Optimal VLSI Delay Tuning by Space Tapering With Clock-Tree Application

Eyal Sarfati, Binyamin Frankel, Yitzhak Birk, *Senior Member, IEEE*, and Shmuel Wimer, *Member, IEEE*

***Abstract*— Interconnect shielding is used in very large scale integration designs to prevent noise interference from the cross-coupling capacitance between adjacent signals. This paper takes advantage of the shields already present in the design and uses them to tune the propagation delay of the clock signals by space tapering, thus eliminating expensive and process variation sensitive dedicated delay buffers. The problem of obtaining the desired delay at a minimum shielding cost (silicon area) by tapering its spacing from the signal wires is solved. A clock-tree synthesis methodology that uses shields to obtain useful skews at the underlying flip-flops is proposed. The method was tested on an industrial 28-nm memory controller and ARM® processor designs, operated in 800-MHz and 1.6-GHz clock speed, respectively, and confirmed its viability for delivering the required useful skews to flip-flops. About 90% of the useful skew problems could be solved by shielding manipulations.**

***Index Terms*— Wire shielding, delay tuning, clock-trees, useful skew.**

I. Introduction

INTERCONNECT shielding is used in Very Large Scale Integration (VLSI) designs to prevent noise interference occurring between signals. Shielding wires are connected to the supply voltage. They extend adjacently to the signal wires to avoid unintentional interfering noise to other signals occurring by *cross-coupling* capacitance. The clock signals spread over the entire silicon die to synchronize the operation of the underlying circuits in digital systems are the noisiest, and hence are shielded. They are a source of *signal integrity* problems, which can be avoided by extensive usage of shielding [1].

Clocks, which should reach each of the underlying circuits simultaneously, are sometimes delayed with respect to each other. This is done by applying one of several well-known design techniques such as *time borrowing* [2], [3] *clock skew* tuning [4], [5], and power supply noise reduction [6], [7], [8], among others. These techniques insert *delay buffers* into the clock distribution network. The internal delay of the buffers is subject to wide, unpredictable changes in *process variation*, and has been aggravated by recent progress in VLSI technologies to the nanometer scale [9], [10]. Inserting delay buffers into a clock network is a delicate task and a design burden. Intentional delays are often inserted into logic signals as well. In particular, *min-delay* buffers are inserted to avoid overly fast propagation of logic signals that cause the miscalculation of logic functions (races, hold violations) [11].

In this work, we suggest replacing the delay buffers used for clock delay tuning with shields. Shields are already present for clocks, so this replacement does not require additional silicon and metal resources. Delay tuning by wire shields has several advantages over delay buffers. First, wires are considerably less sensitive to manufacturing process variations than delay buffers. This makes the design more robust and its operation in silicon more predictable at corners [12]. The second advantage is the ease of late design changes (ECOs). These are usually required to meet timing specifications, which impose further layout changes due to the insertion of the delay buffers. This in turn may cause delays in project schedules. Finally but importantly is the energy consumed by the clock delay buffers that toggle at each cycle. Eliminating them saves power. Although the shields included in the design for noise reduction also consume power, there is no comparison between their consumption and that of delay buffers.

This work aims to lay the groundwork for intentional delay insertion and tuning by shields and space tapering. The use of shields in the context of a complete VLSI design is beyond the scope of this paper. The main contributions of this paper are the following:

- presents the novel approach of using shields and space tapering as a replacement of buffers for delay tuning,
- shows the advantages of shielding over buffers for smaller process variations,
- formulates and solves the optimal space tapering problem, and
- demonstrates the viability of delay tuning by shielding and space tapering for clock-tree synthesis.

The remainder of the paper is organized as follows. Section II provides the background on the fundamental interconnect delay model used in VLSI designs. Section III presents the viability of shield usage for delay tuning and shows that shields are less sensitive to process variations. Section IV explores the nature of the optimal shield design by space tapering and its computational procedure. Section V illustrates how a clock-tree supporting a useful clock-skew can be constructed with wire shielding methodology.

Manuscript received February 9, 2017; revised March 23, 2017 and April 12, 2017; accepted April 13, 2017. Date of publication April 27, 2017; date of current version July 26, 2017. This work was supported by the Israeli Chief Scientist under the HiPer Consortium of the MAGNET Program. This paper was recommended by Associate Editor F. J. Kurdahi. *(Corresponding author: Shmuel Wimer.)*

E. Sarfati and Y. Birk are with the Electrical Engineering Department, Technion–Israel Institute of Technology, Haifa 32000, Israel (e-mail: eyal.sarfati@gmail.com; birk@ee.technion.ac.il).

B. Frankel and S. Wimer are with the Engineering Faculty, Bar-Ilan University, Ramat Gan 52900, Israel (e-mail: binyamin.frankel@gmail.com; wimers@biu.ac.il).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2017.2695100

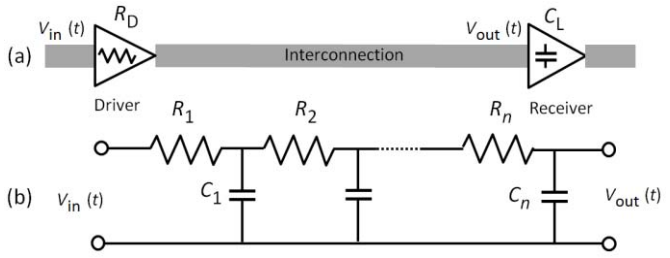


Fig. 1. Driver-to-receiver interconnect (a) and its *RC*-ladder modeling (b).

Section VI reports the simulation results for a 28nm design. Section VII concludes the discussion.

II. Interconnect Delay

The *Elmore delay model* [13] has been widely used in VLSI design since its early days to calculate the interconnect delay [14]. Though simplified, we adopt this RC model rather than RLC model for two reasons. Firstly, inductance which affects delay in multi-gigahertz designs can be neglected in our study which targets near-gigahertz clock cycle [15], [16]. There, inductance, skin effect, induced eddy current and dielectric losses can be neglected. Secondly, it has been shown in [17] that RC model has very good fidelity with RLC model for interconnect tapering optimization.

Consider Fig. 1(a) where a driver connected on the near end, sends a signal along a wire to a receiver connected on the far end. The driver's resistance R_{D} characterizes its driving strength. The receiver has an input capacitance C_{L}. An input unit impulse V_{in} is supplied to the near end at $t = 0$. The Elmore delay δ is defined as

$$\delta = \int_0^\infty t V'_{\mathrm{out}}(t)\, dt, \tag{1}$$

where $V'_{\mathrm{out}}(t)$ is the derivative of the transient response $V_{\mathrm{out}}(t)$ [13].

The interconnection in Fig. 1(a) has distributed resistance and capacitance, and is usually modeled and approximated by the RC ladder shown in Fig. 1(b), where $R_1 = R_{\mathrm{D}}$ and $C_n = C_{\mathrm{L}}$. The Elmore delay in this case is the limit of the sum over all the resistances multiplied by the downstream capacitance [18], as follows

$$\delta = \int_0^\infty t V'_{\mathrm{out}}(t)\, dt \approx \sum_{i=1}^{n} R_i \sum_{j=i}^{n} C_j = \sum_{j=1}^{n} C_j \sum_{i=1}^{j} R_i. \tag{2}$$

The connection from the driver to the receiver usually traverses several metal layers. However, we use the simplified but very popular and useful model shown in Fig. 1 where the interconnecting wire resides on a single metal layer.

VLSI interconnections are designed to meet some predefined delay constraints dictated by the frequency of the clocks synchronizing the operation of the entire system. In another setting, the minimization of the delay in (2) is required. To this end, the expression $\sum_{i=1}^{n} R_i \sum_{j=i}^{n} C_j$ shows that a unit length of the wire close to the driver needs to have a small resistance (a wide wire), since it multiplies a large $\sum_{j=i}^{n} C_j$ downstream capacitance. Similarly, the expression $\sum_{j=1}^{n} C_j \sum_{i=1}^{j} R_i$ shows that a unit length of the wire close to the receiver should have small capacitance (a narrow wire) since it multiplies a large $\sum_{i=1}^{j} R_i$ upstream resistance. Consequently, to minimize the propagation delay, the wire should be tapered, a topic that has been studied extensively in the literature. Fig. 2 illustrates a more general interconnection, where the wire width $w(x)$ varies along its traversal $0 \le x \le L$ from the driver to the receiver.

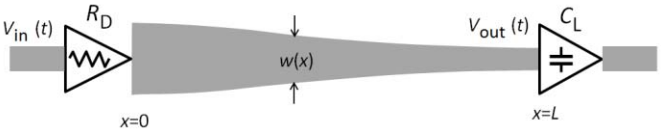


Fig. 2. A tapered interconnect.

The problem of finding $w(x), 0 \le x \le L$ that minimizes (2) was solved analytically in [19]. Though VLSI technologies only allow isothetic rectangular shapes, the continuous formulation highlights the nature of the optimal solution very well. Practically, allowable rectangular shapes can approximate the continuous $w(x)$. The work in [19] prompted other studies [20], [21], [17], [22], [23] all of which explored various aspects of a continuous $w(x)$ and its realization in real manufacturing technologies.

Let $w_{\min}$ be the minimum wire width allowable by the technology in use, and let the resistance and capacitance per $w_{\min} \times w_{\min}$ square of the interconnect metal be r_s and c_s, respectively. The width of the wire $w(x)$ and its length L are expressed as multiplications of $w_{\min}$. There is typically $1 \times w_{\min} \le w(x) \le 3 \times w_{\min}$. The above works studied the$w(x)$ yielding minimum driver-to-receiver delay, given by

$$\delta = R_{\mathrm{D}} \underbrace{\left(\int_0^L c_s w(x)\, dx + C_{\mathrm{L}} \right)}_{(a)} + \int_0^L \frac{r_s}{w(x)} \underbrace{\left(\int_x^L c_s w(y)\, dy + C_{\mathrm{L}} \right)}_{(b)} dx. \tag{3}$$

The term (a) in (3) is the downstream capacitance, which is charged though the driver's resistance R_{D}. The term (b) is the downstream capacitance charged through the resistance $r_s/w(x)$. Fishburn and Schevon found in [19] using the calculus of variations that $w(x)$ minimizing the delay decreases exponentially, as given by the following expression

$$w(x) = \frac{2C_{\mathrm{L}}}{c_s L} W\left(\frac{L}{2} \sqrt{\frac{r_s}{R_{\mathrm{D}}} \frac{c_s}{C_{\mathrm{L}}}} \right) e^{2W\left(\frac{L}{2} \sqrt{\frac{r_s}{R_{\mathrm{D}}} \frac{c_s}{C_{\mathrm{L}}}} \right) \frac{L-x}{L}}, \tag{4}$$

where W is the function satisfying $W(x)\, e^{W(x)} = x$.

Alpert *et al.* [24] showed that wire tapering can yield only a small improvement of propagation delay when combined with maximal repeater insertion. In line with this conclusion

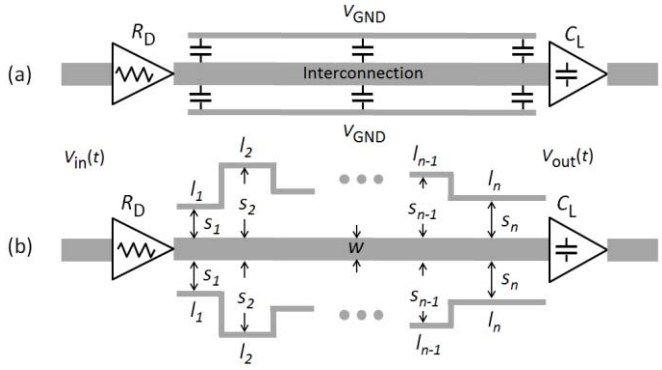


Fig. 3. Shielded interconnect modeling.

we use wires of uniform width (un-tapered), but rather taper the spacing to neighboring shields, using similar mathematical idea as in (3) for the derivation of the optimal tapered spacing. It is important to note that our work deals with shields that have already been inserted by a physical clock-tree synthesis (CTS) tool. Hence, issues such as layer assignment of the vias and interconnects are taken for granted and are not in the scope of the discussion.

III. Delay Tuning by Tapering of Interconnection Shielding

Interconnecting wires are a source of switching noise that arises when they toggle between V_{GND} and V_{DD}. The line-to-line coupling capacitance between parallel adjacent wires is a predominant factor in noise interference, which for sensitive signals can cause circuit failure. Signals that are a source of significant noise are therefore shielded, where the shielding wires are connected to a constant voltage V_{GND} or V_{DD} [1], as shown in Fig. 3(a).

The cross-coupling capacitance between the shielding wires and the interconnect signal introduces further driver-to-receiver propagation delays. This was studied in [25] and [26] in the context of optimizing a global interconnection design methodology, in conjunction with wire width and repeater insertion. These studies considered the delay incurred by shielding as an undesirable burden. However, it can be seen as an aid to solving per-signal problems by satisfying the required delay constraints. Jakushokas and Friedman [26] went a step further by attempting to minimize the cross-coupling noise under delay, power and area constraints. Both studies assumed shielding of fixed spacing from the signal wire, as shown in Fig. 3(a), whereas an optimal per-signal delay tuning may require variable, piecewise-constant spacing, as shown in Fig. 3(b). Karami and Afzali-Kusha [27] attempted to reduce the cross-coupling delay burden incurred by shielding by allowing variable spacing. Inspired by the exponential signal interconnect tapering that minimizes propagation delay in (4), they suggested using exponential shield tapering, and guessed (without proof) that it was optimal. Their assumption was in fact wrong and the optimal tapering obeys a square root shape [28].

Clock-network design is a complex task [29], which is usually done automatically by Electronic Design Automation (EDA) synthesis tools, or manually by specialists when a very high clock rate is needed. Although clocks should theoretically reach all the underlying sequential circuits simultaneously, in fact they do not (see Section I). Here we propose a novel clock delay tuning method that takes advantage of the existing shields.

As shown in Fig. 3(b), let us consider a wire of constant width w connecting the driver and the receiver. A two-sided shield extends along the wire, spaced at $s(x)$, $0 \le x \le L$. To make the illustration independent of nanometers and microns, the wire-to-wire spacing $s(x)$ is expressed as a multiplication factor of $s_{\min}$, which is the minimum wire-to-wire spacing allowable by the technology in use. There is typically $s(x) \in \{1 \times s_{\min}, 2 \times s_{\min}, 3 \times s_{\min}\}$. A commonly used approximation for the unit length line-to-line capacitance of two adjacent wires is given by $c_{ll}/s(x)$, where c_{ll} is a technology parameter.

When the entire shield is positioned in a minimum spacing of 1 ($1 \times s_{\min}$), the normalized driver-to-receiver delay component incurred by the shield is maximal, and is given by

$$\delta^1_{\text{shield}} = R_D c_{ll} L + \frac{r_s c_{ll}}{2w} L^2 = c_{ll} L \left(R_D + \frac{r_s L}{2w} \right). \quad (5)$$

Positioning the entire shield in a maximum space of $3(3 \times s_{\min})$ yields $\delta^3_{\text{shield}} = \delta^1_{\text{shield}}/3$. Positioning the shield in a minimum spacing consumes three times the switching power of maximal spacing, but it occupies a smaller silicon area. A typical tradeoff is to space the shield at $2 \times s_{\min}$, yielding $\delta^2_{\text{shield}} = \delta^1_{\text{shield}}/2$. It follows from (3) that for a constant width w, the signal wire delay component is

$$\delta_{\text{wire}} = R_D (c_s w L + C_L) + \frac{r_s L}{w} \left(\frac{c_s w}{2} L + C_L \right). \quad (6)$$

The range of delay tuning achievable by the shielding is therefore

$$\frac{\delta^1_{\text{shield}} - \delta^3_{\text{shield}}}{\delta_{\text{wire}} + \delta^2_{\text{shield}}} = \frac{\frac{2}{3} c_{ll} L \left(R_D + \frac{r_s L}{2w} \right)}{R_D (c_s w L + C_L) + \frac{r_s L}{w} \left(\frac{c_s w}{2} L + C_L \right) + \frac{1}{2} c_{ll} L \left(R_D + \frac{r_s L}{2w} \right)}. \quad (7)$$

To simplify (7), the following assumptions can be made. The interconnection length is usually a few hundred microns, a case where $c_s w L \gg C_L$. Also, in today's VLSI technologies, $c_{ll} \approx c_s$ [30]. Substitution in (7) yields the following driver-to-receiver *relative delay tuning range* (tuning range for short)

$$\frac{\delta^1_{\text{shield}} - \delta^3_{\text{shield}}}{\delta_{\text{wire}} + \delta^2_{\text{shield}}} \approx \frac{4}{3(2w+1)}. \quad (8)$$

Similar to spacing, usually $1 \le w \le 3$. We conclude from (8) that the tuning range obtained by shielding is 44% for $w = 1$ and 19% for $w = 3$.

To verify the simplified model in (8) we simulated Fig. 3(a) for $100\mu m \le L \le 400\mu m$ and $w = 1, 2, 3$, using 28nm technology. The results in Fig. 4 show that for $L = 400\mu m$ the tuning range varies from 38% for $w = 1$ to 18% for $w = 3$, falling in the ballpark of the simplified model. Furthermore, the trend in Fig. 4 shows that the tuning range increases

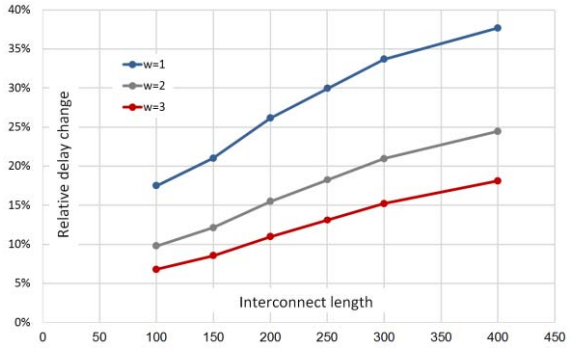


Fig. 4. Relative delay tuning range obtained by shields.

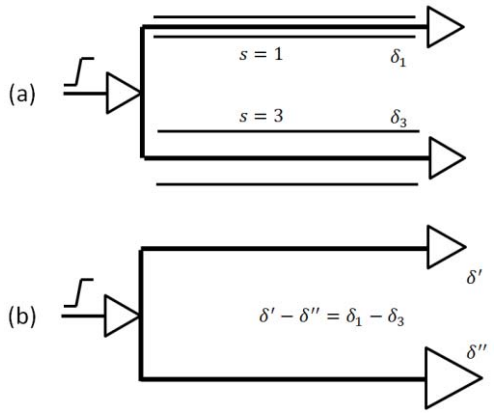
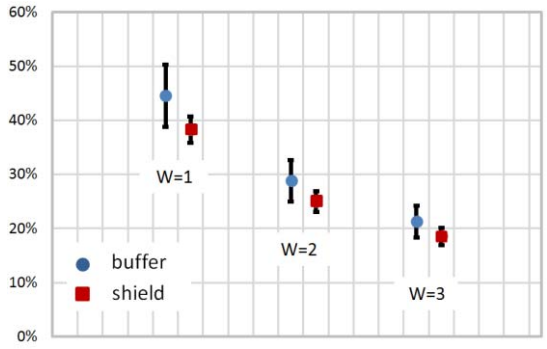


Fig. 5. Setup for the delay variation simulation.

with L, so longer interconnects will benefit more from delay tuning by shields. This is in line with the assumptions of the interconnect capacitance dominance employed in the derivation of (8).

The introduction claimed that the delay obtained by shields is less sensitive to process variations than the delay of buffers. Buffer and interconnection delay sensitivities to process variations were simulated by SPICE for TSMCMPH 28nm technology and the results are shown below. Fig. 5 illustrates the device under test. The simulation took place first for $w = 2$ and $L = 400\mu m$, with X10 buffer size, and buffer's input slew 40pSec. In Fig. 5(a) there are two delays $\delta_1 = 29.2pSec$ and $\delta_3 = 13.6pSec$, which are the driver-to-receiver delay contribution of the shields alone. These delays were obtained in a typical process corner by a shield spacing of $s = 1$ and $s = 3$, respectively. Fig. 5(b) generates the same delay difference with the aid of appropriate buffers for a typical corner. The differences $\delta_1 - \delta_3 = \delta' - \delta'' = 15.6pSec$, representing the tuning range, were implemented for the two configurations and a variety of corners comprising temperature, $V_{\rm dd}$, P/N transistors speed and RC variations.

Fig. 6 shows the simulated variations of the tuning range for the shield and buffer setup in Figs. 5(a) and 5(b), respectively, as the percent of the driver-to-receiver delay, based on the simplified expression derived in (8). The dots are the average delay portion that was measured for buffer (blue) and shield (red). The vertical bars are the delay variations



	Buffers		Shields	
Wire width	Av. [%]	StD. [%]	Av. [%]	StD. [%]
W=1	44.5	5.78	38.3	2.39
W=2	28.8	3.90	24.9	1.91
W=3	21.3	2.94	18.5	1.54

Fig. 6. Percentages of variation ranges for the shield and buffer delays.

obtained across all simulated corners. It is important to note that the shields in Fig. 5(a) hardly affect the slew rate since they already exist in the nominal clock-tree branches shown in Fig. 5(b). Notice the similarity of the delay tuning range for $400\mu m$interconnect length, obtained by the simplification in (8), to the ranges marked in red squares in Fig. 6, obtained by SPICE simulations.

As summarized in the table the shield delay implementation was clearly better than the buffer implementation. The entries in the table are the tuning ranges of the driver-to-receiver delays in percent. The ratios of the variations in the tuning range obtained by shields to that obtained by buffers were $0.41 = 2.39/5.78$, $0.49 = 1.91/3.90$ and $0.52 = 1.54/2.94$, for $w = 1, 2, 3$ respectively, showing that the shield delay tuning sensitivity is about half than buffers. Smaller variations ensure that useful skew will sustain across wide range of operation and silicon conditions.

IV. Optimal Tapered Shielding Design for Delay Tuning

We now address the question of how to obtain a desired delay with shields while consuming minimum area resources. We first show that the optimal space between the signal's wire and the shield's wire must be monotonic increasing from driver to receiver. This leads to a simple set of equations which allows finding the optimal shield shape.

Fig. 3(b) illustrates a piecewise-parallel shield comprising n wire segments of length l_i and spacing s_i, $1 \le i \le n$, $\sum_{i=1}^{n} l_i = L$. Theoretically, there are infinite shield compositions to satisfy a driver-to-receiver delay tuning. While technology and layout constraints may restrict these to a finite number, this number is still huge and an appropriate solution is in order. We first show that the optimal shielding shape is stepwise monotonic increasing. This will be used in below to obtain the accurate shield design.

Let the driver-to-receiver delay incurred in Fig. 3(b) require the addition of a delay denoted by $\delta_{\rm shield}$. Similar to (5),

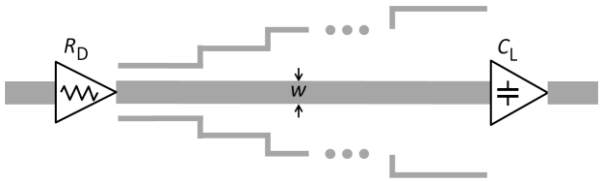


Fig. 7. Optimal wire shielding shape must have monotonic increasing spacing.

the delay contribution of the piecewise-parallel shield can be calculated with the Elmore model by summing the upstream driving resistances along the driver-to-receiver span, multiplied by the downstream capacitive loads. For sufficiently large n there is

$$\delta_{\text{shield}} \approx R_{\text{D}} c_{ll} \sum_{i=1}^{n} \frac{l_i}{s_i} + \frac{r_s c_{ll}}{w} \sum_{i=1}^{n} l_i \left(\sum_{j=i}^{n} \frac{l_j}{s_j} \right). \tag{9}$$

Recall that we are interested in minimizing the area between the signal wire and the shield, given by

$$A = \sum_{i=1}^{n} l_i s_i, \tag{10}$$

which is a waste since it cannot be used for routing. With the above notations, an optimization problem with $2n$ variables s_i and l_i, $1 \le i \le n$, is defined to minimize (10) subject to (9). In the Appendix we prove that the optimal shielding shape is stepwise monotonic increasing, as shown in Fig. 7. The monotonic property of the optimal shield is used below to obtain the accurate shield design.

Two computational approaches that yield an optimal solution are possible. The first involves directly addressing the optimization problem imposed by the constraint equation (9) and the objective to be minimized in (10). This optimization problem has $2n$ variables s_i and l_i, $1 \le i \le n$. Note that due to the optimality of monotonic space tapering the number of variables can be reduced to the number of spaces allowable by the process technology. It can be solved by any appropriate optimization solver or in an analytical manner with MATLAB by writing the KKT conditions explicitly [31].

The second approach relies on an analytical continuous formulation of the optimal shielding problem, which was recently solved in [28]. It can be approximated by a legal discrete solution as shown below. The analytical continuous (albeit non-practical) shield is given by

$$s(x) = \frac{2R_{\text{D}}^2 c_{ll} w}{3\delta_{\text{shield}} r_s} \left[\left(1 + \frac{r_s L}{R_{\text{D}} w} \right)^{\frac{3}{2}} - 1 \right] \sqrt{1 + \frac{r_s}{R_{\text{D}} w} x}. \tag{11}$$

Let $\mathbf{S} = \{s_0, \ldots, s_n\}$, $s_0 < s_1 < \ldots < s_n$, be a finite set of $n+1$ real positive numbers. The function $s(x)$ in (11) satisfies $s(0) = s_0$ and$s(L) = s_n$. The $n+1$ values represent admissible wire-to-wire spacing, and they are *equally spaced from each other* by $(s_n - s_0)/n$. This is a common VLSI technology scenario. A feasible stair approximation $\sigma(x) : [0, L] \to \mathbf{S}$ is a right-continuous, piecewise constant function, satisfying

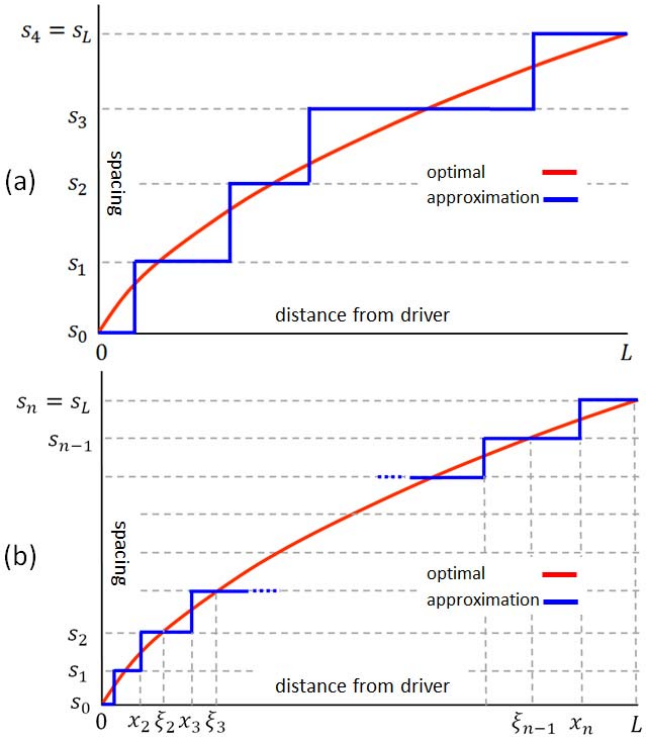


Fig. 8. Stair approximation, (a) arbitrary, (b) optimal.

$\sigma(0) = s_0$ and $\sigma(L) = s_n$. The distance $d(s, \sigma)$ between s and σ is defined by

$$d(s, \sigma) = \max_{0 \le x \le L} |s(x) - \sigma(x)|, \tag{12}$$

and the best approximation σ^* satisfies

$$d(s, \sigma^*) = \min_{\sigma} d(s, \sigma). \tag{13}$$

Due to the non-decreasing monotony of the optimal discrete shielding shown in Fig. 7 and proven in the appendix, and the non-decreasing monotony of $s(x)$ in (11), it is possible to assume without loss of generality that σ^* in (13) is monotonic, taking all the $n+1$ values $s_0, \ldots, s_n$.

Fig. 8(a) shows the optimal square root shielding function $s(x)$ in red and the stair approximation overlaid in blue, for $n = 4$. Since σ^* takes all the $n+1$ values $s_0, \ldots, s_n$, it implies $n+1$ points $\xi_0, \ldots, \xi_n$, $0 = \xi_0 < \xi_1 < \ldots < \xi_{n-1} < \xi_n = L$, for which there is $\sigma^*(\xi_i) = s_i$. Moreover, stair i must be bisected by $s(x)$ at a point x_i, namely

$$s(x_i) = \frac{(s_{i-1} + s_i)}{2}, 1 \le i \le n. \tag{14}$$

Otherwise, some of $x_1, \ldots, x_n$ could be shifted leftwards or rightwards to yield a better approximation than σ^* does. Fig. 8(b) illustrates the best approximation. Note that due to the uniform spacing between s_{i-1} and s_i, $1 \le i \le n$, points $x_1, \ldots, x_n$ are uniquely determined, so σ^* is unique.

Fig. 9 plots in blue the delay accuracy of $\sigma^*(x)$ compared to $s(x)$ as a function of n, defined by their delay difference divided by the delay of $s(x)$. It shows that technologies offering three permissible spaces ($n = 2$) lead to nearly a 1% error, whereas for technologies offering five spaces ($n = 4$)

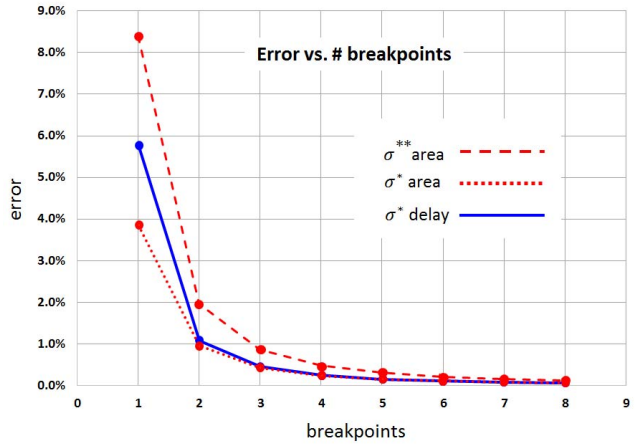


Fig. 9. Best approximation errors and the optimal discrete solution.

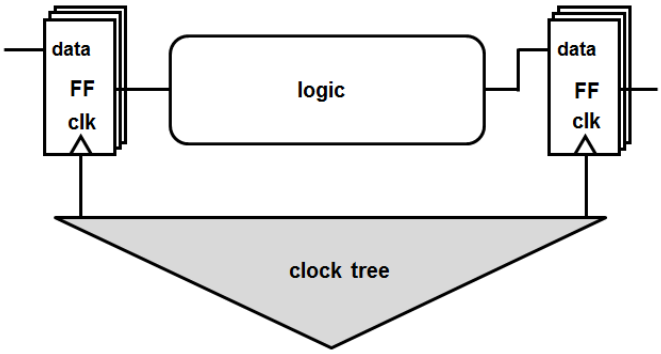


Fig. 10. General synchronous digital logic system.

the error is less than 0.25%. These errors are negligible in terms of practical considerations. The dotted red line is the excessive area implied by $\sigma^*(x)$ compared to $s(x)$, defined by their area difference divided by the area of $s(x)$, which is shown to be negligible for $n \geq 2$.

Finally, we solved the discrete optimization problem of achieving the δ_{shield} delay with a minimum area by using KKT conditions [31] that yield the piecewise constant spacing function denoted by $\sigma^{**}(x)$. This solution achieves the exact required delay δ_{shield}. The dashed red line shows its excessive area compared to $s(x)$, defined by their area difference divided by the area of $s(x)$. Thus, with some tolerance of the delay accuracy, σ^* yields a smaller area than σ^{**}. It is important to note that while finding σ^{**} involves a tedious Lagrange optimization, σ^* is derived by simple substitutions in (11) and (14).

V. Solving Useful Skew Delay Insertion in Clock-Trees

VLSI designers have studied delay distribution in *clock-trees* for a long time. Fig. 10 illustrates a general synchronous digital logic system. The clock signal driving the underlying flip-flops (FFs) should theoretically reach all of them simultaneously. *Time-borrowing* algorithms have been devised to relax timing constraints, thus allowing higher clock speeds [4], [5]. These techniques, called the *useful clock skew* (useful skew for short), shift the arrival time of the clock signal to the sequential circuits in some prescribed amount relatively to a nominal clock referred as zero. The shifts are obtained by inserting delay buffers in the clock-trees. Shifting the arrival time of the clock signal is also an effective technique to reduce *power supply noise* (instantaneous voltage fluctuations), which is a challenge in today's VLSI technologies [6], [8], [7].

A design-flow supporting the useful clock skew determines whether the clock at the input of a certain FF will precede, arrive on time, or be late, relative to a global zero time. An EDA backend design-flow first places the FFs according to timing convergence considerations. The clock signal is then delivered to the FFs shown in Fig. 10 by a physical routing tool called a clock-tree synthesizer (CTS) that inserts delay buffers in the clock-tree to satisfy zero or a useful skew [32], [33].

The CTS input is a list of triplets $\langle (x_i, y_i), t_i \rangle$, $1 \leq i \leq n$, where n is the number of FFs requiring time shifting, t_i is the required clock skew and (x_i, y_i) is the location of the FF as decided by the placement tool. For zero skew $t_i = 0$, whereas for a useful skew $t_i > 0$. CTS usually builds the clock-tree in a bottom-up fashion. It starts at the leaves of the clock-tree where all the FFs are placed. The FFs are clustered in groups sharing a common parent node of the tree. In order to generate an efficient tree with short interconnections, the CTS algorithm aims at clustering FFs placed close to each other. It also tries to cluster FFs having close skew specifications. This way the skew can be satisfied by a single delay buffer located at the parent node, whereas the small differences in the leaves are satisfied by fine delay tuning [34].

The FFs' clustering goal is to capture both delay and location proximity. This is reminiscent of the situation that occurs in post-layout multi-bit flip-flop (MBFF) clustering, where several FFs are merged into a single cell for clock power savings [35], [36]. There, the timing slacks of the underlying FFs obtained after timing closure are translated into rectangular regions in the xy plane, each centered at the position of the corresponding FF. The regions are used to guide FF merging that retains the timing closure of the design.

As in MBFF, our CTS proposal translates the delay requirements of each FF into a region in the xy-plane. For the sake of illustration and simplicity without loss of generality we use a binary clock-tree and an Euclidean distance metric. Binary trees are also widely used in real implementations and are supported by EDA tools; e.g., in H-trees [37], [38]. Table I shows the translation of the shielded wire lengths into delays. The delays were derived from the data used in Fig. 4. The table specifies how much delay per wire-length can be achieved when using shields with a minimum spacing of $s = 1$ and a maximum spacing of $s = 3$ (see Fig. 5). Clearly, any value between the two extreme cases is implementable by using stepwise segmented shielding as was described in Section IV.

The data in Table I make it possible to convert the CTS specifications $\langle (x_i, y_i), t_i \rangle$, $1 \leq i \leq n$, to a list of quadruples $\langle (x_i, y_i), (\alpha_i, \beta_i) \rangle$, $1 \leq i \leq n$. α_i is the shortest distance of shielded wire using spacing $s = 1$ needed to generate the skew t_i, and β_i is the longest distance of shielded wire using spacing $s = 3$ needed to generate the skew t_i. This is depicted

TABLE I
TRANSLATIONS OF SHIELDED WIRE LENGTH INTO DELAYS

length [μm]	25	50	100	200	500
s=1 delay [pSec]	9	12	16	25	69
s=3 delay [pSec]	8	9	12	18	47

TABLE II
TRANSLATIONS OF DELAYS INTO SHIELDED WIRE LENGTH

delay [pSec]	10	15	20	40	80
α [μm]	29	95	144	302	580
β [μm]	55	165	243	427	842

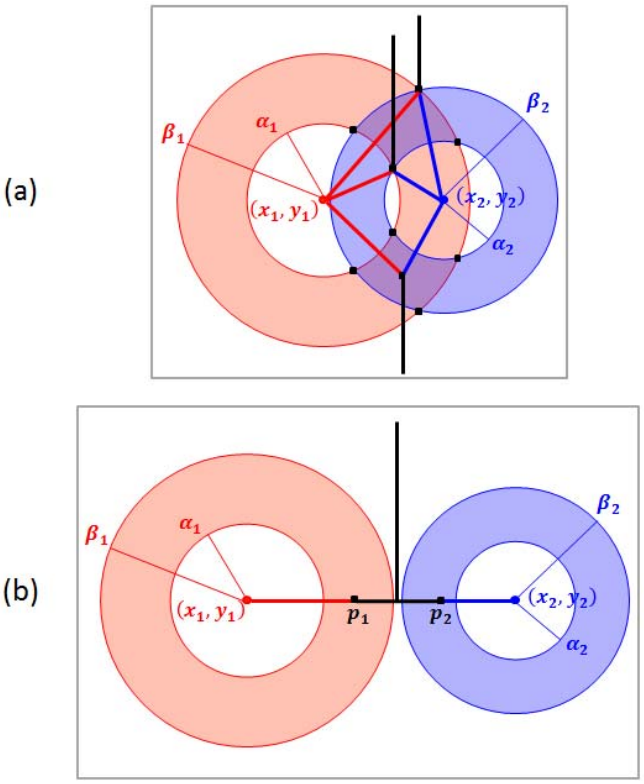


Fig. 11. Useful skew regions of FFs with possible clock-tree constructions. Intersecting useful skew regions (a), nonintersecting (b).

in Fig. 11 that illustrates two FFs: FF_1 and FF_2 located at (x_1, y_1) and (x_2, y_2), respectively. The rings are the feasible useful skew regions obtained by shielded wires connected to the clock input of the respective FFs. Table II shows the timing to distance translation.

The feasible rings enable a bottom-up construction of a clock-tree. Fig. 11(a) shows the case where the parent node of the two leaf FFs, sometimes called *point of divergence*, can be located anywhere in the intersection of the rings. The red and blue connections depict the shielded wires connected to FF_1 and FF_2, respectively. The point of divergence is connected upwards in the tree by the black wire. The role of the CTS routing algorithm is to bring the clock signal to the point of divergence with zero skew. Fig. 11(b) shows the case of disjoint rings. The CTS routing brings the clock signal to points p_1 and p_2 with zero skew.

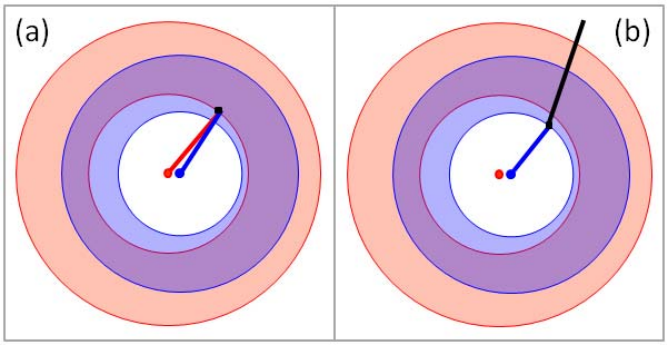


Fig. 12. Clock tree connection, (a) inefficient, (b) efficient.

Supply of the useful skew by the lowest tree level as shown in Fig. 11 is not always efficient. Consider the situations depicted in Fig. 12, where the FF locations almost coincide. Using last-level connections to each of the FFs as shown in case (a) is inefficient and case (b) should be used. There, FF_1 (red) should be connected to FF_2 with a small delay buffer to supply the delay deficit in FF_1. In cases where the useful skews cannot be fully supplied at the leaves, the skew deficit is assigned to the point of divergence and is taken into account at the next level pairing. The detailed implementation of zero skew routing is beyond the scope of this paper and can be found elsewhere, e.g. [32], [33].

The next question is how to pair the FFs at the leaves, and then how to pair the points of divergence at each level in the bottom-up tree construction. For this purpose, we briefly describe a *minimum cost perfect graph matching* heuristic, which is also used to merge FFs in clock-gating design flows [39], [40]. For FFs pairing, an n-vertex weighted graph $G(V, E, w)$ is defined, where $v \in V$ represents a FF. An edge $e(u, v) \in E$ exists if u and v are candidates for a common parent in the clock-tree; e.g., if their corresponding FFs belong to the same clock domain and they are not located too far from each other. A weight $w(e)$ is assigned according to the relative positions of the feasible useful skew rings of the corresponding FFs, as illustrated in Fig. 11. A minimum cost perfect graph matching problem is then solved for $G(V, E, w)$, yielding $n/2$ FFs pairs with their points of divergence.

The $n/2$ points of divergence create a new problem. An $n/2$-vertex weighted graph is defined, to which the vertices' positive useful skew deficits are assigned if the shielded connections to the FFs could not fully supply their required useful skews. To exploit shielded wires at the upper tree level to supply the useful skew deficits, a technique similar to the one in Fig. 11 is used. The edge weights of the new graph reflect the distance between the corresponding points of divergence. The pairing repeats recursively $\log_2 n$ times until a full binary clock-tree is obtained.

VI. SIMULATION RESULTS

To assess the feasibility of achieving the useful skew by adding shielding delays we simulated two Marvell's designs.

TABLE III
ROOT-TO-LEAF INSERTION DELAY

root-to-leaf length [µm]	s=1 delay [pSec]	s=3 delay [pSec]	delay diff [pSec]
1000	177	131	46
1500	264	187	77
2000	389	270	119

Let us consider an ideal clock signal of 1 GHz, which is 1 nSec clock cycle, and 10 pSecslope. A typical propagation delay from the root of a clock tree to a FF's clock input, called the *insertion delay*, is 500 pSec, whereas the desired clock skew may reach up to 10% of the clock cycle, namely 100 pSec. Assuming a die of 2 mm side, the root to leaf wire length of a binary tree implemented like an H-tree is 2 mm $\times \sum_{i \geq 1} 1/2^i \cong$ 2mm. Section V described how to accumulate the required delays along the clock-tree levels. Referring to the dynamic range of shielding delay shown in Fig. 4, and the actual attainable shield delays in Table I, achieving 100 pSec is feasible. Table III presents the root-to-leaf insertion delay and the corresponding delay tuning range by shielding for 28nm technology in 1, 1.5mm and 2 mmdies, obtained by SPICE simulations. The table shows a good fit of the simulation to the above assessment. A key to achieving the useful skew by root-to-leaf accumulation is being able to group close FFs having comparable useful skew requirements as described in Section V.

A. Simulation of Memory Controller

The above useful skew delay insertion methodology was tested by simulation of Marvell's memory controller designed in 28nmHPM TSMC technology, operated at 800MHz. Its die size is nearly 1 mm and the design includes nearly 40K FFs, 1215 of which require useful skew. Fig. 13 shows the locations of these FFs across the die.

Fig. 14(a) shows the useful skew distribution and Fig. 14(b) presents its accumulative histogram obtained by simulations. Table III shows that for a 1 mmdie, a delay tuning range of 46 pSec is attainable. Inspection of Fig. 14(b) indicates that 89% of the FFs requiring a useful skew can be satisfied by shielding alone. The remaining 11% can still use shields, but either serpentine routing or delay buffers must supply the skew deficits.

B. Simulation of ARM® Processor

A second useful skew delay insertion methodology was tested by simulation of Marvell's ARMv7® based processor [41], designed in 28nmHPM TSMC technology, operated at 1.6GHz. Its die size is nearly 1.5 mmand the design includes 70K FFs, 3892 of which require useful skew. Fig. 15 shows the locations of these FFs across the die.

Fig. 16(a) shows the useful skew distribution and Fig. 15(b) presents its accumulative histogram. Table III shows that for a 1.5 mmdie, a delay tuning range of 77 pSec is attainable. Inspection of Fig. 16(b) indicates that 92% of the FFs requiring a useful skew can be satisfied by shielding alone. The remaining 8% can still use shields, but either serpentine routing or delay buffers must supply the skew deficits.

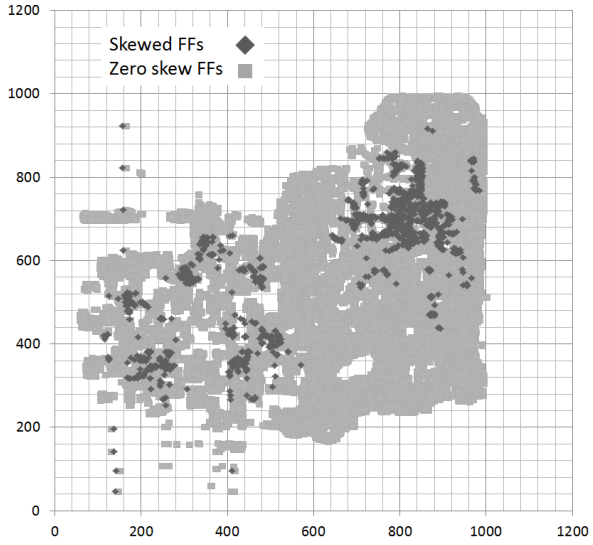


Fig. 13. FFs location across the die of a memory controller.

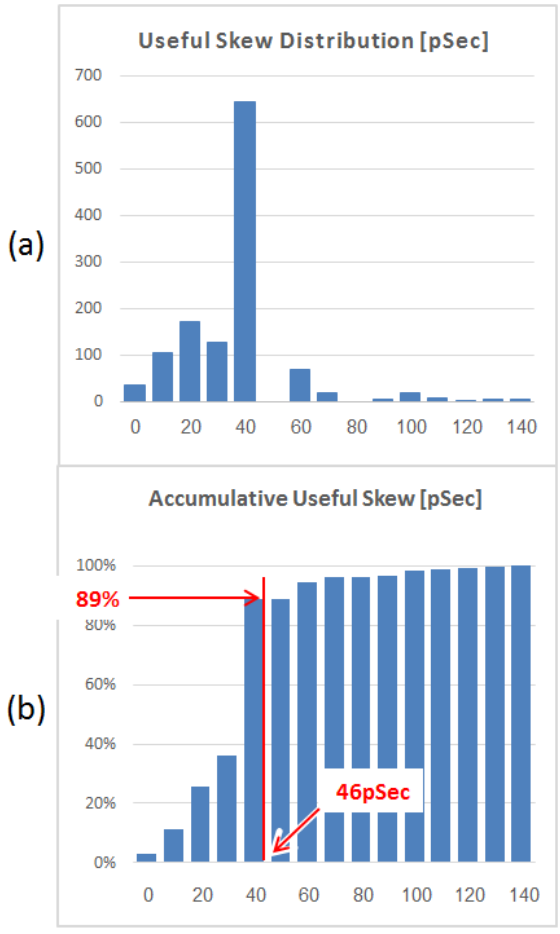


Fig. 14. Useful skew of a memory controller, (a) distribution, (b) accumulative.

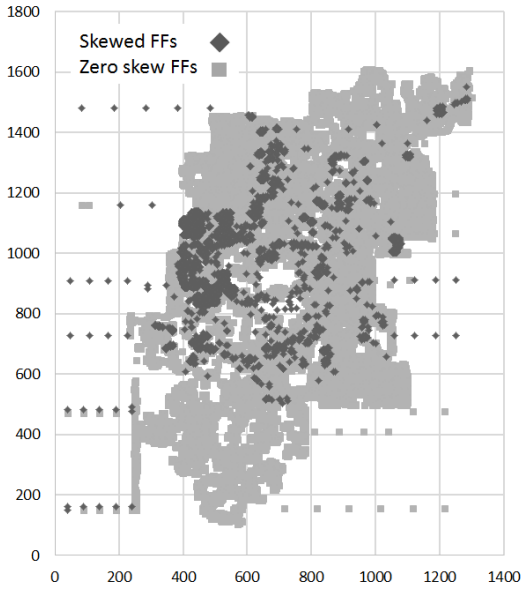


Fig. 15. FFs location across the die of an ARM® based processor.

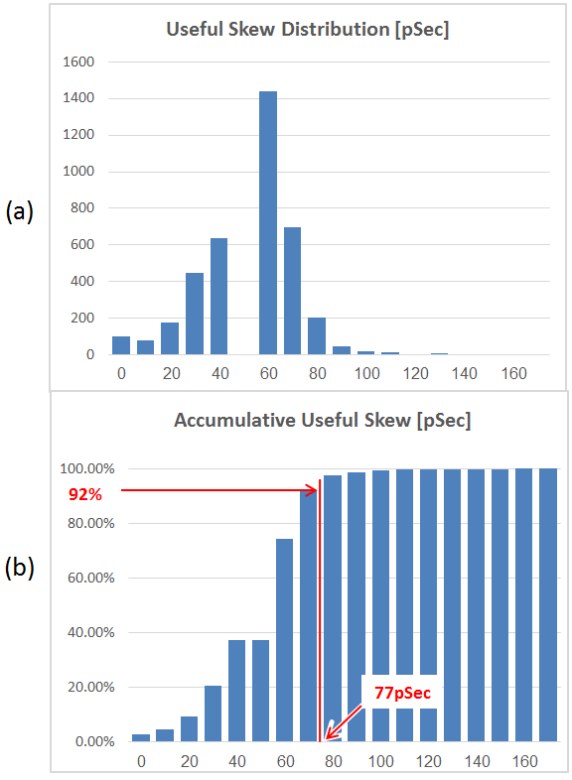


Fig. 16. Useful skew of ARM® based processor, (a) distribution, (b) accumulative.

VII. Conclusion

This paper proposed a CTS methodology that uses the shields already present in the design to tune the propagation delay and obtain a useful skew of the clock signals. It is shown to be a viable alternative to expensive and process variation sensitive delay buffers. The method was tested on an industrial 28nm memory controller design, and demonstrated its ability to deliver the required useful skews to the underlying flip-flops.

Appendix

To show that the optimal shielding shape is stepwise monotonic increasing, we can consider without loss of generality that the shield segments are of equal length; namely $l_i = L/n$. To show the monotony, assume on the contrary that there is a piecewise-parallel shield yielding a minimum area A', for which there is some $1 \leq k < n$ and $s_k > s_{k+1}$. Let $s_k = \alpha$ and $s_{k+1} = \beta$, so $\alpha > \beta$. Let δ'_{shield} be the delay resulting from the piecewise-parallel shield. We get

$$\delta'_{\text{shield}}(\alpha,\beta) = \frac{R_D c_{ll} L}{n}\sum_{i=1}^{n}\frac{1}{s_i} + \frac{r_s c_{ll} L^2}{wn^2} \times \left[\sum_{i=1}^{k-1}\sum_{j=i}^{n}\frac{1}{s_j} + \frac{1}{\alpha} + \frac{2}{\beta} + 2\sum_{j=k+2}^{n}\frac{1}{s_j} + \sum_{i=k+2}^{n}\sum_{j=i}^{n}\frac{1}{s_j}\right]. \tag{15}$$

Let us modify the shield by swapping the k and the $k+1$ segments, so $s_k = \beta$ and $s_{k+1} = \alpha$. This swap yields the area $A''(\beta,\alpha) = A'(\alpha,\beta)$; hence the area between the shield and the wire does not change. The corresponding delay is

$$\delta''_{\text{shield}}(\beta,\alpha) = \frac{R_D c_{ll} L}{n}\sum_{i=1}^{n}\frac{1}{s_i} + \frac{r_s c_{ll} L^2}{wn^2} \times \left[\sum_{i=1}^{k-1}\sum_{j=i}^{n}\frac{1}{s_j} + \frac{1}{\beta} + \frac{2}{\alpha} + 2\sum_{j=k+2}^{n}\frac{1}{s_j} + \sum_{i=k+2}^{n}\sum_{j=i}^{n}\frac{1}{s_j}\right]. \tag{16}$$

Subtracting (16) from (15) yields

$$\varepsilon = \delta'_{\text{shield}}(\alpha,\beta) - \delta''_{\text{shield}}(\beta,\alpha) = \frac{r_s c_{ll} L^2}{wn^2}\left(\frac{1}{\beta} - \frac{1}{\alpha}\right) > 0. \tag{17}$$

We thus obtain a smaller delay with the same area. Let us increase δ''_{shield} by ε defined in (17) as follows. The delay $\delta''_{\text{shield}}(\beta,\alpha)$ is continuous and monotonically increasing with $1/\beta$, and $\delta''_{\text{shield}}(\beta,\alpha) \to \infty$as $\beta \to 0$. There must therefore exist some $0 < \gamma < \beta$ such that $\delta''_{\text{shield}}(\gamma,\alpha) = \delta''_{\text{shield}}(\beta,\alpha) + \varepsilon = \delta'_{\text{shield}}(\alpha,\beta)$. However

$$\begin{aligned} A''(\gamma,\alpha) &= \frac{L}{n}\left(\sum_{i=1}^{k-1} s_i + \gamma + \alpha + \sum_{i=k+2}^{n} s_i\right) \\ &< \frac{L}{n}\left(\sum_{i=1}^{k-1} s_i + \beta + \alpha + \sum_{i=k+2}^{n} s_i\right) \\ &= A''(\beta,\alpha) = A'(\alpha,\beta), \end{aligned}$$

which contradicts the fact that A' is the minimum area achievable for δ'_{shield}.

ACKNOWLEDGEMENT

The authors are thankful to Marvell Corporation for providing the memory controller design data. They are also thankful to the anonymous reviewers for their useful and helpful comments.

REFERENCES

[1] E. Salman and E. Friedman, *High Performance Integrated Circuit Design*. New York, NY, USA: McGraw-Hill, 2012.
[2] S. Lee, S. Paik, and Y. Shin, "Retiming and time borrowing: Optimizing high-performance pulsed-latch-based circuits," in *IEEE/ACM Int. Conf. Comput.-Aided Design-Dig. Tech. Papers, (ICCAD)*, Nov. 2009, pp. 375–380.
[3] S. Tam, S. Rusu, U. N. Desai, R. Kim, J. Zhang, and I. Young, "Clock generation and distribution for the first IA-64 microprocessor," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1545–1552, Nov. 2000.
[4] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Comput.*, vol. 39, no. 7, pp. 945–951, Jul. 1990.
[5] R. B. Deokar and S. S. Sapatnekar, "A graph-theoretic approach to clock skew optimization," in *Proc. IEEE Int. Symp. Circuits Syst.-(ISCAS)*, Jun. 1994, pp. 407–410.
[6] P. Vuillod, L. Benini, A. Bogliolo, and G. De Micheli, "Clock skew optimization for peak current reduction," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 1996, pp. 265–270.
[7] J. Kim, D. Joo, and T. Kim, "An optimal algorithm of adjustable delay buffer insertion for solving clock skew variation problem," in *Proc. 50th Annu. Design Autom. Conf.*, 2013, p. 90.
[8] Y. Kaplan and S. Wimer, "Mixing drivers in clock-tree for power supply noise reduction," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 5, pp. 1382–1391, May 2015.
[9] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2003, p. 900.
[10] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 4, no. 4, pp. 14–19, Jul. 2003.
[11] N. H. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, vol. 2. Reading, MA, USA: Addison-Wesley, 1993.
[12] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the effect of process variations on the delay of static and domino logic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 697–710, May 2010.
[13] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, 1948.
[14] H. B. Bakoglu and J. D. Meindl, "Optimal interconnection circuits for VLSI," *IEEE Trans. Electron Devices*, vol. 32, no. 5, pp. 903–909, May 1985.
[15] Y. I. Ismail, E. G. Friedman, and J. L. Neves, "Figures of merit to characterize the importance of on-chip inductance," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 4, pp. 442–449, Dec. 1999.
[16] Y. I. Ismail and E. G. Friedman, "Effects of inductance on the propagation delay and repeater insertion in VLSI circuits: A summary," *IEEE Circuits Syst. Mag.*, vol. 3, no. 1, pp. 24–28, 1st Quart., 2003.
[17] Y. Gao and D. F. Wong, "Wire-sizing optimization with inductance consideration using transmission-line model," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 12, pp. 1759–1767, Dec. 1999.
[18] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 2, no. 3, pp. 202–211, Jul. 1983.
[19] J. P. Fishburn and C. A. Schevon, "Shaping a distributed-RC line to minimize Elmore delay," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 42, no. 12, pp. 1020–1022, Dec. 1995.
[20] J. P. Fishburn, "Shaping a VLSI wire to minimize Elmore delay," in *Proc. Eur. Conf. Design Test*, Mar. 1997, pp. 244–251.
[21] C.-P. Chen and D. F. Wong, "Optimal wire-sizing function with fringing capacitance consideration," in *Proc. 34th Annu. Design Autom. Conf.*, Jun. 1997, pp. 604–607.
[22] H. Zhang, M. D. Wong, K.-Y. Chao, and L. Deng, "Wire shaping is practical," in *Proc. Int. Symp. Phys. Design*, 2009, pp. 131–138.
[23] M. A. El-Moursy and E. G. Friedman, "Wire shaping of RLC interconnects," *Integr., VLSI J.*, vol. 40, no. 4, pp. 461–472, 2007.
[24] C. J. Alpert, A. Devgan, J. P. Fishburn, and S. T. Quay, "Interconnect synthesis without wire tapering," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 1, pp. 90–104, Jan. 2001.
[25] A. B. Kahng, S. Muddu, and E. Sarto, "Tuning strategies for global interconnects in high-performance deep-submicron ICs," *VLSI Design*, vol. 10, no. 1, pp. 21–34, 1999.
[26] R. Jakushokas and E. G. Friedman, "Resource based optimization for simultaneous shield and repeater insertion," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 742–749, May 2010.
[27] M. A. Karami and A. Afzali-Kusha, "Exponentially tapering ground wires for Elmore delay reduction in on chip interconnects," in *Proc. Int. Conf. Microelectron. (ICM)*, 2006, pp. 99–102.
[28] B. Frankel and S. Wimer, "Optimal VLSI delay tuning by wire shielding," *J. Optim. Theory Appl.*, vol. 170, no. 3, pp. 1060–1067, 2016.
[29] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proc. IEEE*, vol. 89, no. 5, pp. 665–692, May 2001.
[30] "Interconnect summary," in *Proc. ITRS—Int. Technol. Roadmap Semiconductors*, 2013.
[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[32] Y.-C. Hsu, J.-M. Ho, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 11, pp. 799–814, Nov. 1992.
[33] I.-M. Liu, T.-L. Chou, A. Aziz, and D. F. Wong, "Zero-skew clock tree construction by simultaneous routing, wire sizing and buffer insertion," in *Proc. Int. Symp. Phys. Design*, May 2000, pp. 33–38.
[34] S. Wimer, "Optimal weight allocation in rooted trees," *J. Combinat. Optim.*, vol. 31, no. 3, pp. 1023–1033, 2016.
[35] S.-H. Wang, Y.-Y. Liang, T.-Y. Kuo, and W.-K. Mak, "Power-driven flip-flop merging and relocation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 31, no. 2, pp. 180–191, Feb. 2012.
[36] M. P. H. Lin, C. C. Hsu, and Y. C. Chen, "Clock-tree aware multibit flip-flop generation during placement for power optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 2, pp. 280–292, Feb. 2015.
[37] S. Gary, P. Ippolito, G. Gerosa, C. Dietz, J. Eno, and H. Sanchez, "PowerPC 603, a microprocessor for portable computers," *IEEE Des. Test Comput.*, vol. 11, no. 4, pp. 14–23, Dec. 1994.
[38] M. Sarrafzadeh and C.-K. Wong, *An Introduction to VLSI Physical Design*. New York, NY, USA: McGraw-Hill, 1996.
[39] S. Wimer and A. Albahari, "A look-ahead clock gating based on auto-gated flip-flops," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 5, pp. 1465–1472, May 2014.
[40] S. Wimer, "On optimal flip-flop grouping for VLSI power minimization," *Oper. Res. Lett.*, vol. 41, no. 5, pp. 486–489, 2013.
[41] Marvell. (2017). *ARMADA XP*. [Online]. Available: http://www.marvell.com/embedded-processors/armada-xp/

Eyal Sarfati received the B.Sc. degree in electrical engineering from Technion–Israel Institute of Technology, Haifa, Israel, in 2009, where he is currently pursuing the M.Sc. degree. He is now with Marvell, where he is currently a VLSI backend Design Engineer.

Binyamin Frankel received the B.Sc. and M.Sc. degrees in electrical engineering from Bar-Ilan University in 2014 and 2016, respectively, where he is currently pursuing the Ph.D. degree in computer engineering. He is interested in VLSI circuits and systems design optimization.

Yitzhak Birk (M'82–SM'02) received the B.Sc. *(cum laude)* and M.Sc. degrees from the Technion–Israel Institute of Technology, Haifa, Israel, in 1975 and 1982, respectively, and the Ph.D. degree from Stanford University, Stanford, CA, USA, in 1987, all in electrical engineering. He was a Research Staff Member with the IBM's Almaden Research Center. He has been on the Faculty of the Electrical Engineering Department, Technion–Israel Institute of Technology, since 1991, where he currently heads the Parallel SystemsLaboratory.

His research interests include computer and communication systems with much attention to the storage subsystem and to the interplay between storage and communication. The true application requirements are considered in each case. The judicious exploitation of redundancy, coding, and randomization for performance enhancement and cross-disciplinary approaches, has been recurring themes in much of his work.

Shmuel Wimer received the B.Sc. and M.Sc. degrees in mathematics from Tel-Aviv University, Israel, in 1978 and 1981, respectively, and the D.Sc. degree in electrical engineering from the Technion–Israel Institute of Technology, Israel, in 1988.

From 1978 to 2009, he was with industry in research and development, engineering, and managerial positions. He was with IBM, National Semiconductor, and the IAI-Israeli Aerospace Industry. From 1999 to 2009, he was with Intel. From 2011 to 2015, he was an Associate Visiting Professor with the Electrical Engineering Faculty, Technion–Israel Institute of Technology. He is currently an Associate Professor with the Engineering Faculty, Bar-Ilan University, Israel. He is interested in VLSI circuits and systems design optimization and combinatorial optimization.