Interconnect Bundle Sizing under Discrete Design Rules

Konstantin Moiseev, *Student Member, IEEE,*
Avinoam Kolodny, *Member, IEEE,* and Shmuel Wimer

***Abstract*—The lithography used for 32 nm and smaller very large scale integrated process technologies restricts the admissible interconnect widths and spaces to a small set of discrete values with some interdependencies, so that traditional interconnect sizing by continuous-variable optimization techniques becomes impossible. We present a dynamic programming (DP) algorithm for simultaneous sizing and spacing of all wires in interconnect bundles, yielding the optimal power–delay tradeoff curve. DP algorithm sets the width and spacing of all interconnects simultaneously, thus finding the global optimum. The DP algorithm is generic and can handle a variety of power–delay objectives, such as total power or delay, or weighted sum of both, power–delay product, max delay, and alike. The algorithm consistently yields 6% dynamic power and 5% delay reduction for interconnect channels in industrial microprocessor blocks designed in 32 nm process technology, when applied as a post-layout optimization step to redistribute wires within interconnect channels of fixed width, without changing the area of the original layout.**

***Index Terms*—Dynamic programming (DP), gridded design rules, interconnect sizing and spacing, power–delay optimization.**

I. Introduction and Motivation

Power and performance of very large scale integrated (VLSI) systems and their tradeoff are important design considerations in state-of-the-art technology [1]. Power and speed are often in conflict and their tradeoff is challenging, offering opportunities for new design methods and algorithms targeting simultaneous power and delay reduction. Charging and discharging of interconnect capacitances are dominant components of circuit delay and power [6], and their contribution increases in 32 nm and more advanced process technologies.

This paper addresses the problem of power and delay reduction in interconnect channels, also known as wire bundles (bus-like structures of parallel wires, see Fig. 1), under discrete-size design rules. Our objective is to vary the wire widths and the inter-wire spaces in the channel while keeping a fixed total width of the structure, to achieve the optimal power–delay tradeoff curve. At each point on this curve, we obtain the minimum interconnect power for a given delay and vice versa. Simultaneous wire sizing and spacing is effective because wire-to-wire capacitances, which are the dominant part of interconnect capacitance [6], are very sensitive to inter-wire spacing.

Several interconnect resizing algorithms were proposed to increase clock frequency [2], [3], to reduce dynamic power [4], and to maintain some tradeoff between both [5]. Most of prior work on interconnect resizing for power and delay reduction [2]–[5] assumed that interconnect width and spacing can vary in a continuous range allowed by design rules. Modern manufacturing process technologies restrict the admissible width and space of interconnect to very few discrete values. Design and optimization under such restrictions is a challenge.

As compared to previous works in this field, this paper presents a new combination of several approaches described in the literature. First, we employ a global optimization technique, optimizing many nets simultaneously. Single-net optimization is blind to other adjacent nets, hence ignoring the cross-capacitance between nets, thus yielding sub-optimal results. Moreover, single-net optimization cannot account for the area resource available at the block level. Second, the proposed technique optimizes both power and delay simultaneously, which helps to understand tradeoffs between them and to derive practical implications. Third, instead of a single optimal solution the algorithm computes the optimal power–delay tradeoff curve which reveals the full design space. And finally, as to our knowledge, this is the first work which presents a multi-net interconnect optimization technique under discrete design rules.

The allocation of wire widths and spaces from a set of discrete admissible values is a non-deterministic polynomial time (NP)-complete problem [10] and naturally mapped into sequential decision making, for which a dynamic programming algorithm is very useful. The development of the algorithm, the proof of its optimality and its implementation for VLSI interconnects are the main contributions of this paper, the rest of which is organized as follows. In Section II, interconnects and process technology models with their delay and power equations are presented. The dynamic programming (DP) solution is derived in Section III. Results and experiments obtained for real industrial design in 32 nm process technology are presented in Section IV.

II. Delay and Power Modeling of Interconnects in a Bundle

The interconnecting wires at high-metal layers typically run in alternating orthogonal directions. In 32 nm and below, jogs in layers higher than metal 1 are rarely used and can be ignored for any practical optimization. Shifting wires in one layer does not affect spacing/width of the orthogonal wires in the layers above it and below it. The lengths of wires in layers above and below the optimized layer usually reach hundreds of micrometers, while the typical wire shift during the optimization in a given layer is less than a micrometer. Thus, lengths of wires in the adjacent layers usually change by less than 1%. The statistical average of these small changes is zero, such that these variations are negligible for all practical cases. Yet, wire shifts may cause design rule violations in via landing and coverage. In the industrial environment, where wire spacing described herein was successfully employed, the resulting violations were manually fixed by mask designers within reasonable effort [5].

Let $\sigma_1, \ldots, \sigma_n$ be n signals of a wire bundle, and let $I_1, \ldots, I_n$ be their corresponding wires positioned between two shielding wires I_0 and I_{n+1} connected to ground, as shown in Fig. 1. Let $w_1, \ldots, w_n$ be wire widths and $s_0, \ldots, s_n$ be the spaces between them. It is assumed that admissible wire

Manuscript received October 7, 2009; revised December 28, 2009 and March 6, 2010. Date of current version September 22, 2010. This paper was recommended by Associate Editor J. Hu.

K. Moiseev and A. Kolodny are with the Technion-Israel Institute of Technology, Haifa 32000, Israel (e-mail: kostya.moiseev@gmail.com; kolodny@ee.technion.ac.il).

S. Wimer is with the Engineering School, Bar-Ilan University, Ramat-Gan 52900, Israel (e-mail: wimers@macs.biu.ac.il).

Digital Object Identifier 10.1109/TCAD.2010.2051633

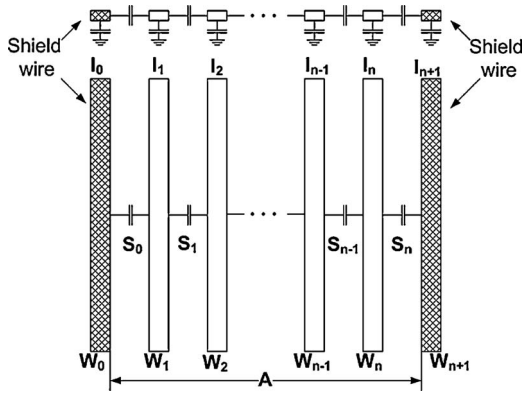

Fig. 1. Fundamental cross-coupling and ground capacitance model. Wires run in parallel and the entire bundle is shielded on both sides by wires connected to ground.

widths and spaces are taken from finite sets, whose cardinality is usually very small, representing gridded design rules

$$w_i \in \mathbf{W} = \{W_1, \ldots, W_q\} \quad \text{and} \quad s_i \in \mathbf{S} = \{S_1, \ldots, S_p\}. \quad (2.1)$$

Sometimes, a mix of discrete values with continuous ranges is allowed, but design practice usually employs only a limited set of values, turning the problem into pure discrete. Lithography may sometimes prohibit certain width and space combinations by imposing interdependencies between the values in (2.1). We'll ignore such restrictions as those do not affect the complexity of the problems and the optimality of the proposed solutions. The area allocated for the wire bundle dictates a total width limit A

$$\sum_{i=1}^{n} w_i + \sum_{i=0}^{n} s_i \le A. \quad (2.2)$$

Let's denote by $D_i(s_{i-1}, w_i, s_i)$ and $P_i(s_{i-1}, w_i, s_i)$ delay and power of signal σ_i correspondingly. The details of power and delay modeling of the wire bundle are standard and appear in [5].

The total sum of delays, maximal delay and total interconnect power consumption are given respectively by

$$D^{\text{sum}}(\bar{s}, \bar{w}) = \sum_{i=1}^{n} D_i(s_{i-1}, w_i, s_i) \quad (2.3)$$

$$D^{\max}(\bar{s}, \bar{w}) = \max_{1 \le i \le n} D_i(s_{i-1}, w_i, s_i) \quad (2.4)$$

$$P(\bar{s}, \bar{w}) = \sum_{i=1}^{n} P_i(s_{i-1}, w_i, s_i). \quad (2.5)$$

Let T_i be the required time of σ_i and $\Delta_i = T_i - D_i$ be its slack. It was shown in [5] that maximizing $\sum_{i=1}^{n} \Delta_i$ is equivalent to minimizing $\sum_{i=1}^{n} D_i$, and has the same solution. It was also shown that maximizing the minimal Δ_i is a similar problem to minimizing the maximal D_i, since both are convex and the same algorithm will solve both. Hence, without loss of generality we'll consider just delays in the derivation.

III. Discrete Width and Space Allocation in an Interconnect Bundle

The computational model of the DP algorithm for the bundle shown in Fig. 1 is presented below. We show that it finds all the optimal power–delay combinations, and analyze its complexity.

A. Size Allocation as a Sequential Decision Problem

The total width A of the bundle in Fig. 1 is a *resource* being *allocated* to the space and width alternating sequence $\omega : (w_0, s_0, w_1, s_1, \ldots, w_n, s_n)$. For the sake of convenience, an artificial width $w_0 = 0$ is introduced.

Sequence ω needs to satisfy (2.1). It is assumed that there exists at least one feasible allocation satisfying

$$\sum_{i=1}^{n} w_i + \sum_{i=0}^{n} s_i = A. \quad (3.1)$$

For a subsequence $(w_0, s_0, w_1, s_1, \ldots w_j, s_j) \subset \omega$, we define

$$D^{\text{sum}}_{0..j} = \sum_{i=1}^{j} D_i(s_{i-1}, w_i, s_i) \quad (3.2)$$

$$D^{\max}_{0..j}(0, j) = \max_{1 \le i \le j} D_i(s_{i-1}, w_i, s_i) \quad (3.3)$$

$$P_{0..j} = \sum_{i=1}^{j} P_i(s_{i-1}, w_i, s_i). \quad (3.4)$$

Equations (2.3), (2.4), and (2.5) can be calculated incrementally by (3.2), (3.3), and (3.4), respectively.

Accumulated sum of delays in (3.2) and max delay in (3.3) are similar in terms of being monotonic, and independent of their previous calculation. Denoting the operations + or max by $\oplus$, we obtain delay and power expressions that get updated step-by-step as follows:

$$D_{0..j} = D_{0..j-1} \oplus D_j(s_{j-1}, w_j, s_j) \quad (3.5)$$

$$P_{0..j} = P_{0..j-1} + P_j(s_{j-1}, w_j, s_j). \quad (3.6)$$

At the final step $j = n$, the objectives (2.3), (2.4), and (2.5) are completely defined.

The objective functions satisfy the following properties.

Property 1: The functions in (3.5) and (3.6) are monotonic non-decreasing in allocation step j.

Property 2: For any $1 \le j \le n - 1$

$$D_{0..n} = D_{0..j} \oplus D_{j+1..n} \quad (3.7)$$

$$P_{0..n} = P_{0..j} + P_{j+1..n}. \quad (3.8)$$

Property 3: After the first j allocations are done, optimization of the rest $n + 1 - j$ allocations depends only on s_j and $A_{j..n} = A - \left(\sum_{i=0}^{j} w_i + \sum_{i=0}^{j} s_i\right)$ which is available for the rest $n + 1 - j$ wires, and its optimization is *independent* of how the first j allocation decisions have been made.

Let Ω be the set of all possible allocations and define a partial order as follows.

Definition 1 (Dominancy): Allocation ω':$(w'_0, s'_0, \ldots, w'_j, s'_j) \in \Omega$ *dominates* allocation $\omega'' : (w''_0, s''_0, \ldots, w''_j, s''_j) \in \Omega$ if:

1) $A - \left(\sum_{i=0}^{j} s'_i + \sum_{i=0}^{j} w'_i\right) \ge A - \left(\sum_{i=0}^{j} s''_i + \sum_{i=0}^{j} w''_i\right)$;
2) $s'_j \ge s''_j$;
3) $D_{0..j}(\omega') \le D_{0..j}(\omega'') \wedge P_{0..j}(\omega') \le P_{0..j}(\omega'')$.

It follows that ω'' cannot yield a better solution than ω' does and can therefore be safely dropped from any further consideration of the optimal solution. Sequence ω'' is thus *redundant*.

It follows that for every pair of $A_{j..n}$ and s_j there is a set of *non-redundant* $\left\{\left[P_k\left(A_{j..n}, s_j\right), D_k\left(A_{j..n}, s_j\right)\right]\right\}_k$ power–delay pairs. Therefore, the triplet $\left\langle A_{j..n}, s_j, \left[P_k\left(A_{j..n}, s_j\right), D_k\left(A_{j..n}, s_j\right)\right]\right\rangle$ fully characterizes the first j allocations with their resultant power and delay, and is the only information required to yield the optimal allocation of all n wires. We encode such a triplet in a so called *state* defined as follows.

Definition 2 (State): A triplet $\left\langle A_{j..n}, s_j, \left[D_k\left(A_{j..n}, s_j\right), P_k\left(A_{j..n}, s_j\right)\right]\right\rangle$ is called *state*.

A state is *feasible* if $A_{j..n} \geq 0$. It follows by definition that $A_{n..n} = 0$ (all area is consumed). A *stage* Λ_j is the set of all feasible non-redundant states obtained by all possible size allocations of the first j wires. The states of a stage are totally ordered by lexicographic comparison of their A, s, and P. Such order is important for efficient insertion, deletion, and redundancy check of states in a stage, allowing to access states in logarithmic time, by using an appropriate data structure. It follows from non-redundancy that ordering by P implies reverse order by D.

B. State Augmentation and Satisfaction of Optimality

Size allocation proceeds from I_j to I_{j+1} as follows. Stage Λ_{j+1} is initially empty. Every state of Λ_j is attempted for augmentation by every possible width and space pair (w, s) satisfying (2.1). Only feasible augmentations satisfying $A_{j+1..n} = A_{j..n} - (w + s) \geq 0$ are considered and a new state $\left\langle A_{j+1..n}, s, \left[D\left(A_{j+1..n}, s\right), P\left(A_{j+1..n}, s\right)\right]\right\rangle$ is thus defined.

If no state with the pair $A_{j+1..n}$ and s exists yet in Λ_{j+1} a new state is added to Λ_{j+1}. Otherwise, if it is found to dominate an already existing state of Λ_{j+1}, the latter is deleted, while a new one is added. If it is found redundant then it is ignored. In this way, Λ_{j+1} is built incrementally and maintains only non-redundant states, until all state augmentation of Λ_j are consumed.

Theorem 1 (Optimality): Stage Λ_n of the DP algorithm contains all the feasible non-redundant, and hence optimal, power–delay pairs that can be obtained by any width and space allocation to n wires.

The proof of the theorem is presented in [10].

Knowing that the DP algorithm yields all power–delay non-redundant pairs, they define the power–delay envelope of the bundle. One can plot the power–delay curve as shown in Fig. 2. This curve is (by definition of dominancy) monotonic increasing in one parameter and monotonic decreasing in the other. The curve divides the first quadrant of the power–delay plane into an upper-right region where all feasible power–delay solution exist and a lower-left region where no feasible solution exists. This envelope has the same nature of the well known shape-function in bottom-up buffer insertion and wire resizing algorithms [7].

C. Time and Memory Bounds of the DP Algorithm

Let $P_{\max}$ and $D_{\max}$ be the maximal power and delay, respectively, incurred by a wire in the problem setting. We define power and delay resolutions $\varepsilon P_{\max}$ and $\varepsilon D_{\max}$, respectively, where $\varepsilon \ll 1$ is an arbitrary small accuracy parameter. We then snap every calculated power and delay to the nearest integral multiple of the corresponding resolution. Such rounding is essential for combinatorial tractability. Values of power and delay will then be restricted to $\{k\varepsilon P_{\max} \mid 1 \leq k \leq n/\varepsilon\}$ and $\{k\varepsilon D_{\max} \mid 1 \leq k \leq n/\varepsilon\}$, whose size is n/ε.

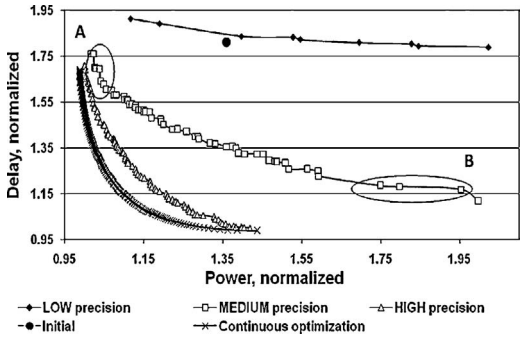

Fig. 2. Power–delay curves for different optimization precisions with entire power–delay envelope for typical metal 3 bundle of 14 wires.

Then the following theorem defines time and storage bounds of the DP algorithm.

Theorem 2 (Time and Storage Bounds): Given an n-signal wire bundle and a process technology having p admissible widths and q admissible spaces, the time complexity of the DP algorithm to find width and space allocation yielding the optimal power–delay curve in accuracy ε is bounded by $O\left(pq^2n^3/\varepsilon \log(n/\varepsilon)\right)$. The storage is bounded by $O\left(qn^3/\varepsilon\right)$.

The proof of the theorem is presented in [10].

IV. Implementation and Experimental Results

The DP algorithm described in Section III was coded in C++ under OpenAccess data model. It was applied to industrial layout blocks from a commercial full-custom processor design in 32 nm process technology, generated by standard tools. Our algorithm was employed in an attempt to reduce the delay and dynamic power. The wire bundles were defined by power and clock wires, which were not allowed to move, such that the optimization preserved the original size of the whole layout.

In the following, X denotes the minimum allowed wire width and minimum allowed inter-wire space $W_{\min} = S_{\min} = X$. Maximum allowed width and space in the given technology are $3X$.

We first experimented with four different wire bundles. Prior to applying the DP algorithm, continuous optimization was performed where wire widths and inter-wire spaces were allowed to take any value in the continuous range $[X, 3X]$. Minimization of average wire delay (total sum divided by number of wires) obtained the value $D_{\min}$ used later in normalizing delays for comparisons of results. Similarly, continuous power minimization yielded the value $P_{\min}$ used later for normalization.

After continuous optimization, we explored the tradeoff between run-time and optimality, where each bundle was optimized three times with the DP algorithm using only realistic discrete sizes. Every experiment used a different setting of allowable sizes. The first set restricted allowable sizes to $\{X, 3X\}$, the second experiment allowed $\{X, 2X, 3X\}$, and a final experiment allowed the complete set $\{X, 1.5X, 2X, 2.5X, 3X\}$ of sizes provided by the 32 nm

TABLE I
OPTIMIZATION RESULTS FOR WIRE BUNDLES DERIVED FROM INDUSTRIAL LAYOUT

Bundle n		Bundle Width, X	Bundle Length, X	Metal Layer	Run-Time (s)					Number of States					Average Diff. From the Continuous Solution (%)				
					L	M	H	C	I	L	M	H	C	I	L	M	H	C	I
1	18	90	2115	3	0.87	6.56	359.03	3182.22	–	25	89	202	373	1	27.75	5.24	1.15	0	25.5
2	10	40	4007	2	0.21	0.91	29.23	2405.26	–	16	23	97	1006	1	58.69	6.62	1.36	0	88.5
3	14	38	1752	3	0.27	2.02	17.63	2242.49	–	9	56	95	277	1	84.61	37.23	5.41	0	58.7
4	12	40	3298	4	0.14	0.95	37.28	934.5	–	3	7	50	170	1	34.30	10.11	3.86	0	28.4

TABLE II
MAXIMIZATION OF MINIMUM SLACK

Wire Index	1	2	3	4	5	6	7	8
Required time	1.00	0.88	0.56	0.51	0.66	0.66	0.70	0.63
Delay before	1.00	0.88	0.56	0.62	0.60	0.59	0.67	0.53
Delay after	1.00	0.84	0.47	0.50	0.58	0.62	0.62	0.55
Slack before	0	0	0	**−0.12**	0.07	0.08	0.03	0.11
Slack after	0	0.03	0.09	0.02	0.08	0.05	0.08	0.08

process technology. For each run, we generated the full set of Pareto optimal power–delay pairs. For one of the bundles, these sets are shown in Fig. 2. The corresponding runs are denoted as "low," "medium," and "high" precision accordingly. The original power and delay calculated for the non-optimized layout is also shown. A power–delay curve obtained by continuous power minimization, where delays were constrained in small steps, starting from $D_{\min}$, is also plotted. It provides an idea of how much design optimality is lost by allowing very few sizes, a very important tradeoff between yield, manufacturing cost and performance. For each value of delay constraint, the resulting minimal power is plotted. The results are summarized in Table I. "L," "M," "H," "C," and "I" columns show results for low, medium, high-precision runs, continuous optimization and initial state correspondingly. The continuous minimization points are called "states" just for convenience. Fig. 2 shows clearly that wire sizes in the original layout as allocated by the commercial tool yielded power and delay far from optimal. It indicates that there is an opportunity for improvement of both power and delay by considering appropriate wire widths and spacing in conjunction with the primary goal of routing completion. The plots also demonstrate that usage of only three sizes sacrifices a lot of optimality, while using five values from X to $3X$ yields optimal performance within about 1-5% difference compared to continuous optimization.

In the vicinity of $D_{\min}$ and $P_{\min}$, the sensitivity to one of the optimization objectives is high, while the sensitivity to the second is low. For example, let's look at the two areas emphasized in Fig. 2. In area A, there are two solutions with very high-delay sensitivity, while in area B the situation is the opposite: there are two solutions with very high-power sensitivity. Thus, tuning a design to one of the corners is quite inefficient: slight improvement in one of the objectives causes a great loss in the other. From the design point of view, the best solution should be located near the middle of the power–delay curve (as close as possible to the origin).

Another experiment demonstrates how DP solves the minimum-slack maximization problem. Half of bundle 1 of Table I is used, and required times were assigned such that one of the signals had a large negative slack and all others had non-negative slacks. Using the min-slack objective function in

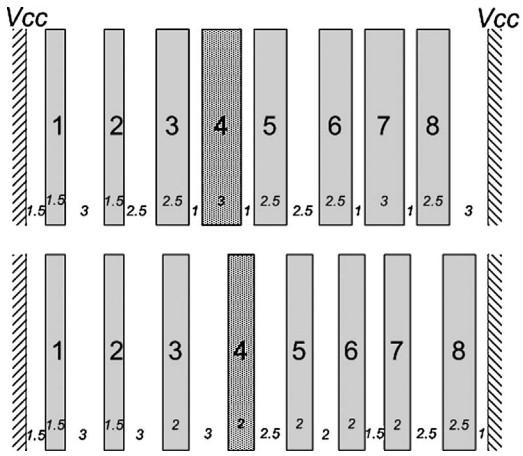

Fig. 3. Bundle cross-section for power-slack optimization before (top) and after (bottom) algorithm run. The slack of critical wire (4) was improved by decreasing wire width and allocating more space around. The values of widths and spaces in X are written at the bottom part of wires.

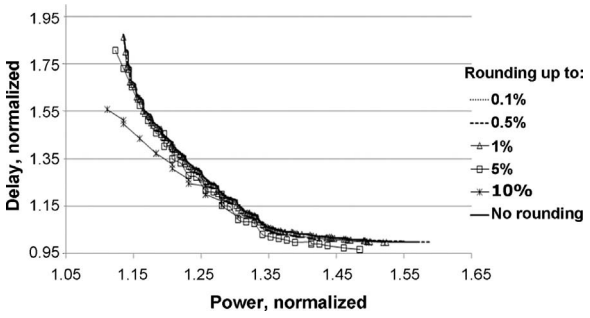

Fig. 4. Optimization with rounding of power–delay pairs: rounding to up to 5% does almost not affect optimization results; rounding up to 10% results in a error of 2.74%.

conjunction with total power, DP succeeded to eliminate the negative slack. Similar to the total delay case, DP calculated the optimal tradeoff curve of power versus min-slack. The results are summarized in Table II, where all delays and slacks are normalized with respect to the maximal wire delay. Having this curve at hand, the end point corresponding to maximal min-slack has been chosen. Obviously, maximal min-slack is paired with maximal power, but other pairs along the curve could be selected for different tradeoffs. As shown in the table, DP changed the initial worst slack from −0.12 to 0. The changes of wire widths and spaces are illustrated in Fig. 3, which readily shows that min-slack maximization decreased wire widths in favor of allocating larger spaces around the critical wire. Notice, that the run-time in case of slack optimization is about ten times faster than for total sum of delays. This is explained by the far smaller sets of possible slack values, as compared to total sum of delays.

TABLE III
OPTIMIZATION WITH ROUNDING OF POWER−DELAY PAIRS

Rounding Resolution	Difference From Highest Resolution Curve (%)	Run Time (s)	Run-Time Improvement
Without rounding	0	410	1X
0.1%	0.18	376.13	1.1X
0.5%	0.51	292.94	1.4X
1%	1.25	194.59	2.1X
5%	0.7	32.86	12.5X
10%	2.74	10.96	37.4X

TABLE IV
RESULTS OF POWER−DELAY MINIMIZATION IN REAL BLOCKS

	Block 1		Block 2	
Size (micrometers)	69 × 68		101 × 150	
Metal layer	2	4	2	4
Initial power	349.1	240.6	622	886.3
$P_{\min}$	333	222.1	598	802.9
$P_{\max}$	527	347.1	956.3	1238
Power reduction (%)	**4.6**	**7.7**	**3.9**	**9.4**
Initial delay	5201	3880	8094	10 635
$D(P_{\max})$	5040	3633	7903	9802
$D(P_{\min})$	6491	4614	10 538	13 159
Delay reduction (%)	**3.1**	**6.4**	**2.4**	**7.8**

In the next experiment, we targeted the algorithm's run-time issue. According to Theorem 2, the run-time of the algorithm is inversely proportional to power (delay) resolution ε. We invoked the algorithm on bundle 1 from Table I with different values of ε and measured distances of the obtained solutions from the solution with the highest resolution. The obtained power–delay curves are presented at Fig. 4 and the results are summarized in Table III. The rounding resolution was measured as a percentage of the maximum power/delay values for the given bundle. As can be seen from the table, the runtime can be improved by up to a factor of 12 as compared to highest resolution curve while the power and delay values differ only by 1.25% from the exact values. It can be seen from Fig. 4 that the sets of points corresponding to the solutions with rounding are lie systematically below the curve representing non-rounded solutions. It can be explained by the fact that, although the rounding is performed to the nearest possible value, the algorithm consistently chooses solutions with smaller values of delay/power, so that the solutions which were rounded upward were most likely pruned.

Finally, we present results obtained on real design blocks used in a full-custom processor design in 32 nm process technology with placement and routing performed by a commonly used commercial vendor tool. As a follow-up of our research we heuristically extended the algorithm presented in this paper to arbitrary layout (treating bundles of parallel wires which are not all of the same length).

Table IV presents the simultaneous power and delay results obtained for two typical blocks. These blocks use metal2, metal3 and metal4 for interconnections. The results are shown in relative units. As shown in the table, a significant simultaneous reduction of power and delay has been achieved. This is explained by the fact that commercial tools, although guiding the place and route for power–delay optimization by controlling the position of cells and specifying width and space for critical signals, do not perform global sizing optimization, which our algorithm does.

V. CONCLUSION AND FURTHER RESEARCH

Today's routing tools are not aware of power dissipation of routed nets. In this paper we presented and solved the novel problem of simultaneous power–delay optimization of a bundle of signals with gridded (discrete value) design rules, which can be applied as a post processing stage after the detailed routing is finished. We developed an efficient DP algorithm which solves the problem exactly. As a result, a power–delay Pareto curve was obtained which can be used by the designer to assess goodness of the current design state and derive important design implications. We showed that five values of available wire widths and spaces are enough to get to as close as 5% to the exact continuous solution and that using just two or three values of widths and spaces is insufficient.

The algorithm run-time can be reduced by a factor of 12 just by rounding power and delay values up to 1.25% of maximal power/delay value.

Finally, a variation of the algorithm developed in this paper has been deployed for improving layouts of complete functional blocks which use lower level metal layers subject to discrete value design rules. The application of the algorithm on real design blocks showed reduction of 6% in interconnect power and 5% in interconnect delay on average. As process technology will progress to 22 nm feature size, more layers will turn to discrete rules, so application of the DP algorithm can cover full-chip routing as well, and further power–delay reduction would be achievable.

ACKNOWLEDGMENT

The authors would like to express their deep gratitude for the excellent comments and suggestions provided by the reviewers, which enabled significant improvements in the manuscript.

REFERENCES

[1] S. Borkar, "Low power design challenges for the decade," in *Proc. Conf. Asia South Pacific Design Automat.*, 2001, pp. 293–296.
[2] J. Cong, L. He, C. K. Koh, and Z. Pan, "Interconnect sizing and spacing with consideration of coupling capacitance," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 9, pp. 1164–1169, Sep. 2001.
[3] S. Wimer, S. Michaely, K. Moiseev, and A. Kolodny, "Optimal bus sizing in migration of processor design," *IEEE Trans. Circuits Syst. I*, vol. 53, no. 5, pp. 1089–1100, May 2006.
[4] E. Macii, M. Poncino, and S. Salerno, "Combining wire swapping and spacing for low-power deep-submicron buses," in *Proc. 13th ACM Great Lakes Symp. VLSI*, 2003, pp. 198–202.
[5] K. Moiseev, S. Wimer, and A. Kolodny, "Power–delay optimization in vlsi microprocessors by wire spacing," *ACM Trans. Design Automat. Electron. Syst.*, vol. 14, no. 4, pp. 55:1–55:28, Aug. 2009.
[6] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, "Interconnect-power dissipation in a microprocessor," in *Proc. Int. Workshop System-Level Interconnect Prediction*, 2004, pp. 7–13.
[7] W. Shi and Z. Li, "An O(nlogn) time algorithm for optimal buffer insertion," in *Proc. DAC*, 2003, pp. 580–585.
[8] D. A. Hodges, H. G. Jackson, and R. A. Saleh, *Analysis and Design of Digital Integrated Circuits*, 3rd ed. New York: McGraw-Hill, 2004.
[9] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA: Freeman, 1979.
[10] K. Moiseev, S. Wimer, and A. Kolodny, "Dynamic programming algorithm for interconnect channel sizing in discrete design rules," Irwin and Joan Jacobs Center Commun. Inform. Technol., Dept. Elect. Eng., Technion-Israel Instit. Technol., Haifa, Israel, Rep. 730, May 2009.