* format, as well as a set of input patterns used for power estimation. AMPS estimates power and delay based on the PowerMill/Nanosim and PathMill tools [9].

The AMPS optimization process consists of a series of delay-mode (DM) runs (dmr1, dmr2…) and power-mode (PM) runs (pmr1, pmr2…) for a desired range of target delay and power values, as shown in Fig. 3. Each AMPS run consists of 10 iterations *(step 7)*. While it is possible to do more iterations, we found no advantage in doing so. Typically we perform 4-8 different target delay and power runs based on the desired ranges. During the optimization process AMPS generates a large number of points in (power, delay) space. Among them, only the dominant points are retained to define the optimized power vs. delay curve *(step 8)*.

The transistor-level optimization for all benchmarks and cell libraries was performed using the sizing mode in which all channel-connected nMOS devices for a cell are sized as a group, and all channel-connected pMOS devices are sized as a separate group. Series/parallel cells have only one group of nMOS devices and one group of pMOS devices while pass-transistor cells can have several nMOS/pMOS transistor groups.

AMPS is provided with a discrete set of allowable transistor sizes. For larger widths, the sizes correspond to an integral number of half folds (*i.e*. transistor widths are 1.0, 1.5, 2.0, 2.5, *etc*., times the maximum nonfolded size. For the 0.18um process, the allowable sizes for both nMOS and pMOS devices range from 1.06um to 13.78um, in steps of 0.53um, and approximately in steps of 0.13um between 0.42um and 1.04um. We have found that allowing a continuous size range for the device sizes does not improve the quality of the results.

After "optimized" curve extraction, netlists for selected points on the power-delay curve are generated. The area of a design point is estimated by the total cell area *(step 9)*. The area of a cell is that produced by *Cellgen* (described in the next subsection). In next step *(step 10)*, we divide the delay range of interest into one or more subranges and choose one point from each sub-range that satisfies a specified criteria (*e.g*., minimum power-delay product, minimum power-delay-area product, or a specific target delay or throughput). The set of chosen points from the initial transistor-level optimization represents the input to the layout phase.

2.5 Cell Layout – Step 11

Layouts for different cells created during the transistor-level optimization are generated using our automatic, parameterized standard-cell layout generator *Cellgen (step 11)*. Once a cell layout is created, it is stored in our library so that a cell with these transistor sizes does not have to be laid out again for the same technology. Since the number of possible cell sizes is finite, we could simply generate all the possible cell layouts and then validate them via fabrication. This would then constitute a fixed library approach, albeit with a fairly large number of cells (on the order of 2000). Indeed this is our plan of record.

Cellgen reads the SPICE output from AMPS to determine the transistor sizes for a set of 37 different combinational cell types (both basic and multi-level) and 5 sequential cell types. *Cellgen* uses a single contact routing method that allows for a variable transistor size that does not affect the general topography of the internal cell routing as shown in Fig. 4. For the 0.18um technology, pMOS transistors have a range of 0.42um to 1.68um per fold, while nMOS transistors have a range of 0.42um to 1.06um per fold. The type of cell and number of folds required to meet the target transistor sizes solely determines the size of a cell. *Cellgen* was designed for cells with equally sized transistors in each network (pull-up or pull-down). Exceptions are the non-symmetrical AOI21/OAI21 cells, where transistor sizes in two different branches can be different. Pass-transistor cells have several groups of equally sized transistors; *e.g*., in the pass-transistor XOR2 cell, all inverters are sized separately, but all nMOS (and pMOS) transistors in transmission gates will be the same.

Although sizing optimization is performed on single-level cells, we found it highly beneficial to create compound or merged cells prior to global placement and routing. *Cellgen* extracts two single-level cells from the library and merges them into a single place and route instance. "Merged cells" are created from basic cells (*e.g*. a NAND2 and an output inverter make a new AND2 cell, whereas an input inverter and NAND2 cell constitute a NAND2B cell). The cell generator wires up the sub-cells, avoiding the use of metal layers as much as possible. This improves subsequent global placement and routing, since vastly fewer nets need to be routed by the global router. The use of the compound cells reduced the number of nets to be routed by an average of 60% for a set of benchmark circuits.

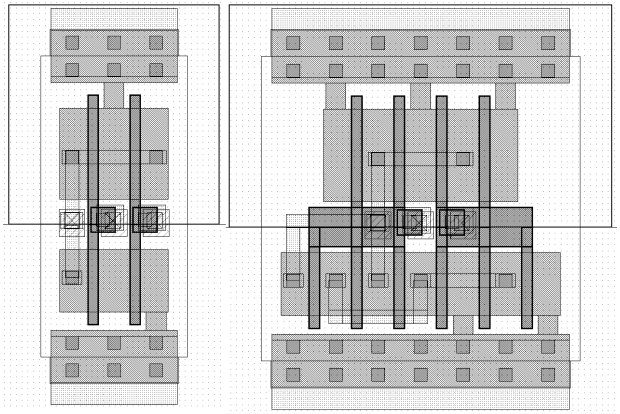

Figure 4. NAND2 cell requiring no folds (left), and NAND2 cell requiring 1 pMOS fold and 2 nMOS folds

2.6 Placement and Routing – Step 12-14

It is our experience that all attempts at meeting timing should reside with the cell sizer, post (initial) layout. This is because at this point the cells can have their sizes optimized for the actual extracted wire loads. A critical advantage of this new approach is that the placement tool can focus on only a single objective, that of minimizing wire length. As a consequence, much lower wire lengths are achieved since the placement algorithm is not distracted by various other concerns such as timing, congestion, *etc*. Of course, considerably lower total wire lengths lead to considerably lower wire loads, and thus to much lower power for the same circuit performance.

Particular placement and routing approaches are required by this new flow. Since minimization of wire length is crucial for minimizing power consumption, we need an approach that generates placements having the lowest possible total wire lengths. Based on our experience, the iTools placer from InternetCAD.com yields substantially lower total wire lengths compared to other EDA placers. Hence iTools is employed.

Furthermore, since resizing cells perturbs the initial global placement and routing, we demand an effective ECO (engineering change order) mode. In particular, after cells have been resized, we require that the placement be topologically similar to the previous one, and that the wire lengths are also largely the same. Otherwise, the loading on the cells will be quite different, invaliding the resizing.

The major EDA companies and much of academic research for more than 10 years have employed what are essentially gate array (or fixed die) routers for standard cell circuits. This hardly makes sense since the die is programmable on all mask layers in the standard cell style, but the EDA companies found it easier to develop reasonably effective fixed die routers as opposed to variable die routers. The latter are able to create routing space wherever needed in order to complete the routing. Hence, variable die routers inherently complete 100% of the routing whereas in fixed die routing this guarantee is absent. The routing quality obtained from a fixed die router is highly dependent on the quality of the user, and on the time spent by the user in numerous

iterations to eliminate unneeded routing space and/or to add needed routing space.

However, the variable die router has another monumental advantage: each net is routed in near minimum length, every time. Congestion has no impact on the variable die router. It simply creates space (*e.g.* increasing row separations or adding feedthroughs) wherever needed to wire up a net using the minimum or near minimum total wire length. Thus, a minor placement change, such as what might occur after an ECO placement run, will yield almost the same lengths for each net.

The fixed die router, however, is simply bullied by congestion. If a congested area lies between two pins, the router has no choice but to detour around it. Invariably this net seems to lie on a critical path and now the load for the driving gate is dramatically larger than before, causing timing to diverge. Furthermore, a minor placement change often dramatically changes the congestion landscape, causing nets to become much longer (or shorter) than previously. As a consequence, significant cell sizing must be performed; this process seems to loop forever. It is our experience that a fixed die router renders our flow to be largely ineffective in accomplishing timing closure.

In our flow, we use the iTools gridded variable die router. To our knowledge, this is the only commercial (or academic) variable die router available. While a non-gridded version is available, the gridded version is considerably faster and *Cellgen* produces gridded cells anyway.

The iTools placer is executed on each design *(step 12)*. The initial circuit was sized according to the fanout-based wire length estimates and placement was performed with no constraints on the wire lengths. We then add wire length constraints that restrict all nets to be shorter than the longest 10% of the nets in the first run. This eliminates the long nets, but has very little effect on performance. The placer is then run one or more times with the above constraints. The wire capacitances were estimated from the reported wirelengths from the iTools placer assuming a wire capacitance of 0.2fF/µm. and passed to the routing phase *(step 13)*.

Parasitic extraction was done with Assura RCX from Cadence *(step 14)*. The DSPF (Detailed Spice Parasitic Format) option was used to extract hierarchically to reduce run-time. At this point Assura is run on all the cells to extract internal parasitic capacitances. Once the netlists are generated, a formatted name was assigned to each cell needed for a design. This name uniquely identifies the cell based on the transistor sizes. Extraction and verification of each new cell is performed only once. All extracted cells are stored in the database and reused in different designs.

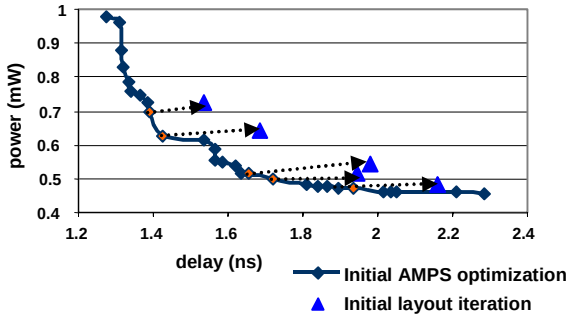

Figure 5. After initial AMPS optimization, 5 points were chosen from the predicted "optimized" curve for the initial layout phase (step 10). Layout phase produces realistic points in power-delay space.

2.7 Clock Tree Synthesis – Steps 15-16

In traditional flows, clock tree synthesis is performed after placement. This greatly disturbs cell placement, and consequently wire lengths, making timing closure extremely difficult. We augmented the iTools placer to add a hierarchical H-tree of symmetrical inverters during the actual placement process. Just like the rest of the flow, the clock tree synthesis is a refinement process. Initially, iTools equalizes the transistor loads on the leaf inverters; however, it cannot assure an equal RC wire load to each flip-flop during placement since routing is yet to be performed.

After initial global placement, each inverter in the tree is sized using logical effort. The result is that the fanout capacitance is constant for each leaf inverter, which in turn tends to equalize the rise and fall time for each buffer and reduces the overall skew. However, sizing cannot compensate for the skew between two flip-flop loads due to RC effects. The residual clock skews are accurately simulated using Hspice on the Assura (RC) extraction of the clock tree directly to the individual flip-flop loads. This skew information is incorporated into the next optimization step, where it is used effectively to optimize the timing *(step 16)*.

2.8 Power/Delay/Area Measurements and Resizing – Steps 17-21

After the layout phase, we perform timing and power measurement using PathMill/NanoSim and compare obtained values with predicted values from the "optimized" curve after

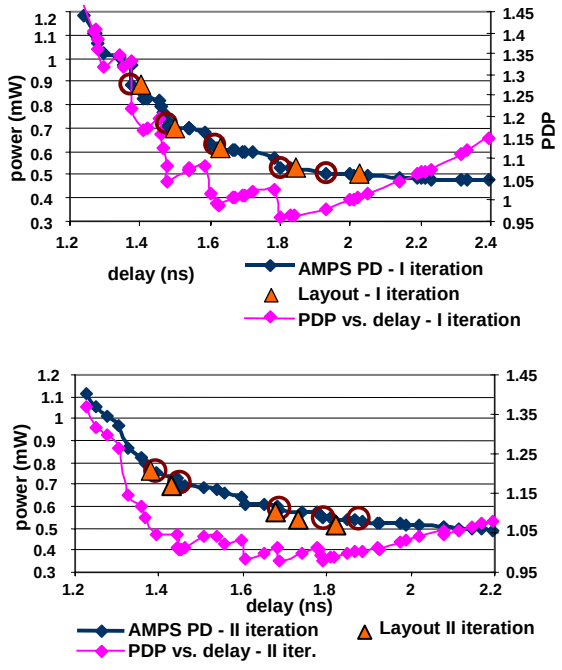

Figure 6. (top) Predicted "optimal" power-delay curve, set of chosen points based on the minimum PDP in specified delay sub-range (circled) and set of actual points after layout phase (triangles) for first (I) and second (II) iteration (bottom)

AMPS optimization. Figure 5 shows a power-delay curve after initial AMPS optimization, a set of chosen points from the curve, and a set of obtained points after the first layout phase and parasitic extraction. The initial "optimized" curve is produced assuming simple wire load models, and therefore the deviation of actual points from predicted points is expected to be the largest in this iteration *(step 17)*.

We run one or more AMPS optimizations *(step 19)* and extract the "optimized" curve over all optimizations *(step 20)*. "Optimized" curves from individual optimizations contribute to part of the overall curve (in the vicinity of starting point). Contrary to the initial iteration, the circuit optimizer is provided with extracted wire loads in any subsequent iteration. Steps 9-17 defined in Fig. 1 are repeated if necessary. The iTools placement is executed in ECO mode using wire length constraints from the previous iteration. The iTools ECO mode initializes the placement of all the cells according to the previous iteration, and then it makes very small moves necessary to even up rows and remove the cell overlap created by cells that changed sizes. Because of the wire constraints and ECO mode, the resulting wire lengths were very similar to what they were in the previous iteration. Typically, one or at most two iterations are needed to produce a converged timing result. Fig. 6 shows the predicted "optimized" power-delay curve, the set of chosen points (circled in both figures) and the set of actual points after layout and parasitic extraction (triangle points) for two iterations of our design flow.

3. PERFORMANCE IMPROVEMENT

To evaluate the circuit performance improvement obtained with our design flow, we compared power-delay products for a given design from the initial iteration to the last iteration. Figure 7 shows a typical progression of our design flow and the performance improvement from iteration to iteration for the *k2* benchmark. It is widely acknowledged that a better metric of performance is average power per cycle times delay squared, or effectively energy times delay squared (EDD). Results for several designs are given in Table 1. The average improvement in EDD after two iterations is 22%.

Table 1. Reduction in EDD

Design	Minimum EDD					
	Initial		iteration I		iteration II	
K2	1.7	1	1.6	1.09	1.4	1.21
C5315	17.8	1	16.6	1.08	14.5	1.23
C7552	29.7	1	25.5	1.17	23.2	1.28
32-stage FIR	1374	1	1208	1.14	1190	1.15
Avg improv.	1		1.12		1.22	

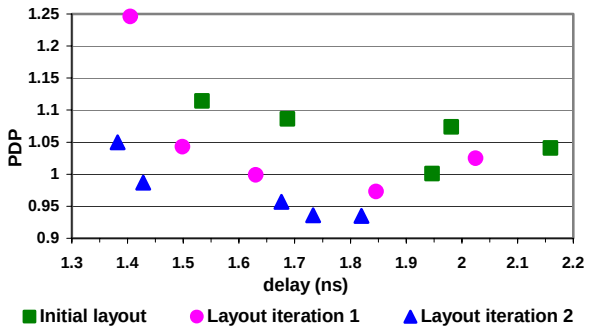

Figure 7. PDP vs. delay for design points after each layout iteration for *k2* circuit

4. CONCLUSION

We have presented a design flow from Verilog/VHDL to layout that mitigates the timing closure problem, while requiring no timing driven placement or routing tools. Timing issues are confined to the cell sizer, allowing the placement algorithm to focus solely on net lengths, resulting in superior layout densities and much lower power. The key enablers are: 1) gridded transistor sizing, 2) variable die routing that allows each net to be routed in the shortest possible length, 3) simultaneous cell placement, routing, gate sizing, and clock tree insertion, 4) an effective incremental (ECO) placement that preserves net lengths from one iteration to the next. We have demonstrated an average improvement of 22% in EDD after two refinement iterations.

5. ACKNOWLEDGMENTS

We are grateful for the financial support provided by the National Science Foundation (NSF), MARCO/C2S2, the NSF Center for the Design of Digital and Analog ICs (CDADIC), Boeing/DARPA, and Intel Corporation.

6. REFERENCES

[1] C.K. Cheng, "Timing Closure Using Layout Based Design Process", (www.techonline.com/community/tech_topic/timing_closure/116).

[2] R. Bryant, *et al*., "Limitations and Challenges of Computer-Aided Design Technology for CMOS VLSI", *Proc. IEEE*, Vol. 89, No.3, March 2001.

[3] O. Coudert, "Timing and Design Closure in Physical Design Flows", *Proc. of the International Symposium on Quality Electronic Design (ISQED)*, 2002.

[4] S. Hojat, and P. Villarrubia, "An Integrated Placement and Synthesis Approach for Timing Closure of PowerPC TM Microprocessors", *Proc. IEEE Int. Conference on Computer Design (ICCD)*, pp. 206-210, October 1997.

[5] B. Halpin, N. Sehgal, and C.Y. R. Chen, "Detailed Placement with Net Length Constraints", *Proc. of the 3rd IEEE Int. Work. on SOC for Real-Time Applications*, 2003.

[6] S. Posluszny *et al*., "Timing Closure by Design, A High Frequency Microprocessor Design Methodology", *Proc. of Design Automation Conf. (DAC)*, pp. 712-717, June 2000.

[7] G. Northrop, and P.F. Lu, "A Semi-Custom Design Flow in High-Performance Microprocessor Design", *Proc. of Design Automation Conference (DAC)*, pp. 426-431, June 2001.

[8] E. Yoneno, and P. Hurat, "Power and Performance Optimization of Cell-Based Designs with Intelligent Transistor Sizing and Cell Creation", *IEEE/DATC Electronic Design Processes Workshop*, Monterey, CA, April 2001.

[9] M. Hashimoto, and H. Onodera, "Post-Layout Transistor Sizing for Power Reduction in Cell-Base Design", *IEICE Trans. Fund.*, Vol.E84-A, pp. 2769-2777, November 2001.

[10] AMPS User Guide 5.4, Synopsys, 2000.

[11] M. Vujkovic and C. Sechen, "Optimized Power-Delay Curve Generation for Standard Cell ICs," *Proc. Int. Conf. Comp. Aided Design (ICCAD)*, San Jose, CA, November 2002.