# Mixed Allocation of Adjustable Delay Buffers Combined with Buffer Sizing in Clock Tree Synthesis of Multiple Power Mode Designs

Kitae Park[1] Geunho Kim[1] Taewhan Kim[1,2]
*pgt@snucad.snu.ac.kr* *ghkim@snucad.snu.ac.kr* *tkim@snucad.snu.ac.kr*
[1]School of Electrical and Computer Engineering, Seoul National University, Seoul Korea
[2]Nano Systems Institute (NSI), Seoul National University, Seoul, Korea

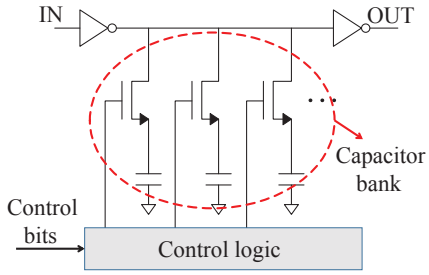
***Abstract*—Recently, many works have shown that adjustable delay buffer (ADB) whose delay is adjustable dynamically can effectively solve the clock skew variation problem in the designs with multiple power modes. However, all the previous works of ADB allocation inherently entail two critical limitations, which are the adjusted delays by ADB are always increments and the low cost buffer sizing has never been or not been primarily taken into account. To demonstrate how much overcoming the two limitations is effective in resolving the clock skew constraint, we characterize the two types of ADBs called CADB (capacitor based ADB) and IADB (inverter based ADB) and show that the adjusted delays by IADB can be decremented, and show that the clock skew violation in some clock trees of multiple power modes can be resolved by applying buffer sizing together with using only a small number of IADBs and CADBs.**

## I. Introduction

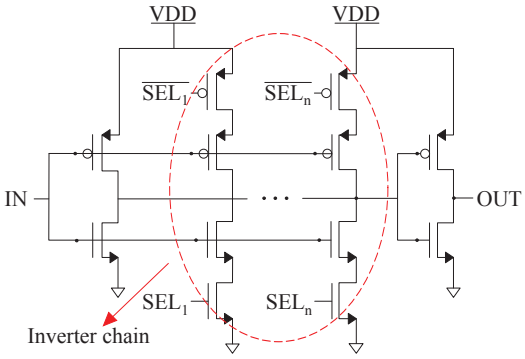
Considerable research effort on the clock tree optimization, for example, clock topology generation, clock routing, clock buffer insertion and sizing, and wire sizing has been made to optimize the clock skew. Those works have a common assumption that the synthesized clock tree is to be used in designs of a single voltage mode condition. For designs of multiple voltage or power mode designs, the signal delay on a clock path may vary as the power mode applied changes. This implies that a clock tree synthesized to meet the clock skew constraint in a power mode may not meet the clock skew constraint in other power modes. Even if the existing works may take into account the clock skew constraint for all power modes, it is very likely that the synthesized clock tree requires a long wirelength or there exist no clock trees that meet the clock skew constraint for all power modes.

On the other side, post-silicon tuning (e.g., [1]–[4]) such as inserting adjustable delay buffers (ADBs) is a widely accepted strategy to handle the timing problem caused by the process and environment variations. Since the delay of an ADB can be adjusted by the delay control inputs, the clock skew variation caused by process variation can be tuned by properly inserting ADBs after manufacturing is completed. The idea of using ADBs in the clock tree of multiple power modes is to replace some of normal clock buffers with ADBs so that the clock skew constraint on all power modes should be satisfied. The ADB allocation problem has been intensively investigated by many works [5]–[9]. Our work overcomes two fundamental drawbacks of the previous works, which are always using ADBs with delay increment only and no attempt to applying (the low cost) buffer sizing to replace the role of ADBs.

This work was supported by Basic Science Research program through NRF grant (No.2011-0029805) in Korea, CISS of Global Frontier project by MSIP (CISS 2011-0031863) in Korea, ITRC program of NIPA by MSIP (NIPA-2013-H0301-13-1011) in Korea, and Samsung Electronics company.




(a) Capacitor based ADB. Delay can increase as more capacitors in the capacitor bank are switched on.




(b) Inverter based ADB. Delay can decrease as more inverters in the inverter chain are switched on.

Fig. 1: The logic structures of ADB implementation: (a) capacitor based ADB (CADB) [8], [10]; (b) inverter based ADB (IADB) [7], [11].

## II. Charasterzation of Delay Adjustable Buffers

Two widely used structures of ADBs are the capacitor based ADB (CADB) in Fig. 1(a) and the inverter based ADB (IADB) in Fig. 1(b). A CADB in Fig. 1(a) is composed of two

978-3-9815370-2-4/DATE14/©2014 EDAA

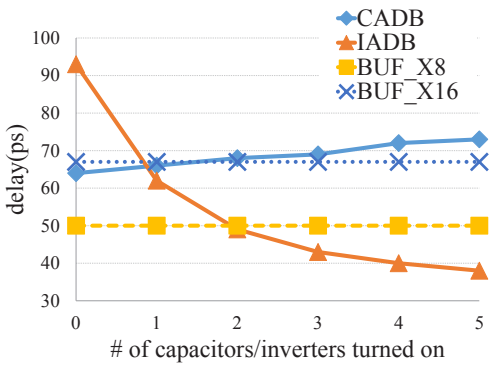



Fig. 2: The delay ranges of CADB, IADB, and normal buffers.

inverters, a capacitor bank, and a capacitor bank controller. If the capacitor bank contains uniform sized capacitors, the CADB delay is linearly proportional to the number of capacitors activated in the bank. That is, *as more capacitors are activated, the CADB delay will increase* due to the more output load of the leftmost transistor. On the other hand, an IADB in Fig. 1(b) is composed of two inverters and a set of chained inverters with SEL pins. The SEL signals provide several driving modes, by which the IADB delay is adjusted. For example, when all the SEL signals are set to logic 0, the inverters connected to the corresponding SEL pins are turned off. Thus, only the leftmost and the rightmost inverters with no SEL pins are turned on and drive output. The delay of ADB can be adjusted by setting some of the SEL signals to logic 1 to activate their chained inverters; Unlike to the CADB, *as more SEL signals are enabled, the IADB delay will decrease* due to the more wide distribution of charge and discharge of current.

We preformed HSPICE simulation for the implementations of CADB and IADB in Fig. 1 together with two normal buffers BUF_X8 and BUF_X16 in 45nm Nangate Open Lell Library. Fig. 2 shows the delay characteristics of CADB, IADB, and the two buffers. It is observed that unlike the delay of the buffers, the CADB delay linearly increases as the number of capacitors turned on increases, whereas the IADB delay decreases as the number of chained inverters turned on increases. The curves in Fig. 2 clearly show that rather than using CADBs only to optimize the clock skew in the design of multiple power modes, carefully using a mixture of IADBs and buffer sizing as well as CADBs will provide a better quality of design, reducing the design overhead significantly.

## III. Mixed ADB Allocation Algorithm Combined with Buffer Sizing

**ADB-based clock skew optimization problem**: *Given an initial buffered clock tree $\mathcal{T}$, power modes $m_1$, $m_2$, $\cdots$, and $m_K$, clock signal arrival time information on all power modes, and clock skew bound, apply CADB allocation, IADB allocation, and buffer sizing to the nodes in $\mathcal{T}$ such that the following quantity of $\Delta A_{tot}$ is minimized:*

$$\Delta A_{tot} = area(CADB) + area(IADB) + area(BS) \quad (1)$$

*where $area(CADB)$, $area(IADB)$, and $area(BS)$ represent the increases of areas by the allocation of CADBs, the allocation of IADBs, and the buffer sizings, respectively while the clock skew constraint is satisfied on every power mode.*

We solve the ADB allocation and buffer sizing problem step-by-step based on the following observations: (1) Since as stated in [8], IADB is suited for coarse-grained delay adjustment while CADB is suited for fine-grained delay adjustment, which is also validated from the data in Fig. 2, which shows that IADB has relatively sharp delay curve (with delay adjustment from -12ps to +42ps) than that of CADB (with delay adjustment from +14ps to +23ps), we place higher priority into IADB allocation than CADB allocation to reduce long clock signal delays in some power modes by using a small number of IADBs first; (2) Since buffer sizing has a relatively much simpler design complexity (e.g., no control lines and switching logic) than that of ADB allocation and requires much less area overhead, we place higher priority into buffer sizing than IADB and CADB allocations.

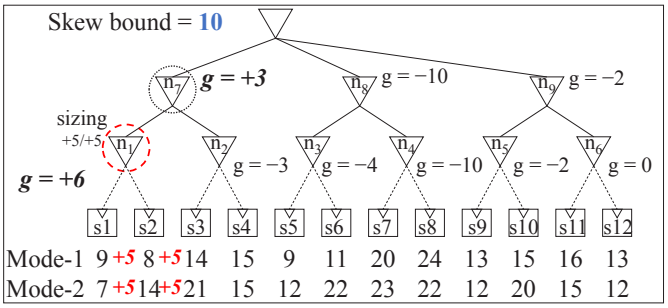
Based on the above observations, our proposed ADB allocation algorithm combined with buffer sizing, called ADB-mix, performs the following four steps to minimize $\Delta A_{tot}$ in $Eq.(1)$: (Step 1) buffer sizing is applied to the initial buffered clock tree. If the clock skew constraint is satisfied, ADB-mix returns the result and stops the execution: (Step 2) IADB allocation is applied to the clock tree obtained in Step 1. If the clock skew constraint is met during the IADB allocation, ADB-mix returns the result and stops the execution; (Step 3) CADB allocation is applied to the clock tree obtained in Step 2. (Step 4) ADB-mix computes the final delay increments or decrements for each clock node on which both of an IADB and a CADB have been allocated, and determines the type and size of an ADB to be finally inserted to the node.

• **Step 1** (*Buffer sizing to minimize $area(BS)$ in $Eq.(1)$*): There exist a number of noticeable works in the literature that addressed the buffer sizing problem to minimize clock skew (e.g., selecting optimal buffers from buffer library using dynamic programming or determining buffer sizes for minimum delay/power zero clock skew) in designs of single power mode. However, it is unlikely that the existing works can be applied to the designs of multiple power modes since minimizing the clock skew in a particular power mode by sizing buffers does affect the clock skews in the other power modes. In this work, we propose a greedy algorithm: For each buffer in node $n_i$ in the clock tree, we attempt to resize the buffer by using a buffer, $buf_j$, in the buffer library and compute a gain $g(\cdot)$:

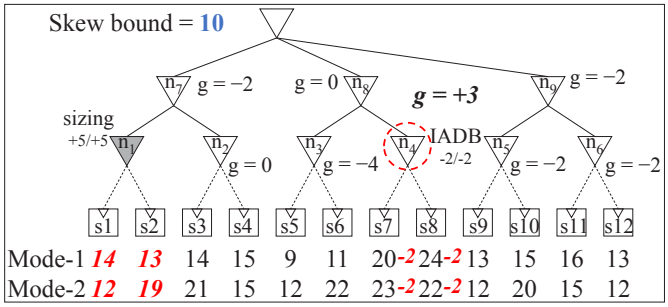
$$g(n_i, buf_j) = \sum_{m_i \in M} (sk(m_i) - sk'(m_i))/BS(n_i, buf_j) \quad (2)$$

where $M$ is the set of power modes, $sk(m_i)$ and $sk'(m_i)$ are respectively the clock skews after and before the buffer sizing in $m_i$, and $BS(n_i, buf_j)$ is the amount of area increase by sizing the buffer in $n_i$ with $buf_j$.

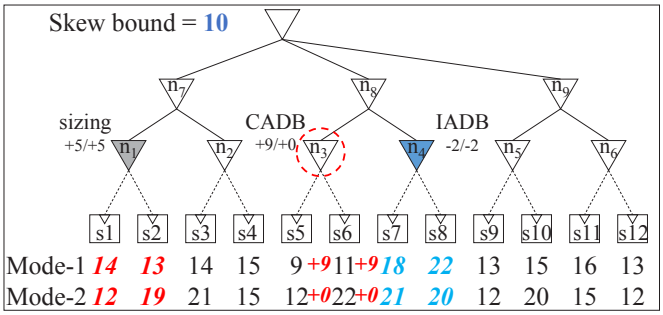
Then, we select, among all the possible buffer sizings that reduce the total clock skew in the clock tree, the buffer sizing which has the least value of gain $g(\cdot)$ if $BS(n_i, buf_j) < 0$ or has the largest value of gain $g(\cdot)$ if such a buffer sizing of $BS(n_i, buf_j) < 0$ does not exist, and perform the sizing, i.e., replacing the buffer in $n_i$ with $buf_j$. We repeat this iteration until the reduced clock skew meets the skew constraint or no further buffer sizings can be applied. For example, the values



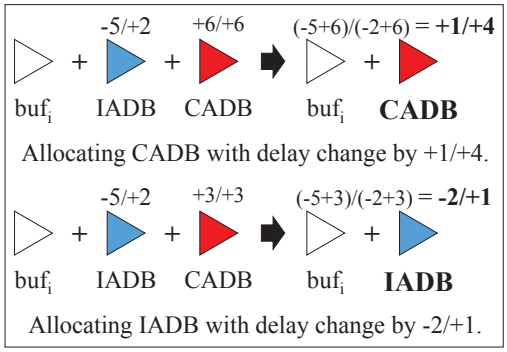

(a) Step 1: sizing the buffer in $n_1$ reduces clock skew from 16 (= 24-8) to 15 (= 24-9) in *Mode-1* and from 16 (= 23-7) to 11 (= 23-12)) in *Mode-2*.




(b) Step 2: allocating an IADB to $n_4$ in the tree in (a) reduces clock skew from 15 (= 24-9) to 13 (= 22-9) in *Mode-1* and from 11 (= 23-12) to 10 (= 22-12) in *Mode-2*.




(c) Step 3: applying conventional CADB allocation algorithm [9] to (b) allocates an CADB to $n_3$, reducing the clock skew from 13 (= 22-9) to 9 (= 22-13) in *Mode-1* and unchanging the clock skew in *Mode-2*.




(d) Step 4: examples of determining the final ADB types for overlapped ADB allocation in a node of clock tree.

Fig. 3: An illustration of the step-by-step procedure of our proposed ADB allocation algorithm ADB-mix: (a) Step 1: applying buffer sizing to reduce clock skew; (b) Step 2: allocating IADBs to reduce the clock skew by decreasing the latest clock delay; (c) Step 3: allocating CADBs to reduce the clock skew by increasing the earliest clock delay; (d) Step 4: finalizing the ADB types in the nodes where both of IADB and CADB are allocated.

of gain $g$ for each node in a clock tree is shown in Fig. 3(a), in which the resizing at $n_1$ with gain $g = +6$ is selected, increasing the times at FFs $s_1$ and $s_2$ to 14 and 13 in *Mode-1* and to 12 and 19 in *Mode-2*, respectively. The increase of earliest delay leads to reducing the clock skew from 16 to 15 in *Mode-1* and from 16 to 11 in *Mode-2*.

• **Step 2** (*Allocating IADBs to minimize* $area(IADB)$ *in* $Eq.(1)$): The IADB allocation problem is to allocate a minimal number of IADBs, taken from the library of IADBs, to satisfy the clock skew constraint. We use a greedy algorithm, which is essentially identical to that in Step 1. The only difference is the gain function $g(\cdot)$:

$$g(n_i, IADB_j) = \sum_{m_i \in M} (sk(m_i) - sk'(m_i))/area(IADB_j) \quad (3)$$

where $sk(m_i)$ and $sk'(m_i)$ are respectively the clock skews after and before the allocation of $IADB_j$ in $m_i$, and $area(IADB_j)$ is the amount of area increase by the insertion of $IADB_j$. For example, in Fig. 3(b) the allocation of IADB at $n_4$ reduces the times at $s_7$ and $s_8$, resulting in the clock skew from 15 to 13 in *Mode-1* and from 11 to 10 in *Mode-2*.

• **Step 3** (*Allocating CADBs to minimize* $area(CADB)$ *in* $Eq.(1)$): Since there are many existing algorithms that have addressed the CADB allocation, we can use any of the conventional algorithms. In our flow, we use ADB-Pullup [9] which optimally allocates CADBs for each power mode. In our context, ADB-Pullup accepts as input the clock tree which may have buffers that have been sized in Step 1 and IADBs that have been allocated in Step 2. The role of ADB-Pullup is just to determine if a delay *increment* is required at each node in the clock tree in every power mode and compute the value of delay increment if needed. For example, allocating one CADB at $n_3$ in Fig. 3(c) reduces the clock skew from 13 to 9 in *Mode-1* and unchanges the clock skew in *Mode-2*, meeting the clock skew bound on all power modes.

• **Step 4** (*Refinement by determining ADB types*): This step is to resolve the overlapped allocation of IADB and CADB in a node of clock tree. We examine the delay decrements and increments computed in each node in Steps 2 and 3, and deterime the type of ADB to be finally allocated in the node. For example, Fig. 3(c) does not have nodes with overlapped ADB allocations, thus, the ADB allocation are completed. However, for the overlapped allocations as depicted in Fig. 3(d), it is required to determine the ADB types: the upper one in Fig. 3(d) shows an example where CADB with delay increment should be finally allocated; the lower one in Fig. 3(d) shows an example where IADB with delay decrement should be finally allocated.

## IV. Experimental Results

We have implemented our proposed algorithm ADB-mix and the algorithm ADB-Pullup in [9] using C++ on a system with 8 2.5Ghz Intel Xeon CPU with 8GB memory. All input clock trees are generated using *Synopsys IC Compiler* with 45nm Nangate Open Cell Library. We have tested the benchmark circuits in ISPD'09 clock network synthesis contest. For each benchmark, we partitioned it into 6∼10 power subdomains, each of which can operate in two different voltage levels: 0.95V and 1.1V. We assumed to use four different power

TABLE I: Comparison of ADB allocation results produced by our ADB-mix and ADB-Pullup [9].

| Circuits | #FF | #Buf | Skew (orig.) | Latency (orig.) | Skew bound | #ADB [9] | Latency [9] | #ADB (ADB-mix) | Latency (ADB-mix) | Red. of #ADBs |
|---|---|---|---|---|---|---|---|---|---|---|
| F11 | 121 | 159 | 144ps | 3010.7ps | 90ps | 30 | 3046.1ps | 21 | 2630.1ps | 30.0% |
| | | | | | 110ps | 14 | 2968.1ps | 14 | 2624.1ps | 0.0% |
| | | | | | 130ps | 3 | 2968.1ps | 2 | 2744.1ps | 33.3% |
| F12 | 117 | 143 | 126ps | 2528.2ps | 90ps | 33 | 2684.7ps | 12 | 2163.7ps | 63.6% |
| | | | | | 110ps | 31 | 2660.7ps | 11 | 2152.7ps | 64.5% |
| | | | | | 130ps | 30 | 2616.7ps | 1 | 2476.7ps | 96.7% |
| F21 | 117 | 156 | 126ps | 2969.2ps | 90ps | 31 | 3033.1ps | 10 | 2568.1ps | 67.7% |
| | | | | | 110ps | 27 | 3013.1ps | 9 | 2566.1ps | 66.7% |
| | | | | | 130ps | 9 | 2998.1ps | 2 | 2719.1ps | 77.8% |
| F22 | 91 | 90 | 126ps | 2043.0ps | 90ps | 27 | 2197.3ps | 14 | 1769.3ps | 48.2% |
| | | | | | 110ps | 15 | 2128.3ps | 12 | 1757.3ps | 20.0% |
| | | | | | 130ps | 15 | 2128.3ps | 10 | 1745.3ps | 33.3% |
| F31 | 273 | 328 | 198ps | 4236.9ps | 90ps | 39 | 4164.1ps | 28 | 3719.1ps | 28.2% |
| | | | | | 110ps | 23 | 4157.1ps | 17 | 3718.1ps | 26.1% |
| | | | | | 130ps | 14 | 4157.1ps | 11 | 3706.1ps | 21.4% |
| F32 | 190 | 262 | 198ps | 4144.0ps | 90ps | 41 | 4313.0ps | 26 | 3779.0ps | 36.6% |
| | | | | | 110ps | 39 | 4233.0ps | 16 | 3782.0ps | 59.0% |
| | | | | | 130ps | 39 | 4233.0ps | 11 | 3768.0ps | 71.8% |
| F33 | 209 | 257 | 162ps | 4147.9ps | 90ps | 23 | 4069.9ps | 4 | 3830.9ps | 82.6% |
| | | | | | 110ps | 12 | 4029.9ps | 4 | 3826.9ps | 66.7% |
| | | | | | 130ps | 5 | 4035.9ps | 4 | 3826.9ps | 20.0% |
| F34 | 157 | 210 | 198ps | 4075.3ps | 90ps | 44 | 4305.6ps | 11 | 3616.6ps | 75.0% |
| | | | | | 110ps | 26 | 4228.6ps | 8 | 3616.6ps | 69.2% |
| | | | | | 130ps | 12 | 4208.6ps | 8 | 3616.6ps | 33.3% |
| F35 | 193 | 239 | 198ps | 4268.9ps | 90ps | 37 | 4197.4ps | 12 | 3649.4ps | 67.6% |
| | | | | | 110ps | 37 | 4197.4ps | 9 | 3660.4ps | 75.7% |
| | | | | | 130ps | 2 | 4153.4ps | 1 | 4126.4ps | 50.0% |
| Ratio | | | | | | 1 | 1 | **0.437** | **0.885** | **56.3%** |

modes and found the worst clock skew over the whole power modes. We have also assumed that each ADB can be adjusted with a granularity of 10ps. We rounded off the delay values obtained by ADB-mix and the algorithm ADB-Pullup in [9] to reflect the discrete delay adjustment of each ADB.

Table I summarizes the comparison of the results produced by ADB-Pullup [9] and our ADB-mix. The seventh and eighth columns show the number of ADBs (i.e., CADBs) to meet skew bound and the maximum clock latency used by ADB-Pullup [9], respectively. The last three columns show the number of ADBs (i.e., CADBs and IADBs) and the maximum clock latency used by ADB-mix, and reduction of ADBs (in percent) by ADB-mix over that by ADB-Pullup [9]. We can clearly see that the ADB reduction is consistent for all testcases. The ADB reduction by ADB-mix is 56.3% on average for clock skew bound of 90ps$\sim$130p with even much shorter clock latencies, i.e., 11.5% shorter than that by ADB-Pullup. For example, for F21, 31 ADBs with latency of 3033ps are required to meet skew constraint of 90ps under all power modes for ADB-Pullup [9] whereas ADB-mix uses 21 less ADBs with 465ps shorter latency to meet the skew constraint. It is also observed that the area increase by buffer sizing is very minimal, less than 0.1%. However, its impact is significant. For example, see the allocation of 1 or 2 ADBs after buffer sizing in F12, F21, and F22 in Table I.

## V. Conclusion

This work proposed a new algorithm to solve the problem of clock skew optimization using adjustable delay buffers (ADBs) under multi-voltage design environment. This work showed that the use of mixed types of ADB as well as buffer sizing could lead to much economical designs.

## References


[1] S. Hu and J. Rabaey, "Unified adaptivity optimization of clock and logic signals," *ICCAD,* 2007.

[2] V. Khandelwal and A. Srivastave, "Variability-driven formulation for simultaneous gate sizing and post-silicon tunability allocation," *ISPD,* 2007.

[3] J. -L. Tsai, L. Zhang, and C. Chen, "Statistical timing analysis driven post-silicon-tunable clock-tree synthesis," *ICCAD,* 2005.

[4] E. Takahashi, Y. Kasai, M. Murakawa, and T. Higuchi, "A post-silicon clock timing adjustment using genetic algorithms," *Proc. of Symposium on VLSI circuits,* pp. 13-16, 2003.

[5] Y. -S. Su, W. -K. Hon, C. -C. Yang, S. -C Chang, and Y. -J Chang, "Value assignment of adjustable delay buffers for clock skew minimization in multi-voltage mode designs," *ICCAD,* 2009.

[6] Y. -S. Su, W. -K. Hon, C. -C. Yang, S. -C Chang, and Y. -J Chang, "Clock skew minimization in multi-voltage mode designs using adjustable delay buffers," *IEEE TCAD,* Vol. 29, December 2010.

[7] K.-Y. Lin, H.-T. Lin, and T.-Y. Ho, "An efficient algorithm of adjustable delay buffer insertion for clock skew minimization in multiple supply voltage designs," *ASPDAC,* 2011.

[8] K.-H. Lim and T. Kim, "An optimal algorithm for allocation, placement, and delay assignment of adjustable delay buffers for clock skew minimization in multi-voltage mode designs," *ASPDAC,* 2011.

[9] J. Kim, D. Joo, and T. Kim, "An optimal algorithm of adjustable delay buffer insertion for solving clock skew variation problem," *DAC,* 2013.

[10] A. Kapoor, N. Jayakumar, and S. P. Khatri, "A novel clock distribution and dynamic de-skewing methodology," *ICCAD,* 2004.

[11] G. N. Roberts, "Adjustable buffer driver," Patent, 1994, US 5361003.