Leakage power optimization for 28nm and beyond

Aishwary Dadheech, Engineer, Dhaval Parikh, Senior Engineer, Jinesh Patel, Engineer, eInfochips - April 07, 2014

As process nodes shrink towards nanotechnology, the supply voltage is scaled down to protect the device from excessive electric field across the gate oxide and the conducting channel. Another reason for supply voltage reduction is to save dynamic power dissipation. On the flip side, this voltage reduction slows down the CMOS transistor.

To overcome this performance loss, the transistors’ threshold voltage is decreased, which increases sub-threshold leakage current. At these nodes, leakage power can contribute more than 50% of the overall chip power – hence, high performance chips have excess power dissipation, even in standby mode. Increasing performance causes more leakage power dissipation.

To counter this challenge, the typical design flow uses positive slack margins of non-critical paths and adds cells with high Vt.

STA is pessimistic in physical design as it uses conservative timing calculations with Graph Based Analysis (GBA). Additional pessimism is introduced by flat OCV (On-Chip Variations) derating and heavy clock uncertainty. These are educated guesses derived from estimations of variations and tool correction errors respectively. But applying flat derates for all the standard cells globally also is not a realistic scenario because all cells cannot have the same (worst) variation.

The flow we’ve developed reduces pessimism and recommends three accurate and realistic timing approaches. We strive to increase the positive margin to maximize leakage power optimization – cells with positive slack get converted to HVT.

Methodology

1. Graph Based Analysis (GBA) vs. Path Based Analysis (PBA)

In STA, there is always a trade-off between runtime and accuracy. GBA improves runtime at the cost of accuracy as it uses the worst input transition to calculate the delay of a given standard cell. This results in pessimism at each stage of the timing path. Typically, clock trees do not get affected by GBA because they are buffers and/or Inverters (single input standard cells), while PBA calculates delay from the start point and traces the path to the end. Only the transition of the input pin along the path is considered (not the worst slew).
Figure (a) shows the calculation of a delay arc A-Z for a three input AND gate. The input slew of pin C is 50ps, and pin A, 20ps. GBA uses the worst slew (50ps) to get the delay for all three arcs, A-Z, B-Z, and C-Z, which is pessimistic for A-Z and B-Z arcs. Compared to this, PBA uses the actual input pin’s slew for each arc delay calculation.

Figure (b) shows pessimism reduction of PBA compared to GBA for a number of timing paths and path slacks.

PBA analysis takes high amount of runtime to recalculate timing, considering real transitions on the nodes in a specific path. Practically, PBA can not be applied on all paths of the design. Our flow applies PBA only on critical or violating paths.

This approach significantly benefits the accuracy while moderately improving the runtime. The flow swings negative slack to be positive, and positive slack to be more positive. It utilizes this positive slack for leakage power optimization by swapping more cells to HVT.
2. On-Chip Variation (OCV) vs. Advanced On-Chip Variation (AOCV)

On-chip variation (OCV) is the discrepancy in timing properties of cells or interconnects due to process, voltage, and temperature (PVT) variation on the chip. PVT cannot be exactly the same across the chip. For example, two inverters on the same chip can have different delays due to PVT variation. In order to consider this effect, OCV has been introduced to the STA. In traditional OCV, the applied derate factor is flat and global. This is a pessimistic approach because it is not possible that all cells will have the worst PVT variation. Actually, this variation depends on distance / location of cell and logic path depth. In advanced OCV (AOCV), the derate values are not flat (fixed), but are more representative of the real scenario.

![Diagram showing OCV vs AOCV derate](image)

**Figure (c)** OCV flat derate is pessimistic compared to AOCV derate in high logic depth path.

We have enabled AOCV in the signoff timing tool and provided an AOCV table that contains the derate values based on path depth vs. distance between standard cells. The derate value decreases as logic depth increases. We can increase the accuracy with design specific AOCV as well. In this case, OCV derates depend on slew and output capacitance of the cell and also the IR drop of the design. AOCV removes the pessimism of normal OCV and achieves higher positive margins.

3. Uncertainties

Our flow recommends use of realistic instead of pessimistic uncertainty. In general, we deploy worst case libraries for STA, when the chip is likely to be deployed in typical conditions. This brings in a certain degree of pessimism to our designs. At the signoff stage, we can use realistic certainties to increase positive slack. This positive slack can be used to increase HVT cell count in the design to minimize leakage power.
Figure (d) (right) shows our optimization algorithm in a flowchart. This flow starts with the final timing analysis. Before going to any pessimism reduction or optimization, we generate timing reports and cell statistics with different Vt classes for our records. The next is to reduce extra pessimism by clock uncertainty reduction and by AOCV derate factors based on logic depth and location. We generate the timing reports to see the difference in slack and path numbers and compare them to the initial reports.

Till this point we have followed GBA. The third step is to set the criteria and targets on slack, slew, and crosstalk delay for cell swapping (e.g., convert cell to HVT only if it has slack >+50ps, slew
<150ps, and crosstalk delta <25ps). For the criteria, we decide the path to swap the standard cells with HVT cells. Within the paths, we also select the type of cells to convert to HVT.

We then identify all cells that meet the desired criteria and convert them to HVT versions of the standard cells. Now, we again generate the GBA timing report and Vt cell statistics to find the number of cells converted to HVT.

We now have more paths that violate the timing because of the heavy HVT conversion. It is now time to remove the pessimism of GBA without adding major runtime to the STA. We apply PBA only on the violating paths in the new GBA report. The runtime will be faster compared to the PBA on the full design.

Next, to get real and accurate timing, we generate timing reports with PBA. We will resolve remaining timing violations by swapping cells back to regular Vt in the violating paths.

For that, we find out one cell with the most delay / slew / delta in each violating path and convert that cell back to regular Vt. Then we check PBA timing again.

We iterate on this till the violating paths become zero, then generate final timing reports and the cell statistics with Vt class.

**Actual results**

We deployed this methodology for a 28nm ASIC physical design. The ASIC is now on the market, and being deployed in multiple products.

After a placement and route cycle, we implemented this leakage power reduction flow on a real design at signoff stage, where the different blocks have complexity ranging from 100K to 500K instance count. Table (a) shows three block statistics, where the impact of uncertainty reduction on worst slack, number of failing paths, and HVT cell percentage is evident.

We reduced uncertainty by 50ps, and achieved an average 8-9% increment in HVT cell count:

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Uncertainty 155ps</th>
<th>Uncertainty 105ps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Worst Slack</td>
<td># of Paths</td>
</tr>
<tr>
<td>TC1</td>
<td>-63ps</td>
<td>376</td>
</tr>
<tr>
<td>TC2</td>
<td>-54ps</td>
<td>1196</td>
</tr>
<tr>
<td>TC3</td>
<td>-53ps</td>
<td>246</td>
</tr>
</tbody>
</table>

Table (a) Uncertainty reduction vs. timing and HVT %

Table (b) shows a flow statistics comparison between pre-optimization data and post-optimization data. It shows instance counts, pre-optimization HVT cell %, post-optimization HVT cell %, pre-optimization leakage power, and post-optimization leakage power.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>157K</td>
<td>43.69</td>
<td>65.83</td>
<td>0.78</td>
<td>0.54</td>
</tr>
</tbody>
</table>
Table (b) Initial/Pre-Opt statistics for 8 blocks / Test cases

The average result at a full-chip level is shown in table (c). We achieved 16.31W leakage power reduction, i.e., 30% improvement with 43% increase in HVT cells.

<table>
<thead>
<tr>
<th>Leakage Power</th>
<th>Initial</th>
<th>Post-optimization</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>55.47W</td>
<td>39.16W</td>
<td></td>
<td>16.31W (30%)</td>
</tr>
</tbody>
</table>

Table (c) Average leakage results for the full chip

Conclusion

We used realistic yet accurate approaches to analyze timings. The timing methods like PBA, AOCV, and Realistic Uncertainty, removed unnecessary pessimism from the timing analysis and provided accurate timing numbers with increased positive slack. Our optimization flow used this positive slack to optimize leakage power by HVT cell conversion.

We have deployed these techniques on a 28nm ASIC that was taped-out in 2013. The results prove the strength and capability of our leakage optimization flow.

The authors are engineers at eInfochips.