Statistical Leakage Power Minimization Using Fast Equi-Slack Shell Based Optimization

Xiaoji Ye  
Department of ECE  
Texas A&M University  
College Station, TX 77843  
yexiaoji@neo.tamu.edu

Yaping Zhan  
Advanced Micro Devices, Inc.  
Austin, TX 78759  
yaping.zhan@amd.com

Peng Li  
Department of ECE  
Texas A&M University  
College Station, TX 77843  
pli@neo.tamu.edu

ABSTRACT
Leakage power is becoming an increasingly important component of total chip power consumption for nanometer IC designs. Minimization of leakage power unavoidably enforces the consideration of the key sources of process variations, namely transistor channel length and threshold variations, since both have a significant impact on timing and leakage power. However, the statistical nature of chip performances often requires the use of expensive statistical analysis and optimization techniques in a leakage minimization task, contributing to high computational complexity. Further, the commonly used discrete cell libraries bring specific difficulty for design optimization and render pure continuous sizing and $V_T$ optimization algorithm suboptimal. In this paper, we present a fast yet effective approach to statistical leakage power reduction via gate sizing and multiple $V_T$ assignment. The proposed technique achieves the runtime efficiency via the use of the novel concept of equi-slack shells and performs fast leakage power reduction on the basis of shells while maintaining the timing yield. When combined with a finer grained gate-based post tuning step, the presented technique achieves superior runtime efficiency while offering significant leakage power reduction.

Categories and Subject Descriptors:
B.6.3 [Hardware]: Design Aids—Optimization

General Terms:
Algorithms, Design, Performance, Verification.

Keywords:
Leakage, Statistical Optimization, Equi-slack shell.

1. INTRODUCTION
With integrated circuit process technologies scaling down to the nanometer range and the low-threshold voltage devices being widely used for high-performance circuits, leakage power is becoming increasingly significant in the total chip power consumption. The predicted leakage power is projected to account for about half of the total chip power within the next few technology generations [1].

At the same time, due to the shrinking of device geometries, it is becoming increasingly difficult to control the fabrication of critical device parameters. Growing process variability has been observed such as the effective channel length variability due to line edge roughness (LER) and threshold voltage variability due to random doping effects. It has been shown that 30% process variations can cause up to 20X leakage power variations [2]. This makes the deterministic power optimization at a prescribed process corner no longer meaningful. Hence, it is imperative to consider process variations in any power optimization task.

Statistical circuit optimization has become an active topic of research in recent years. Sensitivity-based incremental gate sizing techniques have been proposed to improve timing yield while minimizing certain cost function (e.g. chip area) [3, 4]. An unconstrained nonlinear optimization is formulated and solved based on statistical power and delay gradient computation in [5]. A robust gate sizing algorithm based on approximated geometric programming formulation is proposed to optimizing the timing yield in [6]. A timing yield constrained statistical power optimization algorithm is presented in [7] where second order cone programming is employed to distribute slacks at each step of gate sizing.

These algorithms offered new directions to robust circuit design. However, when applied in practice, these approaches may still suffer from the following drawbacks. First, since the convex optimization algorithms are yet able to handle true statistical constraints, statistical timing or power constraints must be converted to deterministic ones before being applied to the optimization solver. This requires unrealistic assumptions such as the constraints of all the gates are statistically independent. Moreover, simplified statistical models are often used, which can introduce inaccuracy as small process variations can result in huge leakage variations. Second, these sizing techniques are continuous in nature and cannot well handle discrete cell libraries. As mixed-integer nonlinear programming is very computationally expensive, continuous formulations are adopted instead. As a result, gate size rounding has to be applied that destroys the global optimum solution obtained in the inner loop of the algorithm. Third, as the circuit size increases, the convex optimization tools may become the bottleneck of both runtime and memory. In gradient-based approaches, heuristics exist to compute statistical sensitivities efficiently, however, at the cost of incurred inaccuracy. In most of the approaches, a new SST A run is required after each optimization step to update the circuit slacks, leading to potential
high computational complexity.

In this paper, we propose a fast yet effective statistical leakage power optimization algorithm based on the novel concept of equi-slack shells. Our algorithm is based upon delay, leakage and power-delay models fully parameterized in process variations. During the optimization phase, interconnected gates with same or close slacks are partitioned into one group, or an equi-slack shell, and sizing is conducted on an equi-slack shell basis starting from an initial design meeting the timing yield. The use of equi-slack shells simplifies the slack update and relaxes the need for a new statistical static timing analysis (SSTA) run after each optimization step, leading to significant runtime speedup of the algorithm. Our leakage minimization algorithm sequentially works on equi-slack shells with descending slack values and sizes down the gates or switches to high $V_t$ with the best statistical power saving in a discrete manner while maintaining the timing yield. Our experiments have shown superior runtime performance of the proposed approach with significant reduction of leakage power.

2. EQUI-SLACK SHELLS

The proposed leakage power minimization algorithms starts from an initial design with met timing yield and finds timing non-critical candidate gates that provide the largest leakage power saving for downsizing or high $V_t$ assignment. Given the inherent nonlinear nature and complexity of gate sizing problem, the following aspects are being addressed in the proposed approach:

- It is desired to facilitate power saving by absorbing available slacks with small increment so as to avoid trapping at a local minimum. Hence, a systematic and “smooth” optimization strategy is needed.
- Efficiency of the sizing algorithm will be improved if multiple optimization gate candidates can be identified and optimized simultaneously.
- Fast statistical slack update schemes relax the need of a new SSTA run after each optimization step and lead to overall efficiency of the optimization task.

![Equi-slack shells](image)

Figure 1: Equi-slack shells.

In this work, we achieve the above objectives simultaneously via the concept of equi-slack shells as shown in Fig. 1, where for simplicity of illustration, deterministic slacks are used. An equi-slack shell consists of topologically connected logic gates with identical or similar slack values. In Fig. 1, four shells with different slack values are shown. In an ideal sense, a leakage minimization task should try to achieve maximum leakage power savings when all the gates in the circuit are optimized to the same slacks limited by the timing target. Our algorithm sequentially works on shells in a descending slack order and within each shell best power saving gate candidates are optimized guide by power-delay sensitivities. For example, Shell C with slack 7 will be the first optimization target. When the gates in shell C are optimized for leakage power reduction, it is guaranteed by our algorithm that the new slack after the optimization would be at least 5, which is the second largest slack value (shell D). If new slack for shell C truly becomes 5 after optimization, then shells C and D are combined to form one equi-slack shell. This process continues until all the gates form a single equi-slack shell with the minimum slack required by timing target and no power improvement can be further made on the shell basis.

The benefits brought by equi-slack shells are twofold. First, by optimizing shells one at a time in a descending slack order and constraining the slack reduction within each shell, the timing target can be easily ensured. Second, partitioning gates into shells according to the slack facilities the desired “smooth” incremental optimization. That is, the reduction of slack within each shell is constrained by its shell slack boundary (i.e. the next largest shell slack). This technique avoids dramatic change of gate slacks and allows regrouping of shells after each optimization. The subsequent optimization can be then performed based on merged (larger) shells, making it possible to find best optimization targets within a wide scope and leading to an improved optimality.

As summarized in Algorithm 1, our algorithm starts by applying an initial SSTA run to the given design to compute the statistical slacks of all the gates. Our SSTA is based on the widely used first-order parameterized algorithm [8, 9] where each gate delay is expressed as a linear function of a set of $N$ process variables, $\dot{X} = [X_1, X_2, \cdots, X_N]^T$, following a multivariate normal distribution. The leakage power variation is described using log-normal distributions in the same set of variables [7, 5]. The initial SSTA can be used to find logic gates with the largest slack values (bounded by mean and variance) and these gates are then used as the seeds to create an initial set of single-gate equi-slack shells.

At each optimization step, the shell with the best slack value is chosen and the slacks of the gates within the shell are reduced, but constrained by an upper bound, which is equal to the difference between the largest and the second largest shell slack values. After each within shell optimization, the statistical slacks of the gates in the shell are efficiently updated. The optimized shell is expanded from the boundary of the shell and new gates are absorbed into the shell based on similarity between slacks values in a statistical sense. Expansion of equi-slack shells may invoke merging between shells with close slack values. This process repeats until no slack can be reduced on a shell basis without violating the timing yield. Finally, a sensitivity-based optimization technique is applied on an individual gate basis.

3. MANIPULATION OF EQUI-SLACK SHELLS

To allow efficient application of the leakage minimization algorithm, equi-slack shells must be manipulated efficiently to perform basic expansion and merging operations. We use simple array/link list type data structures to represent a shell, as shown in Fig. 2. The storages for internal gates,
i.e. those do not have any outside shell fan-ins and fan-outs, and boundary gates are kept separately. The expansion of a shell always starts from the wavefront defined by boundary gates and it may trigger the merging of the two shells with close slack values if they are brought together by expansion, as shown in Fig. 3.

<table>
<thead>
<tr>
<th>Algorithm 1 Equi-Slack based statistical leakage optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Perform an initial SSTA analysis to obtain the statistical slacks for all the gates.</td>
</tr>
<tr>
<td>2: Choose the gates with the largest mean - 3σ slack value and use these gates as seeds to create single-gate equi-slack shells. Insert these equi-slack shells into a priority queue.</td>
</tr>
<tr>
<td>3: repeat</td>
</tr>
<tr>
<td>4: Pop-up the equi-slack shell with the best ( S_r = \text{mean} - 3\sigma ) slack value.</td>
</tr>
<tr>
<td>5: Reduce the leakage by reducing the slacks of the gates within the shell with the slack reduction bounded by the difference between the largest and the second largest shell slack values.</td>
</tr>
<tr>
<td>6: Update the slacks of all gates in the shell.</td>
</tr>
<tr>
<td>7: Expand the selected shell and perform shell merging if necessary.</td>
</tr>
<tr>
<td>8: until the largest shell slack is below a user defined threshold.</td>
</tr>
<tr>
<td>9: Post tune the circuit by conducting sensitivity-based individual gate optimization.</td>
</tr>
</tbody>
</table>

4. LEAKAGE OPTIMIZATION WITHIN EQUI-SLACK SHELLS

To reduce leakage power, circuit designers would down-size the gates and/or switch low Vt devices to high Vt devices. This power saving is at the cost of possible timing yield loss due to the slower gates being used. Therefore, the problem to solve is to minimize the total leakage power with the timing yield satisfied. The general idea is to reduce the power of those gates that lie on non-critical paths, whose delays have less impact on the circuit timing yield. In the following, we will first present our equi-slack shell based optimization strategy.

4.1 Power-Delay sensitivity

In this paper, we will use the power-delay sensitivity \([1]\) as our optimization criterion. Suppose we have a hypothetical 2-input NAND gate with two possible sizes and high/low Vt configurations. The configurations are sorted in an order that the delay increases while the power consumption decreases. For each of the configuration \(i\) (size 1/low Vt, size 1/high Vt, size 2/low Vt, and size 2/high Vt), our power-delay sensitivity is defined as:

\[
Sen_i = \frac{\Delta P_i}{\Delta D_i},
\]

where \(\Delta P_i\) and \(\Delta D_i\) are the corresponding leakage power difference and delay difference between this \(i\)-th configuration and its next configuration, i.e. if we switch this configuration to the next in our library, how much power saving we can obtain with the amount of delay sacrifice. The sensitivity function will help us decide the best power-delay configuration for a particular gate given a slack margin.

4.2 Levelization

We use a compact DAG to represent a combinational circuit. In our compact DAG representation, a single fan-in and fanout path (e.g. an inverter chain) is grouped together and represented using a super node. Let us consider the situation shown in Fig. 4, where three equi-slack shells are shown with slack values \(\text{Slack}_{sh,1} > \text{Slack}_{sh,2} > \text{Slack}_{sh,3}\). Let us assume \(\text{Slack}_{sh,1}, \text{Slack}_{sh,2}\) and \(\text{Slack}_{sh,3}\) are the three largest slacks in the circuit and hence shell A is the current optimization target. Our goal is to efficiently identify a set of power optimal gates in shell A for downsizing or high-Vt assignment. Furthermore, the upper bound for slack reduction within shell \(\text{A}\) is \(\Delta S_{\text{max}} = \text{Slack}_{sh,1} - \text{Slack}_{sh,2}\), where \(\text{Slack}_{sh,2}\) is the second shell slack. Our key observation is that gates located at the same topological level can be optimized independently. For example, in Fig. 4, if we reduce power by switching G4 to a lower power higher delay configuration, such as low-Vt to high-Vt with the slack margin \(\Delta S_{\text{max}}\), no optimization may be further performed on G6 without violating the timing target. However, G4 and G5 can be optimized with the same slack margin for better...
power solution while satisfying the timing target. The rationale behind it is that the node slack is only determined by the latest arrival time of all its fan-ins. Hence, we conclude that the slack margin can be shared by all the gates at the same level, but excluded between different levels.

Then we define the level sensitivity $Sen_{lev,k}$ for each level $k$

$$Sen_{lev,k} = \sum Sen_{k,i},$$

where $Sen_{k,i}$ is the power-delay sensitivity of $i$-th gate at level $k$. For a super node such as gate G3 in Fig. 4, the super node’s sensitivity is chosen to be the largest gate power-delay sensitivity in the chain. To maximally reduce the leakage power, we choose the level with the largest level sensitivity and optimize all the gates in that level independent with the same slack margin $\Delta S_{max}$. In order to consider the important impacts of process variations on leakage power, we choose a level with the largest sensitivity in a statistical fashion as follows.

For the purpose of statistical power-delay sensitivity computation, we use the first-order expressions to represent the variations of delays and slacks, and a first-order parameterized log-normal statistical model to represent the leakage power. Consequently, we use the same expressions to present the variations of $\Delta P_i$ and $\Delta S_i$ in (2) for gate i:

$$\Delta P_i = a_{00}e_{\gamma_1}^{+} X, \quad \Delta D_i = d_{00} + d_{11}^{+} X,$$

where $a_{01} = [a_{01,1}, \ldots , a_{01,N}]^T$ and $d_{11} = [d_{11,1}, \ldots , d_{11,N}]^T$. As compared to the exponential dependency of leakage variations on process parameters, the delay variations are relatively small. Hence, we use a similar first-order parameterized log-normal model for the power-delay sensitivities

$$Sen_{k,i} = c_{00}e_{\gamma_1}^{+} X,$$

where $Sen_{k,i}$ is the statistical power-delay sensitivity for the $i$-th gate in level $k$. The power-delay sensitivity of each gate is pre-characterized by SPICE simulation at a one-time cost for a standard cell library, during which the sensitivity coefficients are fitted. Substituting (5) into (3), the sensitivity of level k is obtained by summing up the sensitivities of all the gates in that level

$$Sen_{lev,k} = \sum c_{00}e_{\gamma_1}^{+} X.$$

Theoretically, the sum of log-normal random variables does not have a closed-form distribution. However, it is demonstrated that the sum operation can be well approximated by Wilkinson’s method [10]. Suppose we want to approximate the sum of $T$ log-normals $e^{\sigma_1}, \ldots , e^{\sigma_T}$, where $1 \leq i \leq T$, and $Y_i \sim N(m_i, \sigma_i)$ with a new log-normal random variable $e^{\sigma_Z}$. The mean and variance of the normal random variable $Z$ are given [10]

$$m_Z = 2\ln(u_1) - \frac{1}{2} \ln(u_2), \quad \sigma_Z^2 = \ln(u_2) - 2\ln(u_1),$$

$$u_1 = \sum_{i=1}^{T} e^{m_i + \frac{\sigma_i^2}{2}}, \quad u_2 = \sum_{i=1}^{T} e^{2(m_i + \sigma_i^2)} + 2 \sum_{j=1}^{T-1} \sum_{i=j+1}^{T} e^{m_i + m_j + (\sigma_i^2 + \sigma_j^2 + 2r_{ij}\sigma_i\sigma_j)/2},$$

where $r_{ij}$ is the correlation coefficient of $Y_i$ and $Y_j$. Since we only do sum approximation on the gates in a level of an equi-slab shell, the quadratic time complexity of the above computation is not a limiting factor in our optimization algorithm. After the level sensitivities are computed, we can compare any two level sensitivities by their means and variances to select the best level for optimization.

### 4.3 Slack update

For each gate under optimization, a gate configuration along its optimal power-delay Pareto front is sought to absorb the given slack margin $\Delta S_{max}$. Proper size rounding will be applied according to the discrete cell library. The use of rounding as well process variations may lead to somewhat different slack reduction between gates in the same level.

Consider the situation shown in Fig. 5, where the third level of the equi-slab shell has the largest level sensitivity and is chosen for optimization. For each gate in this level (G6 and G7), as we downsize it or assign it to high $V’$, we guarantee that the delay increase $\Delta D_i$ is safely below the given the slack margin, e.g., $P(\Delta D_i < \Delta S_{max}) > (1 - \alpha)$, where $\alpha$ is small. Then we determine a safe bound, $\Delta S’_{max} < \Delta S_{max}$ for the maximum slack reduction in this level by computing the statistical max, $D_{max}$, of all the delay increases in this level. The $\Delta S_{max}$ is set to be $\Delta S’_{max} = \mu_{D_{max}} + \gamma \sigma_{D_{max}}$, where $\gamma$ is user defined (e.g. 2.0). $\Delta S’_{max}$ will be adopted for fast slack update.

It is important to note, by adopting the equi-slab shells the slack update is significantly simplified. Since the specified slack margin is not exceeded in optimization (i.e. $\Delta S’_{max} \leq \Delta S_{max}$), only the slacks of the gate within the same shell need to be updated. Two different scenarios exist. First, the slack update of internal gates, i.e. gates whose fan-ins and fan-outs are completely confined within the shell, can be efficiently done. In this case, the slack of each gate can be simply reduced by $\Delta S’_{max}$. However, the same approach can be applied safely to other (boundary) gates but with over-optimism. To clearly see this, let us consider gate G3. Although its fanout gate G6’s delay has been increased, the required arrival time at the output of G3 may be still more constrained by its fanout G10, which is outside of the shell. For any G3’s fanout gate Gk that is outside of the shell, if the following is true

$$RAT_k - D_k < RAT_6 - D_6 - \Delta S’_{max},$$

where $RAT_i$, $AT_i$ and $D_i$ are the required arrival time, ar-
rival time and delay of the $i$-th gate before the optimization, respectively, then reducing gate G3’s slack by $\Delta S'_{\text{max}}$ is overly pessimistic. To fix this problem, we perform an more accurate slack update as:

$$\text{Slack}_3 = \min\{\text{Slack}_3, RAT_6 - D_6 - \Delta S'_{\text{max}}\}. \quad (9)$$

The same slack update is applied to all other boundary gates that are in the fan-in cone of the optimized level and have at least one fanout outside of the shell. Similarly, the slack of gate G12 is updated according to

$$\text{Slack}_{12} = \min\{\text{Slack}_{12}, RAT_{12} - AT_7 - \Delta S'_{\text{max}}\}. \quad (10)$$

The same update is applied to all other boundary gates that are in the fanout cone of the optimized level and have at least one fan-in outside of the shell.

If a boundary gate’s slack reduction is less than $\Delta S'_{\text{max}}$ with a high probability, it will be removed from the shell and forms a new shell with a larger slack. This is assumed to be the case for G3 and G12 in Fig. 5.

5. SENSIVITY BASED POST TUNING

The proposed equi-slack shell based approach is very efficient in runtime. It provides a desired gradual sizing and $V_t$ assignment optimization. The nature of discrete sizing/$V_t$ assignment and process variations nevertheless introduce non-uniformity in slack distribution in an equi-slack shell. Hence, we propose to use a slower but a finer grained gate-based post tuning step to further reduce the leakage power. The aim of this post tuning step is to identify remaining large slacks in the circuit and convert them into further power saving.

Algorithm 3 Gate-based sizing and $V_t$ assignment

1: \textbf{repeat}
2: \hspace{1em} Find a set of $P$ gates with the largest slacks.
3: \hspace{1em} For each chosen gate, compute the statistical power-delay sensitivity.
4: \hspace{1em} Find the gate that corresponds to the best power-delay sensitivity using first-order log-normal models.
5: \hspace{1em} Assign the selected to the next gate configuration to reduce the leakage power.
6: \hspace{1em} Check the timing through a SSTA run.
7: \hspace{1em} Revolve the change if the timing yield is violated.
8: \hspace{1em} \textbf{until} No further leakage power saving.

This gate-based fine grained optimization uses the same statistical power-delay sensitivity as guidance. Once the gate that has a high slack value and the optimal statistical power-delay sensitivity (in the sense of mean and variance of the log-normal model), it is either downsized or assigned to high $V_t$ for leakage reduction. The action is accepted unless that the timing yield is violated, verified through a SSTA run. This process repeats until no further leakage reduction can be made, as outlined in Algorithm 3.

6. EXPERIMENTAL RESULTS

The proposed equi-slack shell based statistical leakage optimization algorithm has been implemented in C++. The delay and leakage models were pre-characterized by using the 45nm predictive technology model [11]. Each gate in the cell library has discrete sizes ranging from 1x to 10x with minimum gate length and dual-$V_t$, resulting a total of 12 configurations. The ranges of the process variations ($\sigma/\mu$) are: $L_{\text{eff}}$: 8%, $V_t$: 5%, and $T_{ox}$: 5%. The algorithm was applied to a set of ISCAS85 benchmark circuits where a TILOS-like algorithm [12] was conducted to provide an initial design (with low $V_t$ assignment) that satisfies the timing yield. Our experiments were conducted on a PC running Linux operating system.

In Table 1, we compare the deterministic optimization and statistical optimizations on leakage reduction ($\mu$W) under the timing yield targets of 99.73% and 95.44%. Both the deterministic optimization and the statistical optimization take the same equi-slack approach and meet the same timing yield target, but by using the nominal and statistical delays/leakages during optimization respectively. The total leakage powers are calculated as the sum of statistical leakages of all the gates in the circuit based on the derived optimal gate configurations resulted in the two optimization runs. In the table, we first show the means and standard deviations of the total leakage power for the initial design, the deterministic optimization and the statistical optimization, respectively (columns 2-7). Then, we show the percentage improvements of leakage mean and standard deviation of the statistical optimization over the initial design and the deterministic optimization (columns 8-11), then finally the runtime of the statistical optimization. Clearly, our statistical optimization algorithm can provide significant leakage power saving over both the initial design and the deterministic optimization. The algorithm runtime is also 3 – 5X times faster compared to the algorithm reported in [7].

In Fig. 6, the leakage vs. delay plots under 99.73% timing yield of c499 are shown for both the deterministic and statistical optimizations. Clearly, the statistical optimization produces more optimal leakage reduction. The same observation can be made in Fig. 7, where the leakage vs. timing yield curves for c1355 are shown. To see the impacts of deterministic and statistical optimizations on the leakage power distribution, the leakage power PDFs of c499 are shown in Fig. 8. In Fig. 9, we investigate the mean value of power-delay product as a function of delay target under 99.73% timing yield.

7. CONCLUSION

We have presented a novel equi-slack shell based statistical leakage optimization technique. The use of the equi-slack shells significantly speedups circuit optimization and leads to superior runtime of the algorithm. Our experimental results have shown that significant leakage power can be achieved by the proposed technique.
<table>
<thead>
<tr>
<th>CKT</th>
<th>#Gates</th>
<th>W/O Opt</th>
<th>Det Opt</th>
<th>Stat Opt</th>
<th>Impr Init</th>
<th>Impr Det</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c432/160</td>
<td>16.5</td>
<td>9.2/8.6</td>
<td>4.3/3.7</td>
<td>7.9/7.2</td>
<td>3.4/3.1</td>
<td>52.2/56.8</td>
<td>99.73%</td>
</tr>
<tr>
<td>c499/202</td>
<td>19.5</td>
<td>13.5/12.5</td>
<td>5.6/5.3</td>
<td>35.9/35.7</td>
<td>35.7/35.9</td>
<td>17.1/16.7</td>
<td>170/167</td>
</tr>
<tr>
<td>c1355/340</td>
<td>26.3</td>
<td>13.9/13.8</td>
<td>7.0/6.5</td>
<td>32.6/32.6</td>
<td>32.6/32.6</td>
<td>13.2/12.8</td>
<td>12.9/12.9</td>
</tr>
<tr>
<td>c3540/1669</td>
<td>31.2</td>
<td>14.9/14.6</td>
<td>10.4/9.8</td>
<td>23.9/24.1</td>
<td>23.9/24.1</td>
<td>10.0/9.7</td>
<td>10.0/9.7</td>
</tr>
<tr>
<td>c5315/2307</td>
<td>34.6</td>
<td>16.4/16.0</td>
<td>13.7/13.4</td>
<td>32.6/32.6</td>
<td>32.6/32.6</td>
<td>9.0/9.7</td>
<td>9.0/9.7</td>
</tr>
<tr>
<td>c880/383</td>
<td>38.6</td>
<td>17.6/17.2</td>
<td>20.0/19.6</td>
<td>50.9/50.9</td>
<td>50.9/50.9</td>
<td>5.0/4.8</td>
<td>5.0/4.8</td>
</tr>
<tr>
<td>c499/202</td>
<td>42.5</td>
<td>19.1/18.7</td>
<td>26.5/26.1</td>
<td>70.5/70.5</td>
<td>70.5/70.5</td>
<td>10.0/9.7</td>
<td>10.0/9.7</td>
</tr>
<tr>
<td>c1355/340</td>
<td>46.1</td>
<td>21.6/21.1</td>
<td>34.0/33.6</td>
<td>90.5/90.5</td>
<td>90.5/90.5</td>
<td>8.0/7.9</td>
<td>8.0/7.9</td>
</tr>
<tr>
<td>c3540/1669</td>
<td>51.2</td>
<td>24.1/23.7</td>
<td>42.5/42.1</td>
<td>110.5/110.5</td>
<td>110.5/110.5</td>
<td>5.0/4.8</td>
<td>5.0/4.8</td>
</tr>
</tbody>
</table>

8. REFERENCES

Figure 6: Leakage vs. delay for c499 under 99.73% timing yield.

Figure 7: Leakage vs. timing yield for c1355.

Figure 8: Leakage distributions of c499 under 99.73% timing yield.

858