Abstract—The complexity of timing optimization has been increasing rapidly in proportion to the shrinking CMOS device size, due to the increased number of channel-connected transistors in a path, and the rising magnitude of process variations. These significant challenges can be addressed through the implementation of designs with an optimal balance between static and dynamic circuits. This paper presents a process variation-aware Path Oriented IN Time (POINT) optimization flow for mixed-static-dynamic CMOS logic designs, where a design is partitioned into static and dynamic circuits based on timing critical paths. Implemented on a 64-b adder and ISCAS benchmark circuits, the POINT optimization flow demonstrated an average improvement in delay by 44% and average improvement in delay uncertainty from process variations by 37% in comparison with a state-of-the-art commercial optimization tool.

I. INTRODUCTION

The advancement in CMOS technology towards nanometer process has created an avenue for several new initiatives, while at the same time has also presented several challenges such as reliability, leakage current and increased delay uncertainty as expressed in (1) where $T_{\text{max}}$ and $T_{\text{min}}$ are the maximum and minimum delays of a critical path due to process variations. At 180 nm CMOS technology, these process variations have caused about 30% variation in chip frequency, along with 20X variation in chip leakage [9]. This increasing magnitude of process variations is predicted to cause detrimental effects as CMOS technology is advancing to 45 nm, highlighting the imminent requirement for advanced timing optimization flows to address these challenges.

$$\text{Uncertainty} = T_{\text{max}} - T_{\text{min}} \quad (1)$$

Several literatures exist on timing optimization [1-4], but most of them focus on static CMOS circuits and technologies using dual threshold voltages. TILOS [2] presented an algorithm used for iteratively sizing transistors by a factor in the critical path. This algorithm does not guarantee a convergence of timing optimization and is not a deterministic approach. MINFLOTRANSIT [3] is an algorithm proposed for transistor sizing based on iterative relaxation, but requires generation of directed acyclic graphs iteratively for timing optimization.

Together with timing optimization, several methods have been proposed to reduce the effect of process variations on chip performance [5-9]. These methods deal with statistical variations and are not optimal for designs with large magnitude of parameter variations. The Adaptive Body Biasing (ABB) technique presented in [7, 9] is implemented on post-silicon designs where each die receives a unique bias voltage, thus reducing delay uncertainty. However, this method is not feasible for addressing intra-die variations as each block in the design requires a unique bias voltage. Another limitation using this method is the increased leakage power due to reduced threshold voltage. Although all these methods limit the effect from process variations, none of them encompass timing optimization in their flow.

One of the circuit styles used predominantly for timing optimization is the domino dynamic logic. However, its usage is limited by challenges from high power consumption, and complexity in transistor sizing due to charge sharing, noise-immunity, and sensitivity to process variations. Both the static and dynamic circuits have their prospective advantages. Using both circuit styles effectively can result in a better performance than static and dynamic CMOS alone. This paper presents a Path Oriented IN Time (POINT) optimization flow for a Mixed-Static-Dynamic (MSD) CMOS logic for optimizing both timing and uncertainty from process variations. The proposed POINT optimization flow is integrated into existing state-of-the-art commercial CMOS design optimization flows.

II. TRANSISTOR SIZING ALGORITHM

The delay of dynamic CMOS circuits is highly dependent on the number and size of transistors in the critical path. Increasing size of transistors increases the discharging current, thus reducing the path delay. However, increasing transistor sizes will increase the load capacitance of channel-connected transistors on other paths and substantially increase their delays. The delay of a dynamic CMOS circuit can be reduced through an optimum balance of load capacitance from multiple paths in the design.

Research has demonstrated that process variations cause a significant variation in frequency. Figure 1 shows a 2-b Weighted Binary-to-Thermometric Converter (WBTC) with two timing paths: path-A and path-B highlighted, with their delay distribution from process variation in Fig. 2. It is evident from Fig. 2 that the choice of transistor sizes can vary the delay uncertainty to a large extent. A process variation-aware transistor sizing algorithm for Load Balance of Multiple Paths (LBMP) proposed in our previous work [10] can address both the challenges of transistor sizing for delay minimization and reducing the uncertainty from process variations together at the same time.
When process variations are considered, the conventional method computes worst delay based on the mean from the delay distribution, while accounting for only intra-die variations. As inter-die variations are equally important, the proposed process variation-aware LBMP transistor sizing algorithm computes delay from the sum of mean and standard deviation $\sigma$, accounting for both intra-die and inter-die variations as shown in (3).

As discharge time of transistors near ground is higher compared to the transistors near the output, the LBMP algorithm assigns high priority (weight) to transistors near ground. Also, as increasing the size of transistor that appears in the most number of paths reduces the overall delay, the LBMP algorithm computes the number of paths a transistor is present in (repeats). Once the repeat and the weight profiles of all transistors are computed, process variations are induced to obtain delay profiles of all the paths. The transistors in the top 20% critical paths are grouped to set-x, and their sizes are increased as in (2).

As delay of the critical path is also dependent on the loading from channel-connections, the LBMP algorithm minimizes this load by reducing sizes of transistors in the neighboring paths. The 1st order connection transistors for set-x are identified and grouped to set-y. Then, transistors in set-y that are not in the critical path are grouped to set-z. For each transistor in set-z, it is checked if the transistor is present in critical path of previous iteration. If so, its size is decreased and calculated by (3-4). If not, its size is decreased and calculated by (5). Once new transistor sizes are determined, process variations are induced to identify the new critical paths. If the new worst-case path delay is higher than in the previous iteration, sizes of transistors in set-z of the new worst-case path are changed to the average of its old and new sizes.

\[
\text{New\_Size} = \frac{\text{Old\_Size} + \text{Temp\_New}}{2} \quad (4)
\]
\[
\text{New\_Size} = \text{Old\_Size} \left(1 - \frac{\text{Repeats}}{1 + \text{Repeats}}\right) \times \text{Weight} \quad (5)
\]

Figure 1 shows a 2-b WBTC used in parallel adders. At the onset of the implementation of process variation-aware LBMP algorithm, the worst-case delay of 2-b WBTC was 355 ps. After implementing the LBMP algorithm, the worst-case delay has reduced to 157 ps, accounting for a 55.77% delay improvement. In addition, the delay uncertainty from process variations also reduced from 158 ps to 62 ps, accounting for a 60.75% improvement.

The efficiency of the LBMP transistor sizing algorithm is further validated through implementation on several benchmark circuits in IBM 130nm CMOS process. The design profiles along with their respective performance improvements are shown in Table 1.
Table 1: Timing Optimization results from LBMP Algorithm

<table>
<thead>
<tr>
<th>Design</th>
<th># Inputs</th>
<th># Outputs</th>
<th># Paths</th>
<th># Transistors</th>
<th>Initial Delay (ps)</th>
<th>Final Delay (ps)</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCT-2</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>226</td>
<td>109</td>
<td>52</td>
</tr>
<tr>
<td>2-b WBTC</td>
<td>4</td>
<td>6</td>
<td>34</td>
<td>28</td>
<td>355</td>
<td>157</td>
<td>55</td>
</tr>
<tr>
<td>4-b UWBTC</td>
<td>4</td>
<td>15</td>
<td>83</td>
<td>83</td>
<td>152</td>
<td>103</td>
<td>33</td>
</tr>
<tr>
<td>74181 – CLA</td>
<td>10</td>
<td>6</td>
<td>18</td>
<td>24</td>
<td>209</td>
<td>103</td>
<td>51</td>
</tr>
<tr>
<td>74181 – E mod</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>225</td>
<td>110</td>
<td>51</td>
</tr>
<tr>
<td>C2670 -CLA</td>
<td>24</td>
<td>1</td>
<td>15</td>
<td>39</td>
<td>218</td>
<td>108</td>
<td>51</td>
</tr>
<tr>
<td>C3540- CC5</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>484</td>
<td>208</td>
<td>57</td>
</tr>
<tr>
<td>C3540-CC8</td>
<td>7</td>
<td>3</td>
<td>35</td>
<td>17</td>
<td>427</td>
<td>216</td>
<td>50</td>
</tr>
<tr>
<td>C3540-CC9</td>
<td>8</td>
<td>3</td>
<td>22</td>
<td>47</td>
<td>283</td>
<td>171</td>
<td>39</td>
</tr>
<tr>
<td>C3540-UM12-7</td>
<td>9</td>
<td>1</td>
<td>24</td>
<td>50</td>
<td>485</td>
<td>178</td>
<td>63</td>
</tr>
</tbody>
</table>

III. POINT OPTIMIZATION FLOW

At the architecture level, one of the common limitations in most of current design optimization flows is their inability to account for process variations in timing analysis and optimization. Process variations are considered only at placement and route. After placement and route, if the design fails to meet the timing constraints, the entire flow is reiterated. Also, this process may result in design failing to meet timing, and may end up in timing failure and might miss the time-to-market window. The proposed process variation-aware Path Oriented IN Time (POINT) optimization flow as shown in Fig. 4 answers these challenges of accounting for process variations during timing optimization.

Research [1] has shown that Mixed-Static-Dynamic (MSD) logic results in better timing over conventional static CMOS circuits alone. The proposed process variation-aware POINT optimization relies on this principle, and performs timing optimization through effective partition of the design between static and dynamic logic.

Initially, a high-level description of a design is input to a synthesis and optimization. Following synthesis and optimization, Static Timing Analysis (STA) is performed on the static CMOS circuits to identify the timing critical modules. A strategy similar to LBMP algorithm is followed here to find the timing critical modules. These modules are identified based on the number and significance of critical delay paths flowing through them. Once these timing critical modules are identified, dynamic CMOS circuits for the same are designed, and timing optimization is performed using the process variation-aware LBMP algorithm.

The next step in the algorithm is replacement of static CMOS timing critical modules with the performance optimized dynamic CMOS circuits in the critical paths. With the updated MSD circuit design, the next step in the POINT optimization flow is clock tree design and timing verification. After the design is checked for clock signal timing constraints, STA is further performed to verify timing convergence. If timing is not met, new timing critical modules are identified and the MSD circuit is further optimized using the process variation-aware LBMP algorithm. Following the timing convergence, the MSD circuit design is exported for placement and route. Overall, the POINT optimization flow is a deterministic approach always moving towards an optimum solution.

Figure 4: POINT Optimization Flow
The process variation-aware POINT optimization flow was implemented in IBM 130nm CMOS process and verified on three ISCAS benchmarks and a 64-b parallel binary adder. Figure 5 shows the block diagram of the ISCAS benchmark C3540, an 8-b ALU with 1669 gates [11]. Through initial design synthesis and optimization using a commercial optimization tool, delays of top 5 critical paths in C3540 were found to be 3.59, 3.46, 3.16, 3.13, and 2.97 ns respectively (highlighted in Fig 5). The modules with highest delay in C3540 were found to be M5 with a delay of 1.63 ns, M4 with a delay of 0.78 ns, and M12 with a delay of 0.63 ns. The modules chosen for timing optimization in the POINT flow are M5 and M12. Although M4 had the second highest delay, it was not chosen for optimization as it does not repeat in as many critical paths as M5 and M12. This selection of modules for optimization is crucial to limit the increase in power consumption.

After the timing critical modules were updated with the performance optimized dynamic CMOS circuits, timing verification was performed on the C3540. One iteration of POINT optimization flow on C3540 has reduced the delays of top five critical paths to 2.42, 2.29, 2.28, 2.38, 2.22 ns respectively. Through implementation of the POINT optimization flow on C3540, the worst-case delay reduced from 3.59 ns to 2.42 ns, an improvement of 32%. In addition, the uncertainty from process variations also reduced from 3.17 ns to 1.89 ns, an improvement of 40% over the results obtained from the commercially state-of-the-art optimization tool. Similarly, the POINT optimization flow was implemented on several other ISCAS benchmark circuits and a 64-b parallel binary adder. Results obtained from the POINT Optimization flow are shown in Table 2.

![Figure 5: ISCAS benchmark - C3540 [11]](image)

### IV. CONCLUSION

In this paper, challenges in timing optimization of CMOS circuits with the shrinking device sizes have been presented. A solution addressing these challenges of timing optimization and increased delay uncertainty from process variations has been presented through the process variation-aware Path Oriented IN Time (POINT) optimization flow for mixed-static-dynamic CMOS circuits.

Through implementation on three ISCAS benchmark circuits and a 64-b parallel binary adder, the process variation-aware POINT optimization flow has demonstrated a performance improvement in the delay and the delay uncertainty from process variations by 44% and 37% respectively.

#### Table 2: POINT Optimization Flow results

<table>
<thead>
<tr>
<th>Design</th>
<th># PI</th>
<th># PO</th>
<th># Gates</th>
<th>Delay Improvement</th>
<th>Uncertainty Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>74181</td>
<td>14</td>
<td>8</td>
<td>74</td>
<td>43%</td>
<td>14%</td>
</tr>
<tr>
<td>C2670</td>
<td>233</td>
<td>140</td>
<td>1193</td>
<td>39%</td>
<td>32%</td>
</tr>
<tr>
<td>C3540</td>
<td>50</td>
<td>22</td>
<td>1669</td>
<td>32%</td>
<td>40%</td>
</tr>
<tr>
<td>adder64</td>
<td>130</td>
<td>65</td>
<td>1491</td>
<td>61%</td>
<td>63%</td>
</tr>
</tbody>
</table>

### REFERENCES


