Post-Synthesis Leakage Power Minimization

Mohammad Rahman and Carl Sechen
Department of Electrical Engineering, University of Texas at Dallas, United States
rahman@utdallas.edu, carl.sechen@utdallas.edu

Abstract—We developed a new post-synthesis algorithm that minimizes leakage power while strictly preserving the delay constraint. A key aspect of the approach is a new threshold voltage \(V_T\) assignment algorithm that employs a cost function that is globally aware of the entire circuit. Thresholds are first raised as much as possible subject to the delay constraint. To further reduce leakage, the delay constraint is then iteratively increased by \(\Delta\) time units, each time enabling additional cells to have their threshold voltages increased. For each of the iterations, near-optimal cell size selection is applied so as to reacquire the original delay target. The leakage power iteratively reduces to a minimum, and then increases as substantial cell upsizing is required to re-establish the original delay target. We show results for benchmark and commercial circuits using a 40nm cell library in which four threshold voltage options are available. We show that the application of the new leakage power minimization algorithm appreciably reduces leakage power after multi-\(V_T\) synthesis by a leading commercial tool, achieving an average post-synthesis leakage reduction of 37% while also reducing total active area and maintaining the original delay target.

Keywords- algorithm, leakage power, multiple threshold voltage \(V_T\) optimization

I. INTRODUCTION

For modern processes leakage power is almost comparable to the dynamic power even when the device is in the active mode. Efforts to mitigate this problem have resulted in multi-threshold voltage \(V_T\) transistor implementations. The objective of this work is to provide an algorithm that reduces the leakage for a synthesized digital integrated circuit, indeed seeking to minimize leakage power, while reducing dynamic power and strictly maintaining the synthesized target delay.

Prior work has investigated leakage optimization using two \(V_T\)’s and sizing [1]-[9]. Most of the algorithms for this approach allow the cell sizes to remain in the continuous domain, while the \(V_T\) is pre-assigned to either the high or low value. For example, in [1], a level-based back-tracing local slack-based algorithm is utilized to assign high-\(V_T\) to non-critical cells. In [2][3], first the \(V_T\) is fixed at the higher value. Then cell sizing is performed to obtain specified area (dynamic power) and delay goals. Slack based heuristics are incorporated to select candidate cells on the critical path to be converted to the lower value of \(V_T\). An explicit enumeration and pruning based transistor sizing and \(V_T\) assignment technique is proposed in [4]. The performance of transistor-level \(V_T\) assignment (meaning each transistor in a cell can receive a separate \(V_T\) assignment) is improved in [5] via careful reduction of cell variants (combinations of different \(V_T\) transistors) during the enumeration process. In [6] \(V_T\) is allowed to be a variable in the optimization process. Then discrete \(V_T\) values are assigned by nearest rounding and discrete cell sizes are obtained by snapping to the closest available cell in a library with high drive strength granularity. In [7], a discrete dual-\(V_T\) solution is obtained by modeling each gate as a parallel combination of a high and low \(V_T\) cell. Then a general-purpose nonlinear optimizer was utilized to obtain a nonzero continuous size for only one of them. In [8], parallelism is utilized for discrete \(V_T\) assignment and continuous cell sizing. A dynamic programming based algorithm was presented in [9] for simultaneous gate sizing and threshold voltage assignment in the discrete domain. These previous research efforts mainly used analytical models for leakage with only two \(V_T\) options. However, from an industrial impact point of view, a \(V_T\) selection methodology based on existing standard sign-off data (i.e., .lib files for both delay and leakage) is required, in addition to the need to handle an arbitrary number of \(V_T\) choices (some due to elongated channel lengths). In contrast to the above prior work, we compare our results with those of the latest version of a leading commercial synthesis tool using multi-\(V_T\) synthesis mode and for recently designed industrial circuits containing 300k cells or more. We use industry standard tools (e.g., PrimeTime) to measure delays and leakages.

More recently, a conjugate-gradient based nonlinear programming method for \(V_T\) assignment was reported [13]. While this tool was reportedly integrated into a commercial tool, it is not available for comparison. This method suffers from the drawback that the delay generally pushes out from the initial (synthesis) delay target. Also, it relies on path-based timing analysis in which the number of paths considered is very limited. Hence it is not guaranteed to find the actual worst path, and usually will not according to our experience.

In our new \(V_T\) selection algorithm, threshold voltages are raised as much as possible while strictly maintaining the delay goal. A key aspect of the approach is a cost function that is globally aware of the entire circuit. The algorithm iteratively globally ranks all of the cells based on a cost function which is the total slack elimination (in the whole circuit) divided by the leakage reduction (for that cell). The lowest cost cell is swapped to the next highest available \(V_T\), as long as the delay is not increased. The procedure iterates until no cell can feasibly be swapped to a higher \(V_T\). To the best of our knowledge, this is the first approach that strictly maintains the delay target with the aid of an efficient incremental static timing technique. Thus, there is no degradation in either worst negative slack (WNS) or total negative slack (TNS).

The leakage power we report and minimize is that specified in the industry standard .lib cell characterization data provided by a cell library provider. Likewise, when computing delays and slack, our static timing analyzer (STA) uses the .lib cell characterization data and the same delay propagation scheme employed by the leading static timing sign-off tools (e.g., PrimeTime). Thus, the worst-case path delay we report is the
same as that reported by PrimeTime. All paths from primary input to primary output, register-to-register, primary input to register and register to primary output are included in static timing analysis. Extracted wire resistances and capacitances are included in the delay computations.

II. \( V_T \) SELECTION ALGORITHM

In order to control industrial library development and maintenance costs, the \( V_T \)'s are the same for all of the transistors within a given cell, and thus that is the version of the problem we address here. Changing a cell’s \( V_T \) preserves its area or footprint, which enables \( V_T \) assignment post layout. The pre-characterized standard cell library database file (\( .lib \)) includes the average leakage power for each cell in the library.

The number of \( V_T \) options is usually around 4, and thus we have found that the optimization of \( V_T \) in the continuous domain and then later mapping it to a discrete value is not effective. On the other hand, finding the exact optimum solution for discrete \( V_T \) assignment will require an integer program whose run time is prohibitive even for modest sized designs. Hence, we developed a heuristic algorithm for \( V_T \) selection.

Our \( V_T \) selection algorithm is applied post synthesis. Thus it is often the case that some degree of \( V_T \) assignment was already done since multi-\( V_T \) synthesis usually results in lower leakage designs. Furthermore, some cell size optimization was also performed in order to meet the specified delay target. The problem we address is therefore: assign \( V_T \)'s such that the leakage power is minimized while strictly maintaining the delay goal. In other words, we seek to swap cells to a higher \( V_T \) to absorb the available slack in the design without sacrificing delay. For example, if raising the \( V_T \) of a cell causes the worst-case delay of the circuit to exceed the delay target, then that move is rejected.

When a cell is swapped to a higher \( V_T \), its leakage power is reduced; however, this not only impacts its slack, but also the slacks of cells in its fan-out and fan-in cones. In the circuit fragment shown in Fig. 2.1, swapping cell U19 to a higher \( V_T \) will increase its stage delay and its output slew rates. The arrival time and slew rate for input pins of U19 may also change due to changes in input pin capacitances. This will impact the output arrival time and slew rate of the effective fan-out cone of U19 (which is defined as the union of the fanout cones of all cells driving U19) and the required time of the cells in the fan-in cone of U19. In Figure 2.1, the effective fanout cone of U19 consists of cells U14, U18, U16, and U15 (green). The fan-in cone of U19 consists of U10 and U17 (red). Thus, the slack of all these cells will be reduced due to the \( V_T \) swap of only U19. Swapping a register to a higher \( V_T \) will increase its clock-to-Q delay and the input pin setup time. The changes in clock-to-Q delay will impact the arrival time of cells only in its fan-out cone. On the other hand, the change in setup time will change the required time of the cells only in the register’s fan-in cone.

We have found that the order in which to examine cells for \( V_T \) increment is the most critical part of the overall strategy. The ideal cell to select for \( V_T \) increment is one that reduces the leakage the most while also consuming the least amount of the total slack available in a circuit. This will enable more cells to subsequently have their \( V_T \) increased.

We thus developed a global cost function (1) inspired by the concept of potential slack [12]. We used the ratio of total slack reduction from the design divided by the total leakage power reduction as our global cost function. Cells are investigated in the order of least cost.

\[
\text{Cost} = \frac{\text{Total slack reduction from design}}{\text{Total leakage reduction}}
\] (1)

We used an efficient incremental timing technique, extended from [14], to obtain the total slack change of a design by updating arrival time and slew-rate accurately. First, we calculate the arrival time (forward sweep) and required time (backward sweep) for all cells. Then we trial move each cell, one at a time, to the next higher \( V_T \); all remaining cells are at their initial \( V_T \). An incremental timing update (both delay and slew rate), presented in Figure 2.2, is performed to calculate the total slack change due to the increment of a \( V_T \). We restore the delay changes before computing the cost of the next cell. Should the \( V_T \) change of a cell cause a delay violation, its cost becomes effectively infinity. Cells with finite cost comprise the so-called swappable list. The use of priority queues simplifies the incremental timing algorithm relative to [14].

After obtaining the ranking using (1), we examine cells sequentially, in the order of least cost. We accept the \( V_T \) change of a cell only if there is no delay violation. Note that the cells in the swappable list cannot be wholesale swapped to the next higher \( V_T \), as each \( V_T \) change leads to some loss of slack causing the possibility of a delay push-out in a subsequent “swap”.

After investigating the possibility of incrementing the \( V_T \) of every cell in the so-called swappable list, the global ranking of the cells using (1) is re-executed, creating an entirely new swappable list. Note that a cell incremented to a higher \( V_T \) can appear on a subsequent swappable list if a yet higher \( V_T \) is possible for that cell (without a delay violation).

We continue this loop until we cannot get a single cell in the swappable list. Rather than swapping a cell directly to the extremely highest \( V_T \) (if we have more than two \( V_T \) options), we increase the \( V_T \) one step at a time. We have observed that this incremental perturbation technique helps to more effectively absorb the slack available in the design. Our \( V_T \) selection algorithm is illustrated in Figure 2.3.

III. LEAKAGE POWER MINIMIZATION

Our main objective is to seek the minimum possible leakage power for a synthesized design. Simply increasing \( V_T \)'s as
much as possible may not yield the lowest leakage power, since it may be possible to sacrifice some dynamic power to obtain yet lower leakage power. While this won’t always be a desired trade-off, it certainly will be for certain battery-powered portable electronics, and especially those that operate at low frequency and for which battery charge is more constrained.

**Algorithm 1: Increment_Timing (Cₜ)**
1. initiate a Forward_Priority_Queue (FPQ) and a Backward_Priority_Queue (BPQ)
2. add all the drivers of cell Cᵢ to FPQ
3. while (FPQ_Size > 0) do
   4. pop a cell c from FPQ and update delay
   5. if (delay violation) do
      // Vₜ change is not allowed //
      // Restore original delays & edge rates //
   6. move c and remaining cells in FPQ to BPQ
   7. else if (output arrival time or slew rate changed) do
      add fan-out cells of c to FPQ
   8. else
      add c to BPQ // propagation can stop here //
   9. while (BPQ_Size > 0) do
      pop a cell c from the BPQ queue
   10. if (delay violation at step 5) do // restore mode //
   11. restore delay of c
   12. add fan-in cells of c to BPQ (until drivers of cell Cᵢ) //only
   13. if the cell is touched during forward traversal //
   14. else
      update required time and change in slack
   15. if (slack changed) do
      add fan-ins of c to BPQ
   16. return cell to its previous Vₜ

**Algorithm 2: Vₜ Selection**
1. Generate_Global_Ranking()
2. if (swappable_cell_count > 0) do
   3. for each cell Cᵢ in the swappable list do
      4. swap cell Cᵢ to the next higher Vₜ
   5. Increment_Timing(Cᵢ)
   6. if (delay violation) do
      7. return cell to its previous Vₜ

**Generate_Global_Ranking()**
1. for each cell in the design do
   2. swap the cell to the next higher Vₜ
   3. Increment_Timing(Cᵢ)
   4. if (no delay violation) do
      5. cost = (Total slack reduction)/(Total leak. power red.)
      6. swappable_cell_count++
   7. else
      8. cost = ∞
   9. return cell to its previous Vₜ and restore changes
10. Sort the cells based on cost

The Vₜ selection algorithm is initially applied using the same delay target that was employed by the synthesis tool. However, more cells can be moved to higher Vₜ's if the delay target were to be increased. In an attempt to find a lower leakage solution, we iteratively increment the original delay target, each time by an additional Δ time units (Δ being approximately the inverter FO4 delay), and re-run Vₜ assignment. A typical result is shown in Figure 3.1 for a 160k-cell circuit, where leakage power is shown as the delay target is iteratively extended. Each point in the plot represents a different Vₜ distribution of the cells in the design, with no change in cell sizes.

However, the circuit must adhere to the delay target used in synthesis. Thus each of the additional netlists must be subjected to cell size optimization in order to recapture the original synthesis delay target. This is accomplished using a near-optimal cell size selection technique [10], which in turn was extended from [11]. Eventually, as the original delay target is incremented substantially prior to Vₜ assignment, it may no longer be possible to recover the original delay target by cell size optimization due to so many cells having substantially increased threshold voltages.

The results obtained after cell sizing for each individual point comprise the leakage power versus active area (or cell dynamic power) profile for the specified synthesis delay target for the design. After a certain point the additional Vₜ increments, enabled by pushing out the original delay target, no longer result in reduced leakage power due to the additional leakage caused by the increase in active area needed to re-establish the original delay target. The minimum in the leakage versus active area plot constitutes the minimum leakage power design point. A typical example is in Figure 4.2. As expected, the leakage power iteratively reduces to a minimum value, and then starts to increase as substantial cell up sizing is required to re-establish the original delay target. Our leakage power minimization algorithm is shown in Figure 3.2.
IV. RESULTS

The algorithms were implemented in C and tested on a 2.9 GHz Intel Xeon CPU running Linux. We used a state-of-the-art 40nm industrial standard cell library. We accounted for multiple PVT corners, as guided by the supplier of the industrial circuits, using weak, 0.9V, 125°C for delay, and strong, 1.05V, 105°C for leakage power.

The cell library includes four different threshold voltage options: (i) standard V_T (SVT), (ii) extended standard V_T (XSVT, with an extended channel length), (iii) high V_T (HVT), and (iv) extended high V_T (XHVT). We extracted the average leakage power of the cells from the pre-characterized library data base file (.lib). The average leakage reduction when swapping an INV cell to a higher V_T is illustrated in Figure 4.1. There is a 62% reduction in the leakage between an SVT cell and an XSVT cell. Using an HVT cell resulted in a 72% leakage power reduction. Finally, the XHVT cell reduces the leakage power by 86% over the SVT cell.

To evaluate our optimization techniques, we performed physical driven synthesis for a set of ITC99 benchmarks and current industrial designs using the latest version of the leading commercial synthesis tool, using its multiple-V_T (MVT) optimization mode. We used 0.75 core utilization and 6 layers of metal for place and route. The results obtained from these synthesized designs were used as our baselines (Table I).

![Image: Inverter average leakage power for different V_T's](image)

Figure 4.1: Inverter average leakage power for different V_T's

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Period (ps)</th>
<th>Cell Count</th>
<th>Active Area (mm)</th>
<th>Leak. Power (mw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B17</td>
<td>700</td>
<td>23958</td>
<td>45.3</td>
<td>9.5</td>
</tr>
<tr>
<td>B17_1</td>
<td>700</td>
<td>23850</td>
<td>41.7</td>
<td>8.7</td>
</tr>
<tr>
<td>B18</td>
<td>1000</td>
<td>59607</td>
<td>107.4</td>
<td>23.1</td>
</tr>
<tr>
<td>B18_1</td>
<td>910</td>
<td>58311</td>
<td>102.2</td>
<td>21.5</td>
</tr>
<tr>
<td>B19</td>
<td>1200</td>
<td>120251</td>
<td>178.4</td>
<td>34.4</td>
</tr>
<tr>
<td>B19_1</td>
<td>1200</td>
<td>114820</td>
<td>174.8</td>
<td>34.2</td>
</tr>
<tr>
<td>IND1</td>
<td>940</td>
<td>141133</td>
<td>170.2</td>
<td>39.1</td>
</tr>
<tr>
<td>IND2</td>
<td>740</td>
<td>159820</td>
<td>199.8</td>
<td>45.9</td>
</tr>
<tr>
<td>IND3</td>
<td>1780</td>
<td>294054</td>
<td>426.6</td>
<td>71.1</td>
</tr>
<tr>
<td>IND4</td>
<td>1580</td>
<td>275016</td>
<td>446.4</td>
<td>80.5</td>
</tr>
</tbody>
</table>

For the first set of experiments, we evaluated the benefits of applying only our V_T assignment algorithm, using the delay target from synthesis for the set of circuits. In this case, we started with the initial V_T assignment obtained from physical synthesis, although we obtained virtually the same results if we re-initialized all V_T’s to the lowest value (SVT) prior to executing our V_T optimization algorithm. That is, we optimized the V_T assignment of the synthesized designs (i.e., did not change the cell sizes) for the given delay targets.

To analyze the quality of our V_T assignment strategy (Opt_V_T) we compared our results with two different methods over a set of benchmarks. In method 1, termed Level, we implemented the level-based back-tracing algorithm of [1] to assign higher V_T to noncritical cells, where we extended it to handle multiple-V_T’s along with accurate delay and slew-rate estimation and propagation. In this approach all of the cells in a particular logic level are investigated for V_T swap, and the logic levels are visited sequentially starting from the primary output level. We also used a random exhaustive search technique in Method 2 (termed Rand). In an iterative fashion, we select a gate randomly, swapped it to next higher V_T and retain the swap only if there was no delay push-out. We continue this process until we obtain a large number of consecutive failures (e.g., 10n, where n is the total cell count in the design). We retain the best solution found among 20 different trials for Method 2. This method is very time consuming, and could only be executed on the smaller designs.

The comparison of our new method (Opt_V_T) with Level and Rand is presented in Table II. Note that we performed V_T swapping on designs that were sized using the leading commercial synthesis tool for a stringent delay constraint. Our new V_T optimization algorithm (Opt_V_T) obtained a 26% average reduction in leakage power for the designs, whereas Level and Rand yielded only 17% and 19% average leakage power reductions, respectively.

We next applied step 7 of our leakage minimization algorithm in Figure 3.2, namely cell_sizing, for the original delay target (k = 0). Since cell size optimization reduces active area while preserving the synthesis delay target, additional leakage savings result compared to V_T optimization alone. Leakage and active area reductions achieved after cell size optimization for the set of designs is shown in Table III. We obtained an average of 6% additional leakage reduction, for an overall average leakage reduction of 32% compared to the leading commercial synthesis tool. This was achieved while also reducing active area (corresponding to dynamic power) by an average of 15% and strictly maintaining the delay target. The percentage reduction in both leakage and active area shown in Table III are computed with respect to the corresponding baseline.

We next executed steps 3 through 7 (for k larger than 0) of the leakage power minimization algorithm in Figure 3.2. In other words, we iteratively increment the original delay target, each time by an additional Δ time units, and re-run V_T assignment. In order to pull the worst-case delay back to the original delay target, we then re-run cell size optimization (step 7 in Figure 3.2). The end result is the generation of leakage power versus active area plot for each design. Leakage power-versus-active area is shown for an ITC99 benchmark and an industrial circuit in Figures 4.2 and 4.3, respectively. It clearly shows the trade-off between active area and leakage power, and demonstrates the presence of a minimum leakage power point.

With respect to the minimum leakage power point, our algorithm achieved a 37% reduction in total leakage power for design B19 (Figure 4.2) and a 34% reduction in total leakage power for design IND4 (Figure 4.3) with respect to the corres-
ponding commercial MVT synthesis base line. At the minimum leakage solution, the total leakage power and active area obtained for the set of designs is presented in Table IV. The average reduction in leakage power was 37%, which is 5% more than the initial solution given in Table III, at the cost of 8% in active area. Nonetheless, the active area is still better than the commercial synthesis baseline by an average of 7%. As in the previous tables, all delays and leakage powers were as reported by Synopsys’ PrimeTime. (For reference, if each of the benchmarks had the $V_T$’s of all of their cells set to the lowest $V_T$ (SVT) in Table I, the optimized leakage powers in Table IV are 70-80% smaller in each case.)

### Table II. Leakage Power Reduction After $V_T$ Assignments

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Leakage Power (mw)</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B17</td>
<td>7.9</td>
<td>7.6</td>
</tr>
<tr>
<td>B17_1</td>
<td>7.3</td>
<td>7.2</td>
</tr>
<tr>
<td>B18</td>
<td>18.6</td>
<td>18.3</td>
</tr>
<tr>
<td>B18_1</td>
<td>17.6</td>
<td>17.4</td>
</tr>
<tr>
<td>B19</td>
<td>26.7</td>
<td>-</td>
</tr>
<tr>
<td>B19_1</td>
<td>26.4</td>
<td>-</td>
</tr>
<tr>
<td>IND1</td>
<td>34.8</td>
<td>-</td>
</tr>
<tr>
<td>IND2</td>
<td>40.0</td>
<td>-</td>
</tr>
<tr>
<td>IND3</td>
<td>61.1</td>
<td>-</td>
</tr>
<tr>
<td>IND4</td>
<td>68.9</td>
<td>-</td>
</tr>
<tr>
<td>Average</td>
<td>16.8</td>
<td>19.3</td>
</tr>
</tbody>
</table>

### Table III. Leakage and Active Area Reduction After Opt. $V_T$ and Cell Sizing

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Leakage Power (mw)</th>
<th>Reduction (%)</th>
<th>Active Area (mm)</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B17</td>
<td>6.2</td>
<td>35.3</td>
<td>38.9</td>
<td>16.1</td>
</tr>
<tr>
<td>B17_1</td>
<td>6.0</td>
<td>30.5</td>
<td>35.9</td>
<td>14.1</td>
</tr>
<tr>
<td>B18</td>
<td>14.8</td>
<td>35.9</td>
<td>94.1</td>
<td>12.4</td>
</tr>
<tr>
<td>B18_1</td>
<td>14.1</td>
<td>34.4</td>
<td>89.1</td>
<td>12.8</td>
</tr>
<tr>
<td>B19</td>
<td>23.5</td>
<td>31.7</td>
<td>152.2</td>
<td>14.7</td>
</tr>
<tr>
<td>B19_1</td>
<td>23.2</td>
<td>32.1</td>
<td>147.9</td>
<td>15.4</td>
</tr>
<tr>
<td>IND1</td>
<td>28.2</td>
<td>27.9</td>
<td>144.1</td>
<td>15.3</td>
</tr>
<tr>
<td>IND2</td>
<td>30.4</td>
<td>33.8</td>
<td>166.3</td>
<td>16.8</td>
</tr>
<tr>
<td>IND3</td>
<td>50.9</td>
<td>28.4</td>
<td>349.5</td>
<td>18.1</td>
</tr>
<tr>
<td>IND4</td>
<td>56.5</td>
<td>29.8</td>
<td>367.6</td>
<td>17.7</td>
</tr>
<tr>
<td>Average</td>
<td>32.0</td>
<td>15.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The $V_T$ distributions for the combinational logic cells and sequential cells (registers) for the commercial synthesis tool solution point and at the minimum leakage solution point for two industrial designs (IND2 and IND4) are shown in Figures 4.4 and 4.5, respectively. The majority of the cells are assigned to XSVT and XHVT (the so-called extended channel length $V_T$’s). It’s apparent that our leakage power minimization algorithm (Figure 3.2) results in appreciably more cells being assigned to XHVT, for example, 48% more combinational cells and 20% more sequential cells for IND2 and 21% more combinational cells and 38% more sequential cells for IND4.

### Table IV. Minimization of Leakage Power

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Leakage Power (mw)</th>
<th>Reduction (%)</th>
<th>Active Area (mm)</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B17</td>
<td>5.7</td>
<td>40.0</td>
<td>46.1</td>
<td>-1.8</td>
</tr>
<tr>
<td>B18</td>
<td>13.9</td>
<td>39.8</td>
<td>99.3</td>
<td>7.5</td>
</tr>
<tr>
<td>B19</td>
<td>21.6</td>
<td>37.2</td>
<td>164.1</td>
<td>8.0</td>
</tr>
<tr>
<td>IND1</td>
<td>26.1</td>
<td>33.2</td>
<td>156.6</td>
<td>8.0</td>
</tr>
<tr>
<td>IND2</td>
<td>28.4</td>
<td>38.1</td>
<td>180.7</td>
<td>9.6</td>
</tr>
<tr>
<td>IND3</td>
<td>46.3</td>
<td>34.9</td>
<td>386.8</td>
<td>9.3</td>
</tr>
<tr>
<td>IND4</td>
<td>53.1</td>
<td>34.0</td>
<td>417.8</td>
<td>6.4</td>
</tr>
<tr>
<td>Average</td>
<td>36.8</td>
<td>15.1</td>
<td></td>
<td>6.7</td>
</tr>
</tbody>
</table>

The $V_T$ distributions for the combinational logic cells and sequential cells (registers) for the commercial synthesis tool solution point and at the minimum leakage solution point for two industrial designs (IND2 and IND4) are shown in Figures 4.4 and 4.5, respectively. The majority of the cells are assigned to XSVT and XHVT (the so-called extended channel length $V_T$’s). It’s apparent that our leakage power minimization algorithm (Figure 3.2) results in appreciably more cells being assigned to XHVT, for example, 48% more combinational cells and 20% more sequential cells for IND2 and 21% more combinational cells and 38% more sequential cells for IND4.
Table V shows average run times for the $V_T$ selection algorithms for a set of designs. In Figure 4.6, run time as a function of design size (number of cells, $n$) is shown. Linear regression shows that the empirical run-time complexity of the $V_T$ selection (Figure 2.3) is $O(n^{1.5})$.

### Table V. AVG. CPU TIME FOR OPT $V_T$

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Cell Count ($n$)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6288</td>
<td>3210</td>
<td>6</td>
</tr>
<tr>
<td>B20</td>
<td>13700</td>
<td>69</td>
</tr>
<tr>
<td>B22</td>
<td>20517</td>
<td>87</td>
</tr>
<tr>
<td>B17</td>
<td>23958</td>
<td>105</td>
</tr>
<tr>
<td>B18</td>
<td>59607</td>
<td>221</td>
</tr>
<tr>
<td>B19</td>
<td>120251</td>
<td>481</td>
</tr>
<tr>
<td>IND1</td>
<td>141133</td>
<td>702</td>
</tr>
<tr>
<td>IND2</td>
<td>159820</td>
<td>751</td>
</tr>
<tr>
<td>IND3</td>
<td>273016</td>
<td>1053</td>
</tr>
<tr>
<td>IND4</td>
<td>294054</td>
<td>1351</td>
</tr>
</tbody>
</table>

For the $V_T$ selection algorithm, step 1 in Figure 2.3 requires incremental timing $n$ times and one sorting once all the cost computations have been completed. If $k$ is the fixed upper bound on the maximum number of cells to be traversed for an incremental timing, the time complexity of step 1 is $O(kn + n \log n)$. Let $I$ be the fixed upper bound on the maximum number of $V_T$ swappables obtained per iteration and $m$ be the fixed upper bound on the maximum number of iterations required until no cell will be available in the swappable list. Then the time complexity of the $V_T$ selection algorithm is $O(m*(I^k + kn + n \log n))$, which is equivalent to $O(n \log n)$ given the fixed constants. It is not surprising we empirically observe $O(n^{1.5})$.

We also applied Opt$_{V_T}$ (Algorithm 2 in Fig. 2.3) recently to two additional current industrial designs in 28nm technology. IND5 (600 MHz) had 337k sizeable cells (excludes test and clock cells). Cell size optimization [10] reduced the total active area by 43% compared to the synthesized design, which only used SVT, for the same delay and the leakage was reduced from 164.9 mW down to 102.2 mW. Opt$_{V_T}$ then reduced the leakage down to 33.7 mW, which is 80% lower. The second (IND6, 900 MHz) had 395k sizeable cells and was produced using MVT synthesis. After cell size optimization reduced the active area by 31% for the same delay, the leakage was reduced from 311 mW down to 221 mW. Opt$_{V_T}$ then reduced the leakage down to 91.6mW (71% lower), also for the same delay.

### Table VI. ADDITIONAL INDUSTRIAL CIRCUITS

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Synthesis</th>
<th>Cells</th>
<th>Syn. Base Line</th>
<th>Cell Sizing + Opt$_{V_T}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IND5</td>
<td>Single $V_T$</td>
<td>336,864</td>
<td>164.9 mW</td>
<td>740 mm</td>
</tr>
<tr>
<td>IND6</td>
<td>Multi $V_T$</td>
<td>395,550</td>
<td>311 mW</td>
<td>753 mm</td>
</tr>
</tbody>
</table>

### V. CONCLUSION

We developed a new post-synthesis algorithm that minimizes leakage power while preserving the delay constraint. A key aspect of the approach is a new $V_T$ assignment algorithm that employs a cost function that is globally aware of the entire circuit. The thresholds of the cells are first raised as much as possible subject to the delay constraint. Then the delay constraint is iteratively pushed out by $\Delta$ time units, each time enabling additional cells to have their threshold voltages increased. In order to preserve the delay constraint, near-optimal cell size selection is employed for each of the $V_T$ assignments. As expected, the leakage power iteratively reduces to a minimum value, and then starts to increase as substantial cell upsizing is required to re-establish the original delay target.

We showed results for a number of benchmark and commercial circuits using a contemporary 40nm cell library in which four threshold voltage options were available. Compared to the best possible multi-$V_T$ synthesis results by the leading commercial synthesis tool, we showed that the application of the new leakage power minimization algorithm reduces leakage power an average of 37% while also reducing total active area and maintaining the original synthesis delay target. All delays and leakage powers were as reported by Synopsys’ PrimeTime.

### REFERENCES


