[15] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: a system for sequential circuit synthesis," Dep. Electr. Eng. Comput. Sci., Univ. California, Berkeley, CA, Tech. Rep. UCB/ERL M92/41, 1992.

[16] Device Group, University of California, Berkeley. (2004, Jan.). *Predictive Technology Model* [Online]. Available: http://www-device.eecs.berkeley.edu/~bsim3

A Metal-Only-ECO Solver for Input-Slew and Output-Loading Violations

Chien-Pang Lu, *Member, IEEE,* Mango Chia-Tso Chao, *Member, IEEE,* Chen-Hsing Lo, *Member, IEEE,* and Chih-Wei Chang, *Member, IEEE*

***Abstract*—To reduce the time-to-market and photomask cost for advanced process technologies, metal-only engineering change order (ECO) has become a practical and attractive solution to handle incremental design changes. Due to limited spare cells in metal-only ECO, the new added netlist may often violate the input-slew and output-loading constraints and, in turn, delay or even fail the timing closure. This paper presents a framework, named metal-only ECO slew/cap solver (MOESS), to resolve the input-slew and output-loading violations by connecting spare cells onto the violated nets as buffers. MOESS performs two buffer-insertion schemes in a sequential manner to first minimize the number of inserted buffers and then resolve timing violations, if any. The experimental results based on industrial designs demonstrate that MOESS can resolve more violations with fewer inserted buffers and less central processing unit runtime compared to an electronic design automation vendor's solution.**

***Index Terms*—Engineering change order (ECO), physical design, slew/loading violation.**

I. Introduction

For current process technologies, the cost of photomasks increases dramatically per generation [1], [2]. To reduce this expensive cost of photomasks, the incremental design changes are enforced to be implemented by changing only the metal layers while the base layers (for cells) remain the same. As a result, the original photomasks used for printing the cells can be reused in the next tape-out. This reuse of the base-layer photomasks can not only save the cost of photomasks themselves but also reduce the tape-out turn-around time since the base layers could be manufactured in advance. This type of incremental design changes is referred to as metal-only engineering change order (ECO).

To realize metal-only ECO, some design techniques have to be developed. First, spare cells need to be spread all over the design so that the change can be made at every possible location. This spare-cell allocation determines the affordable ECO size and its area overhead. Electronic design automation (EDA) vendors already provide some solutions to it. Second, a more complicated router is required to efficiently handle a large number of existing obstacles and design rules in ECO. Some previous work addressed these issues by using an implicit connection graph [3], a graph-reduction technique [4], or a timing-aware router [5]. Third, the violations of timing factors may significantly increase after metal-only ECO. Thus, a solver which can automatically remove those timing-related violations is needed to shorten the timing closure of metal-only ECO. Unfortunately, the current solutions provided by EDA vendors are not effective enough.

Input slew and output loading are two important timing factors to sign off the timing closure, which are limited by the slew constraint and loading constraint, respectively. Any violation to these two constraints may lead to a wrong timing estimation of the design, and in turn, degrade its performance and yield.

Several buffer-insertion techniques [6]–[10] are proposed to resolve the violation of the slew, loading, and timing constraint. However, most previous works assume that its gate placement is able to change, and hence cannot be applied to metal-only ECO.

In metal-only ECO, solving the timing-related violation relies on the utilization of pre-placed spare gates. [11] proposed a technology-remapping technique to fix timing violations, which may require more pre-placed spare cells to support the desired remapping. [12] inserts constant values to the inputs of spare cells and applies a technology-mapping technique to replace the original cells with spare cells. It may require more universal but larger-area spare cells, such as and-or-invert and multiplexer.

Some commercial tools also provide options to support buffer insertions in metal-only ECO. However, the final location of the inserted buffers often deviates from the ideal location due to the lack of physical information on spare cells and routing resources during the buffer insertion.

This paper presents a metal-only-ECO framework, named metal-only ECO slew/cap solver (MOESS), which resolves slew and loading violations by using pre-placed spare gates as inserted buffers. MOESS also can resolve the timing violations, implicitly or explicitly. For each violation, the proposed framework first finds the best buffer candidates from all spare gates and then utilizes a commercial back-end tool to insert the selected buffer through the tool's interface. Therefore, the focus of this framework is on how to accurately estimate the output loading (input slew) of the buffers newly inserted with the adopted back-end tool. This framework can be applied based on any commercial back-end tool as long as the design database can be queried through an open interface.

II. Proposed ECO Slew/Loading Solver

A. Overall Flow of MOESS

After the metal-only ECO is done by using netlist difference and spare cell mapping, tools will report the pins violating slew or loading constraints. Based on this timing report, MOESS will insert buffers through a commercial automatic placement and routing (APR) tool [15] to resolve each slew or loading violation. In MOESS, the slew constraint of a gate g can be transferred into a corresponding loading constraint, denoted as OAL_g (output available loading). The definition of OAL_g is the maximum output loading of g which can generate a output

Manuscript received June 12, 2009; revised August 14, 2009. Current version published January 22, 2010. This paper was recommended by Associate Editor P. Saxena.

C.-P. Lu and C.-H. Lo are with Mstar Semiconductor, ChuPei, Hsinchu Hsien 302, Taiwan (e-mail: knuth.lu@mstarsemi.com; jack.lo@mstarsemi.com).

M. C.-T. Chao and C.-W. Chang are with the Department of Electronics Engineering, National Chiao-Tung University, Hsinchu 300, Taiwan (e-mail: mango@faculty.nctu.edu.tw; xesuschang@gmail.com).

Digital Object Identifier 10.1109/TCAD.2009.2040011

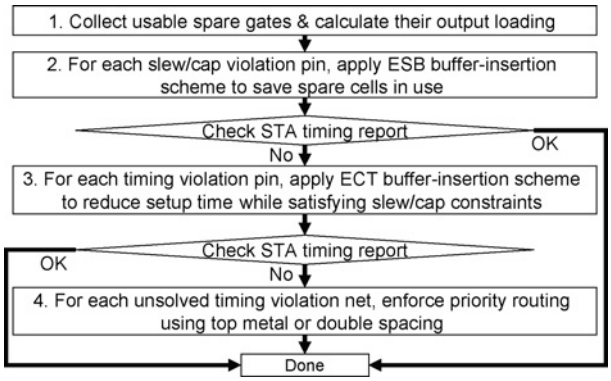

Fig. 1. Overall flow of MOESS.

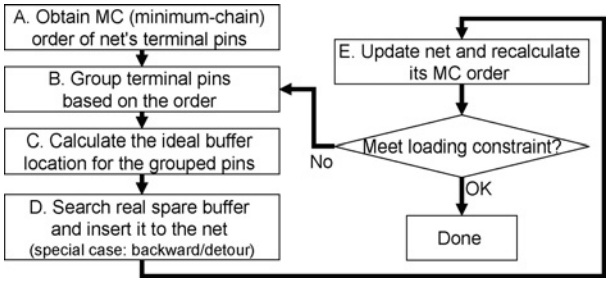

Fig. 2. Flow of ESB buffer-insertion scheme.

slew smaller than the slew constraint assuming that the input slew of g is equal to the slew constraint. Thus, input-slew violations can be resolved by the same approach as used for solving output-loading violations.

Fig. 1 shows the overall flow of MOESS. The first stage of MOESS is to increase the candidate pool of spare buffers by collecting usable spare cells. The second stage of MOESS is to apply ECO save buffer (ESB) buffer-insertion scheme to resolve the slew and loading violations using minimum buffers. After the slew/loading violations are resolved, most timing violations can be resolved as well. Then a timing-analysis tool will report the remaining critical paths violating timing constraints. Along those critical paths, we identify the nets which are inserted with buffers in stage 2. Then, we perform the ECO care timing (ECT) buffer-insertion scheme to re-insert the buffers and shorten their setup time while solving slew and loading violations. For the timing violations which still cannot be resolved, MOESS will enforce the priority routing on the critical nets by using top metal or double spacing, and then re-route the other non-critical nets.

Fig. 2 shows the steps of the ESB scheme for solving a violation net. Each step will be briefly introduced in later sections. For details, please refer to our previous work [13].

B. Obtain Minimum-Chain Order of Terminal Pins

When grouping the terminal pins to be driven by a buffer, we hope that the grouped pins could be not only in the same neighborhood and but also in the same direction toward the violation gate. Otherwise, the wire loading of driving the grouped pins can be too large. To obtain such grouping, we modified a minimum-chain (MC) algorithm in [14] to get the *MC order* of terminal pins. The concept of this minimum-chain algorithm is to assign the closest pin as the next ordered pin each time, starting from the violation gate g (the order of g is 0). By connecting the terminal pins one by one with such order, their total wire length can approach minimal. Also, the terminal pins with adjacent MC order are more likely in the same direction toward the violation gates as well.

C. Group Terminal Pins Using MC Order

In step B of Fig. 2, terminal pins of the violation net are first grouped assuming a type-t buffer b is used. We start from the buffer type with the highest driving capability to the one with the lowest. Then, we follow the MC order to serially add the terminal pins into the group p_list. The objective here is to obtain a group of pins p_list such that the output loading of b for driving all grouped pins in p_list is close to but not exceed the OAL_b. The following equation estimates the output loading of b for driving p_list [denoted as $GOL_b(p\_list)$]

$$GOL_b(p\_list) = \sum_{i=1}^{n} \left(InC_{p_i} + WL(p_i, p_{i-1}) \right) \tag{1}$$

where n is the size of p_list, p_i is the ith ordered pin in p_list, InC_{p_i} is the input capacitance of p_i, $WL(p_i, p_{i-1})$ is the wire loading estimation between p_i and p_{i-1}, and $WL(p_1, p_0)$ is equal to 0. The computation of $WL(p_i, p_{i-1})$ will be detailed in Section IV.

D. Calculate Ideal Buffer Location

To ensure that the inserted type-t buffer can drive all the grouped pins in p_list, we first calculate the *output remained load* of the buffer b, denoted as ORL_b, using

$$ORL_b = OAL_b - GOL_b(p\_list). \tag{2}$$

The amount of ORL_b determines the affordable wire length connecting from inserted buffer b to the last-ordered pin p_n in p_list. Thus, the ideal location of the inserted buffer b must satisfy the following equation:

$$|X_b - X_{p_n}| \cdot U_h(b, p_n) + |Y_b - Y_{p_n}| \cdot U_v(b, p_n) \leq ORL_b \tag{3}$$

where X_a and Y_a represent the X-axis and Y-axis coordinates of pin a, respectively. $U_h(p_1, p_2)$ and $U_v(p_1, p_2)$ represent the wire loading associated with a horizontal and vertical distance units, respectively. To make the buffer b closer to the source pin g, we limit the ideal location of b on the straight line between g and p_n, which is represented by

$$(Y_b - Y_{p_n})/(X_b - X_{p_n}) = (Y_{p_n} - Y_g)/(X_{p_n} - X_g). \tag{4}$$

Last, we can obtain the ideal location of b by solving both (3) and (4), assuming the equality holds in (3).

E. Search Real Spare Gate

We first use the Manhattan distance between the last-ordered pin p_n and the ideal buffer location as the radius to draw a diamond-shape region centered at p_n. The buffer found in this diamond-shape region can satisfy (3). To make the buffer closer to the violation gate g, we use the same radius to draw another diamond-shape region centered at the ideal buffer location. We then attempt to select the buffers locating in the intersection of the two regions. This searching can make sure that the selected buffer, if any, is on the way toward the violation gate g. Finally, we select the type-t buffer closest to the ideal location in the intersection region. If such type buffer

TABLE I
COMPARISON BETWEEN MOESS [15] AND ON SOLVING SLEW, LOADING, AND TIMING VIOLATIONS FOR MULTIPLE METAL-ONLY ECO PROJECTS

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Proj.	Instance	Pro-	Spare	ECO	#Violation		Worst Slew				Worst Loading				Worst Slack				#Spare Buffer			Runtime		
(Ver.)	Count	cess	Count	Size	Slew	Load	Ori.	[15]	MOESS	Imp.	Ori.	[15]	MOESS	Imp.	Ori.	[15]	MOESS	Imp.	[15]	MOESS	Imp.	[15]	MOESS	Speedup
da(3)	190.4K	0.18	7.6K	142	40	0	16.0n	5.0n*	1.9n	68%	4.3	<1	<1	–	−3.5n	−2.2n*	>0	100%	78	40	49%	20m	1m	19.0×
db(3)	210.8K	0.18	9.1K	1030	6	0	2.4n	1.8n	1.8n	–	<1	<1	<1	–	>0	>0	>0	–	12	5	58%	22m	1m	21.0×
dc(4)	242.9K	0.18	5.5K	507	71	0	6.6n	3.8n*	2.0n	47%	1.4	<1	<1	–	−2.3n	−0.3n*	>0	100%	91	68	25%	31m	3m	9.3×
dd(3)	309.3K	0.18	10.4K	1904	47	0	7.2n	2.1n*	2.0n	5%	1.9	<1	<1	–	−1.6n	>0	>0	–	83	48	42%	27m	2m	12.5×
de(2)	871.1K	0.13	62.4K	127	0	35	0.9n	0.9n	0.9n	–	1.4	1.22*	1.1	8%	>0	>0	>0	–	63	35	44%	172m	5m	33.4×
df(2)	1.3M	0.13	48.8K	1276	15	243	2.0n	1.3n*	0.9n	31%	3.1	3.5*	1.2	66%	−0.6n	−0.1n*	>0	100%	377	277	27%	222m	37m	5.0×
dg(4)	1.6M	0.13	80.5K	1702	166	258	5.6n	1.2n*	1.0n	17%	9.3	4.6*	1.2	74%	−0.2n	−0.4n*	>0	100%	314	259	18%	252m	45m	4.6×
Average										24%				21%				57%			38%			14.9×

cannot be found in the intersection region, then we change the buffer type to one with lower driving capability and repeat step B to step D.

F. ECT Buffer-Insertion Scheme

Although the slew or loading violation can be solved by the ESB scheme, the delay of some paths may exceed the timing constraint due to the extra gate delay of inserted buffers. Our internal experiments found that most of those timing violations result from the sharing of a common buffer between a timing-critical path and long new-added wires. Note that those new-added wires can be designed as multi-cycle paths to meet the timing constraint but the original paths cannot. Thus, the ECT scheme utilizes a similar approach as the ESB scheme but separate the grouping of long-wire terminal pins from the others, such that the loading of those long wires would not be added to the critical path.

III. EXPERIMENTAL RESULT OF MOESS

A. Experiment Setup

The ECO flow, including netlist difference, spare gate mapping, and routing, is performed based on a commercial APR platform [15]. We compare the results of MOESS with an EDA vendor's buffer-insertion solution for metal-only ECO [15]. In vendor's solution, we use the command "run gate buffer wire-slew/cap-ECO" to insert buffers for each violation net.

The benchmarks used in this experiment are all industrial projects. The spare-cell count in each project is 3–5% of the total cell count. The slew constraint in use is a pre-defined constant associated with the process technology and the cell library. The loading constraint in use is defined as a ratio to the value of the library-suggested constraint. In our experiments, the slew and loading constraints are 2.0 ns and the ratio of 1 for the 0.18 μm process, respectively. The slew and loading constraints are 1.0 ns and the ratio of 1.2 for the 0.13 μm process, respectively.

B. ECO on Industrial Projects

In Table I, we first report the comparison results on seven industrial projects. Column 1 lists the project name and its ECO version in parentheses. Columns 2–4 list the instance count, the adopted process technology, the spare-cell count, and the size of ECO in instances for each project, respectively. Columns 6 and 7 list the number of reported slew violations and loading violations, respectively, before any buffer-insertion scheme is applied. Columns 8–10, 12–14, and 16–19 list the worst input slew, the worst output-loading ratio to the library-suggested constraint, and the worst slack, respectively, reported (1) before any buffer-insertion scheme is applied (denoted by ori.), (2) after an EDA vendor's solution is applied (denoted by [15]), and (3) after MOESS is applied (denoted by MOESS). Columns 11, 15, and 19 also list the improvement of MOESS over [15] (denoted by imp.) in the worst input slew, the worse output-loading ratio, and the worst slack, respectively. The number followed by a "*" means that the corresponding value violates the constraint. In Columns 20–22 and 23–25, we report the number of spare buffers in use and the CPU runtime for both [15] and MOESS, and the corresponding improvement or speedup of MOESS over [15].

As the results show MOESS can resolve all the slew, loading, and setup-time violations for these seven projects while the vendor's solution violates the slew constraint in three projects, the loading constraint in two projects, and the setup time constraint in four projects. The average improvements of MOESS on the worst slew, worst loading, and worst slack are 24%, 21%, and 57%, respectively. Also, the number of used spare buffers by MOESS is smaller than that by [15] for each project, which saves more ECO resources for the next generation of ECO. This reduction to the number of used spare buffers is 38% in average. Furthermore, the runtime consumed by MOESS is less than that by [15] for each project as well. The average speedup of MOESS is 14.9×.

The above experimental results demonstrate the effectiveness and efficiency of our buffer-insertion algorithm.

C. Statistics of Inserted Buffers

Table II reports the detailed statistics of the inserted buffers used in Table I for both [15] and MOESS. In Table II, columns 2 and 3 list the number of buffers inserted. Columns 4 and 5 list the number of original violation gates remaining violated after the buffers are inserted. Columns 6 and 7 list the number of inserted buffers whose output loading exceeds its output available load (constraint violated). Columns 8 and 9 list the average ratio of the output loading over its output available load for all buffers not violating the constraint (i.e., buffers not in column 6 or 7).

As the result shows, the buffers inserted by MOESS can always satisfy the slew/loading constraint and further help to resolve the original violations. On the contrary, buffers inserted by [15] usually deviate from their ideal location, and hence may result in new violations on the inserted buffers or fail to resolve the original violations. Columns 10 and 11 further list the maximum and average distance between the ideal location and the final location of the buffers inserted by [15].

TABLE II
STATISTICS OF INSERTED BUFFERS USED IN TABLE I

1	2	3	4	5	6	7	8	9	10	11
Proj.	# of Inserted Buffers		Original Violations Unsolved		Violations on Inserted Buffers		Avg. Loading Ratio		Distance bw. Ideal & Real in [15]	
(Ver.)	[15]	MOESS	[15]	MOESS	[15]	MOESS	[15]	MOESS	Maximum	Average
da(3)	78	40	9	0	6	0	0.13	0.33	1036.8	445.15
db(3)	12	5	0	0	0	0	0.22	0.55	1159.6	504.21
dc(4)	91	68	43	0	5	0	0.11	0.27	919.4	450.86
dd(3)	83	48	13	0	3	0	0.12	0.39	769.8	349.93
de(2)	63	35	5	0	0	0	0.17	0.37	812.8	306.4
df(2)	377	277	40	0	7	1	0.18	0.42	1537.1	460.1
dg(4)	314	259	60	0	13	0	0.16	0.25	763.4	268.7
Average							0.16	0.37	999.8	397.9

One important reason why MOESS uses a smaller number of inserted buffers than [15] is that its ESB buffer-insertion scheme attempts to select the buffer which can use as much of its driving capability as possible to resolve the violation. At the same time, the output loading of the inserted buffer is also kept under the constraint. As the result shows in Table II, the buffers selected by MOESS utilize 37% of their driving capability in average, which is 2.3 times larger than that of [15] (16%). Also, the loading of all those buffers is kept under the constraint. This result again demonstrates the effectiveness of the proposed ESB scheme.

D. Solving Setup-Time Violations

Table III reports more information about the setup-time violations (collected by PrimeTime) for the same projects used in Table I. In Table III, columns 2 and 3 list the slack and the number of paths violating the setup-time constraint, respectively, after ECO is performed and before any buffer insertion scheme is applied. Then columns 4 to 9 list the same information after three buffer insertion schemes ([15], ESB, and ECT) are applied individually. As the result shows, after the proposed ESB scheme is applied, the total number of setup-time-violated paths drops from 686 to 36 for all projects. Note that the ESB buffer-insertion scheme only focuses on solving the slew/loading violations. Thus, this result first demonstrates that majority of the setup-time violations can be resolved when the slew/loading violations are resolved, which is why MOESS performs the ESB scheme before the ECT scheme. Next, the result also shows that the ESB scheme can resolve more setup-time violations than [15]. This is because the ESB scheme can resolve the slew/loading violations more effectively than [15] does. Last, for those inserted buffers located on a setup-time-violated path after the ESB scheme is applied, we remove the inserted buffers and apply the ECT scheme to insert new buffers for solving the slew/loading violations. Then the remaining 36 setup-time-violated paths can all be resolved, which again demonstrates the effectiveness of the proposed ECT scheme.

TABLE III
SLACK AND THE NUMBER OF PATHS VIOLATING SETUP-TIME CONSTRAINT BEFORE AND AFTER BUFFER INSERTION

1	2	3	4	5	6	7	8	9
Proj.	Before Buf. Insert		After [15]		After ESB		After ECT	
(Ver.)	Slack	Vio. Path	Slack	Vio. Path	Slack	Vio. Path	Slack	Vio. Path
da(3)	−3.5n	148	−2.2n	16	−0.1n	5	>0n	0
db(3)	>0n	0	>0n	0	>0n	0	>0n	0
dc(4)	−2.3n	266	−0.3n	42	−0.1n	18	>0n	0
dd(3)	−1.6n	34	>0n	0	>0n	0	>0n	0
de(2)	>0n	0	>0n	0	>0n	0	>0n	0
df(2)	−0.6n	172	−0.1n	54	−0.1n	3	>0n	0
dg(4)	−0.2n	66	−0.4n	14	−0.1n	10	>0n	0
total		686		126		36		0

IV. DETAILS OF WIRE-LOADING ESTIMATION

MOESS runtime efficiency mainly results from its quick wire-loading estimation of a multiple-terminal net, which is used in Section II-C for grouping the terminal pins to be driven by an inserted buffer, and in Section II-D for calculating the ideal buffer location. Since a net's wire loading is highly correlated to its wire length, to estimate its wire loading needs to estimate its final routing length first. The idea used in MOESS is to: 1) break a multiple-terminal net into several two-terminal nets; 2) take the routing length of a two-terminal net as its Manhattan distance times a ratio function of the routing congestion within the net's neighborhood; and 3) sum the estimated wire loading of those two-terminal nets. Such a ratio function can be obtained and tabulated in advance based on the statistics collected from the previous usage of the adopted APR tools. Thus, the wire loading of a multiple-terminal net can be calculated by a quick table lookup instead of actually performing the detail route and resistance and capacitance extraction. This quick estimation is cost-effective especially when determining which terminal pins should be driven by an inserted buffer, where different combinations of terminal pins would be tried iteratively for a violated high-fanout net. In the following sections, we will discuss several key issues about our wire-loading estimation.

A. Accuracy of Using Different Density Definitions

In MOESS, we use the routing-ratio function realtime radiation monitor device [$RRMD(vd)$] to represent the ratio of a two-terminal net's routing length over its Manhattan distance, where the via density vd is the input parameter representing the degree of routing congestion. To construct a tabular function of $RRMD(vd)$, we first divide the nets into different groups based on the value of their via density and then collect the distribution of the routing ratios within a group. Note that the collected statistics precludes the nets where obstacles are in between its terminals. This is because MOESS would detour the selection of the inserted buffers along the boundaries of obstacles when obstacles exist in between.

For large designs in advanced logic processes, six or more metal layers are usually used and each layer (more

TABLE IV
STATISTICS OF $RRMD(vd)$ WHEN USING THE AVERAGE PERCENTAGE OF OCCUPIED VIA AMONG ALL LAYERS AS THE DEFINITION OF vd

Avg Via Density	0–0.1	0.1–0.2	0.2–0.3	0.3–0.4	0.4–0.5
Probability (%)	68.6	26.8	3.1	0.9	0.5
Mean	1.127	1.176	1.221	1.236	1.298
Std. Dev._de	0.132	0.194	0.232	0.216	0.278
95th percentile	1.38	1.53	1.68	1.66	1.82
99th percentile	1.62	1.88	2.17	2.04	2.41

TABLE V
STATISTICS OF $RRMD(vd)$ WHEN USING THE MAXIMUM PERCENTAGE OF OCCUPIED VIA AMONG ALL LAYERS AS THE DEFINITION OF vd

Worst Via Density	0–0.1	0.1–0.2	0.2–0.3	0.3–0.4	0.4–0.5
Probability (%)	19.5	56.2	16.2	4.5	3.6
Mean	1.126	1.129	1.19	1.214	1.213
Std. dev._de	0.122	0.139	0.199	0.230	0.233
95th percentile	1.36	1.39	1.55	1.63	1.67
99th percentile	1.58	1.65	1.89	2.11	2.13

TABLE VI
AVERAGE RATIO OF THE ESTIMATED ROUTING LENGTH OVER THE REAL ROUTING LENGTH FOR THE OUTPUT NETS OF THE INSERTED BUFFERS IN TABLE I

# of terminals	2	3	4	5–8	9–12	13–16	>17	Total
# of nets	95	118	113	238	81	33	55	732
Avg. ratio	1.203	1.271	1.336	1.333	1.330	1. 282	1.346	1.304
Nets<ratio 1	1	0	0	0	0	0	0	1

specifically, between each two adjacent metal layers) has its own percentage of occupied via. In the following experiment, we use two definitions to calculate the via density vd: 1) the average percentage of occupied via on a layer; and 2) the maximum percentage of occupied via among all layers. Tables IV and V list the probability, mean, standard deviation, 95th percentile, and the 99th percentile of $RRMD(vd)$ for each interval of vd when the first and second definitions of via density are used, respectively. As the result shows, the standard deviation of most intervals obtained by using the second definition is smaller than that by using the first one. It implies that using the maximum via percentage as the parameter of $RRMD(vd)$ can lead to a narrower distribution and hence a more accurate estimation of $RRMD(vd)$. Thus, *MOESS* uses the second definition of the via density as the input parameter for $RRMD(vd)$.

B. Real Routing Length Versus Estimation

One main objective of our wire-loading estimation is to obtain an available spare buffer whose output loading can be under the constraint when being inserted to resolve the violation. Thus, when mapping a via density vd to its $RRMD(vd)$, we return the 95th percentile of the distribution associated with the interval of vd, not the mean. This 95th percentile of routing ratio often leads to a conservative estimation of the wire loading, such that we may not be able to find the best buffer whose driving capability can be fully utilized to resolve the violation. Also, for a multiple-fanout net, our estimation will break the net into several two-terminal nets and then sum the loading of each, which is again a conservative estimation since the real routing pattern of a multiple-fanout net is like a Steiner tree instead of a piece-wise connected chain.

Table VI lists the number of nets, average ratio of the routing length estimated by MOESS over the real routing length, and the number of nets whose above ratio is smaller than 1 for different numbers of net's terminals. The nets used in this table are the output nets of the inserted buffer in Table I. As the result shows, only 1 out of total 733 nets has its real routing length larger than our estimation. Actually, that net does not result in a loading violation since usually the selected buffer seldom locates on the boundary of the search area. Also, for a net with more terminals, the estimated routing length usually tends to be more conservative due to the difference between a Steiner tree and a piece-wise connected chain. Therefore, the experimental result demonstrates that the loading of the selected spare buffers can be controlled under the constraint while an average 30% margin is imposed onto the estimation.

V. CONCLUSION

In this paper, we fist proposed an efficient and effective framework to resolve the slew, loading, and timing constraint in metal-only ECO. Also, we discussed the accuracy and limitation of using the via density to predict the wire loading of an inserted connection. The experimental results obtained from real industrial projects showed that the proposed framework can significantly increase affordable scale of a metal-only ECO with fewer spare gates and less runtime in use, compared to a current vendor's solution.

REFERENCES

[1] International Technology Roadmap for Semiconductors [Online]. Available: http://www.itrs.net/
[2] A. Balasinski, "Optimization of sub-100-nm designs for mask cost reduction," *J. Microlithography Microfabrication Microsyst.*, vol. 3, pp. 322–331, Apr. 2004.
[3] J. Cong, J. Fang, and K. Khoo, "An implicit connection graph maze routing algorithm for ECO routing," in *Proc. Assoc. Comput. Machinery/IEEE Int. Conf. Comput. Aided Design*, Nov. 1999, pp. 163–167.
[4] Y. L. Li, J. Y. Li, and W. B. Chen, "An efficient tile-based ECO router using routing graph reduction and enhanced global routing flow," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 345–358, Feb. 2007.
[5] S. Dutt and H. Arslan, "Efficient timing-driven incremental routing for very-large-scale integration ciruits using distributed file system and localized slack-satisfaction computations," *Design Autom. Test Eur.*, vol. 1, pp. 768–773, Mar. 2006.
[6] W. Shi and Z. Li, "Fast algorithm for optimal buffer insertion," *IEEE Trans. Comput. Aided Design*, vol. 22, no. 4, pp. 492–498, Apr. 2005.
[7] P. J. Osler, "Placement driven synthesis case studies on two sets of two chips: Hierarchical and flat," in *Proc. Assoc. Comput. Machinery Int. Symp. Physical Design*, 2004, pp. 190–197.
[8] C. Alpert, A. Kahng, B. Liu, I. Mandoiu, and A. Zelikovshy, "Minimum-buffered routing of non-critical nets for slew rate and reliability control," in *Proc. Assoc. Comput. Machinery/IEEE Int. Conf. Comput. Aided Design*, 2001, pp. 408–415.
[9] P. Saxena, N. Menezes, P. Cocchini, and D. A. Kirkpatrick, "Repeater scaling and its impact on computer aided design ," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 23, no. 4, pp. 451–463, Apr. 2004.
[10] S. Hu, C. J. Alpert, J. Hu, S. K. Karandikar, Z. Li, W. Shi, and C. Sze, "Fast algorithm for slew-constrained minimum cost buffering," in *Proc. Design Autom. Conf.*, 2006, pp. 308–313.
[11] Y. P. Chen, J. W. Fang, and Y. W. Chang, "ECO timing optimization using spare cells and technology remapping," in *Proc. Assoc. Comput. Machinery/IEEE Int. Conf. Comput. Aided Design*, 2007, pp. 530–535.
[12] Y. M. Kuo, Y. T. Chang, S. C. Chang, and M. Marek-Sadowska, "Engineering change using spare cells with constant insertion," in *Proc.*

Assoc. Comput. Machinery/IEEE Int. Conf. Comput. Aided Design, 2007, pp. 544–547.

[13] C. P. Lu, M. Chao, C. H. Lo, and C. W. Chang, "A metal-only-ECO solver for input-slew and output-loading violations," in *Proc. Assoc. Comput. Machinery Int. Symp. Physical Design*, 2009, pp. 191–198.

[14] S. Sait and H. Youssef, *VLSI Physical Design Automation: Theory and Practice*, vol. 6. Singapore: World Scientific, 1999, pp. 165–167.

[15] Magma Design Automation. (2008 Oct. 4, 23:17:45). Blast Fusion, mantle version 2005.03.168-linux24_x86_64.

Routing With Constraints for Post-Grid Clock Distribution in Microprocessors

Rupesh S. Shelar, *Member, IEEE*

***Abstract*—Microprocessors typically employ a global grid followed by block-level buffered trees for clock distribution. The trees are connected to the grid by routing wires along reserved tracks. The routing of these clock wires, which present load to the grid, is constrained by delay/slope requirements at inputs of the block-level trees. This leads to a capacitance minimization problem during multiterminal routing, where routes use the reserved tracks and obey the constraints. This paper presents an algorithm that addresses the problem, improving wirelength by 14% over a competitive approach. The algorithm is employed for post-grid clock distribution in a 45 nm technology microprocessor.**

***Index Terms*—Design aids, integrated circuits, layout, microprocessors, placement, routing.**

I. Introduction

In high-performance microprocessor circuits, clocks are distributed employing hybrid networks, containing global grid followed by buffered gated trees, to reduce the skew [1], [2], [3]. These circuits are designed hierarchically by partitioning the chip into different layout areas, each containing several blocks comprising tens of thousands of standard cells, macros, etc. The buffered gated clock trees are created inside each block to distribute the clock to sequentials. The input to the clock trees is provided by routing clock wires, on reserved tracks, from the global grid. This routing is referred to as the post-grid clock distribution or the global clock routing in this paper.

The post-grid clock distribution problem poses a challenge because of the problem size and constraints on routing without the luxury of buffer insertion on the routes. Typically, there are thousands of clock terminals in each layout area. The constraints include delay/slope requirements at inputs of the clock trees and a limit on the load presented to grid drivers. Traditionally, this routing is carried out manually, and possibly iteratively to meet the slope requirements or to reduce the load on the grid. This may affect the time to market, as the block-level and full-chip timing convergence depends on the clock arrival times at the sequentials over entire die area. Moreover, commonly used techniques to converge the block-level timing such as up-sizing the sequentials eventually result in increased loads on the grid drivers. If the load is too high, the clock may not toggle with the desired transition times. However, it is difficult to predict whether the block-level timing convergence has indeed led to such a situation, since the exact load depends on clock ports in other blocks in the layout area and is not known until global clock routing is carried out. This motivates a search for a fast algorithm for post-grid clock distribution. The clock wires also contribute significant load on the grid. Reducing the load often leads to improved reliability of the grid and power savings. This motivates a quest for the algorithm that can minimize the wire capacitance.

To the best of our knowledge, there is no published algorithm addressing the problem of multiterminal routing with constraints in the context of post-grid global clock distribution. Microprocessor designers carry out this routing mostly manually, often employing the nearest source heuristic (explained in Section IV). The heuristic may result in sub-optimal wirelength and may also cause the poor slopes at the inputs of the local clock trees. The work in [4] proposes shorting the inputs of the buffers every two to three stages in a buffered H-tree so that per stage skew attenuates with the number of stages. Recent link-insertion algorithms to reduce skew [5], [6], [7] in buffered clock trees are similar. Other related work includes that on zero-skew routing [8], [9], [10], or that on buffered clock tree synthesis [11]; these focus on wirelength/skew/power minimization for (buffered) trees.

We propose a polynomial time algorithm to solve the routing problem with slew constraints in the context of post-grid global clock distribution. The algorithm is practical, as it runs in seconds on real microprocessor design. The resulting routing obeys the constraints and improves wirelength by 14%, on average, over commonly used nearest source heuristic. The algorithm has been employed to carry out routing for post-grid global clock distribution in a high-end microprocessor design in a 45 nm technology, demonstrating its practical use. The algorithm may also be useful for application specific integrated circuits (ASICs) running at GHz frequencies using hybrid clock distribution schemes.

The rest of the paper is organized as follows. Section II introduces preliminaries. Section III describes a graph theoretic formulation for multiterminal routing with constraints underlying the post-grid clock distribution problem. Section IV describes algorithms that address the problem. Section V discusses results obtained employing those for clock distribution in a microprocessor. Section VI concludes the paper.

II. Preliminaries

High-end microprocessors employ the hybrid network as shown in Fig. 1 to distribute the clock. The phased-locked loop (PLL) generates the clock and drives the global grid, as shown in Fig. 1(a). The grid, implemented using spines [2], distributes the clock to block-level buffer trees. The PLL and the grid are designed manually. The last level in the grid comprises wires, as shown in Fig. 1(b). These wires, referred to as the global grid wires, are driven by multiple

Manuscript received June 16, 2009; revised August 25, 2009. Current version published January 22, 2010. This paper was recommended by Associate Editor P. Saxena.

The author is with the Technology and Manufacturing Group, Intel Corporation, Hillsboro, OR 97124 USA (e-mail: rupesh.s.shelar@intel.com).

Digital Object Identifier 10.1109/TCAD.2009.2040012