Process Variation-Aware Timing Optimization for Dynamic and Mixed-Static-Dynamic CMOS Logic
Kumar Yelamarthi, Member, IEEE, and Chien-In Henry Chen, Member, IEEE

Abstract—The advancement in CMOS technology with the shrinking device size towards 32 nm has allowed for placement of billions of transistors on a single microprocessor chip. Simultaneously, it reduced the logic gate delays to the order of pico seconds. However, these low delays and shrinking device sizes have presented design engineers with two major challenges: timing optimization at high frequencies, and minimizing the vulnerability from process variations. Answering these challenges, this paper presents a process variation-aware transistor sizing algorithm for dynamic CMOS logic, and a process variation-aware timing optimization flow for mixed-static-dynamic CMOS logic. Through implementation on several benchmark circuits, the proposed transistor sizing algorithm for dynamic CMOS logic has demonstrated an average performance improvement in delay by 28%, uncertainty from process variations by 32%, while sacrificing an area of 39%. Also, through implementation on benchmark circuits and a 64-b parallel binary adder, the proposed timing optimization flow for mixed-static-dynamic CMOS logic has demonstrated a performance improvement in delay by 17% and uncertainty from process variations by 13%.

Index Terms—Dynamic CMOS logic, mixed-static-dynamic CMOS, process variations, timing optimization, transistor sizing.

I. INTRODUCTION

The semiconductor technology has been advancing rapidly to accommodate the design of several new microprocessors. This advancement along with the shrinking device size has allowed for placement of nearly two billion transistors on Intel’s latest processor [1]. On the other hand, it presented design engineers with two major challenges: timing optimization for designs operating at several giga-hertz frequencies, and minimizing the vulnerability from process variations.

The performance of microprocessors has been driven traditionally by CMOS technology and several micro-architectural improvements [2]. One such method is using custom dynamic CMOS circuits in microprocessors to improve the timing performance [2], [3]. However, dynamic CMOS circuit usage is limited due to many challenges, primarily transistor sizing. Other challenges including charge sharing, noise-immunity, leakage current, environmental and semiconductor process variations etc, have also caused an elevated complexity in timing optimization.

These semiconductor process variations occur when the design parameters deviate from the ideal values pre- and post-fabrication due to changes in the operating environment. In the CMOS technology that is migrating towards 32 nm channel length, parameter variations are not scaling down as fast as nominal values, resulting in increased ratio of variations to nominal value [4]. Inherently, the requirement to account for these process variations has increased significantly.

The magnitude of intra-die channel length variations has been estimated to increase from 35% of total variation in 130 nm to 60% in 70 nm CMOS process, and variation in wire width, height, and thickness is also expected to increase from 25% to 35% [5]. Also, research has shown that process variations may cause up to 30% variation in chip frequency, along with 20X variation in chip leakage [6]. This increased magnitude of variations lowers the design performance by one generation [7] and might even result in design failure [5]. For variation-sensitive circuits such as SRAM arrays and dynamic logic circuits, process variations may result in functional failure and yield loss [5].

Addressing these challenges, this paper presents two timing optimization algorithms: 1) a process variation-aware transistor sizing algorithm for dynamic CMOS logic; and 2) a process variation-aware timing optimization flow for mixed-static-dynamic (MSD) CMOS logic. Utilizing the advantages of both static and dynamic logic CMOS logic, the proposed timing optimization flow partitions a design into static and dynamic logic, and performs timing optimization while accounting for process variations. The research presented is an extended version of our previous work [8].

Both proposed algorithms are validated through implementation on several benchmark circuits and a 64-b binary adder. This paper is organized as follows. Section II introduces previous work on transistor sizing and process variations. Section III presents the process variation-aware transistor sizing algorithm for dynamic CMOS logic. Section IV presents the implementation of process variation-aware transistor sizing algorithm on several benchmark circuits. Section V presents a MSD 64-b adder to validate the performance improvement through MSD logic. Section VI presents the process variation-aware timing optimization flow for MSD CMOS logic and its application to several benchmark circuits. Conclusions are presented in Section VII.
II. BACKGROUND

A. Transistor Sizing

Significant research was performed for timing optimization through transistor sizing [9]–[17], but majority focused on static CMOS circuits and technologies using multiple threshold voltages. TILOS presented in [10] reduces the delay by increasing transistor sizes based on path significance. A major limitation in TILOS is its inability to deal with interacting paths. Also, it does not guarantee the convergence of timing optimization, and hence is not a deterministic optimization technique. MINFLO-TRANSIT [11] is based on an iterative relaxation method that requires generation of iterative directed acyclic graphs, which therefore increases complexity of optimization. Another algorithm proposed for transistor sizing is the convex optimization method [12] that relies on identifying the design space using hyper-planes. Limitations of this algorithm include complexity of finding the half-space for a design with a large number of transistors, requirement of optimization packages, and inability to account for process variations.

Although all these algorithms perform transistor sizing to some extent, they fall short of accounting for important characteristics of the dynamic CMOS circuits. They do not account for capacitance from interacting paths, require optimization packages, generation of directed acyclic graphs, and primarily do not account for process variations.

B. Process Variations

Many researchers have presented techniques to reduce the effect of process variations on chip performance [5], [6], [18]–[21]. Most deal with statistical variations and are not optimal for designs with a large number of parameter variations. A variable strength programmable keeper based on die leakage was proposed in [19], utilizing three keeper transistors in parallel with widths of W, 2W, and 4W. Based on the digital bit input, appropriate keepers are turned on and mapped to an effective keeper width. A major limitation in this method is the requirement for additional hardware to program the keeper transistors. It was shown in [20] that transistors in series stacks are less susceptible to process variations over transistors in parallel stacks. Accordingly, work in [20] proposed insertion of a series dummy transistor in a parallel stack to reduce the impact from process variations. An adaptive body biasing (ABB) technique was presented in [21] to compensate for variation tolerance. The ABB technique is implemented post-silicon where each die receives a unique bias voltage, reducing the variance of frequency variation. Although this method is feasible for the inter-die variation, it is not practical for the intra-die variation, as each block in the design requires a unique bias voltage. Another limitation is increased leakage power from reduction in threshold voltage. Although each method presented here has its unique advantages, none of the algorithms account for both timing optimization and process variations at the same time.

III. LOAD BALANCE OF MULTIPLE PATHS (LBMP) TRANSISTOR SIZING ALGORITHM

Research demonstrated the major contributors towards delay uncertainty to be gate length, channel width, oxide thickness, threshold voltage, supply voltage, junction capacitance, gate width etc., [3]. This delay uncertainty is not only dependent on all the transistors in a respective path, but also on transistors in the interacting paths due to device channel connections.

Consider a 2-b weighted binary-to-thermometric-converter (WBTC) in Fig. 1 with 34 timing paths listed in Table I. Here, the delay of path-34 not only depends on transistors T0, T1, T2, and T3, but also on transistors in the interacting paths. Increasing sizes of T0, T1, T2, and T3 reduces delay of path-34, but increases delay of the interacting paths-25, 31 due to load capacitance from transistor channel-connections. This increase in delay can be minimized through balancing of device channel loading, i.e., transistor sizing in critical and interacting paths. In addition, this method not only reduces the delay, but also the uncertainty from process variations as depicted in (1). Here, \( T_{\text{Max}} \) is the worst-case delay, \( T_{\text{Min}} \) is the best-case delay due to process variations

\[
T_{\text{Uncertainty}} = T_{\text{Max}} - T_{\text{Min}} \tag{1}
\]

Fig. 3 shows the method used to induce process variations during the timing analysis flow. A Monte Carlo parameter generator is used to perturb design parameters such as threshold variation aware load balance of multiple paths (LBMP) transistor sizing algorithm computes delay from the sum of mean and standard deviation, accounting for both intra-die and inter-die variations as in Fig. 2.
values of approximately 10% are used for these parameter variations from the statistics available in [4]. These modified design parameters are input into Cadence Spectre (SPICE simulator) along with the design netlist to obtain the delay profiles (mean, standard deviation, and uncertainty) of all delay paths.

With an insight of increasing the sizes of transistors that appear in the most number of paths reduces delays of most paths, the process variation-aware LBMP algorithm computes the number of paths a transistor is present in and denotes this number as repeats for the respective transistor. In a stack of transistors connected in series, the discharge time of a transistor increases with its relative distance from the output. Accordingly, the LBMP algorithm assigns weight of 0.05 is assigned to stack of transistors closest to the output includes transistors $T_{28}$. However, for the respective transistor, in a stack of transistors connected in series, the discharge time of a transistor increases with its relative distance from the output. Accordingly, the LBMP algorithm assigns weight of 0.05 is assigned to stack of transistors closest to the output includes transistors $T_{28}$. However, for the respective transistor, in a stack of transistors connected in series, the discharge time of a transistor increases with its relative distance from the output. Accordingly, the LBMP algorithm assigns weight of 0.05 is assigned to stack of transistors closest to the output includes transistors $T_{28}$.

<table>
<thead>
<tr>
<th>Path No.</th>
<th>Transistors</th>
<th>Path No.</th>
<th>Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>18</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>2</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>19</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>3</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>20</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>4</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>21</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>5</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>22</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>6</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>23</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>7</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>24</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>8</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>25</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>9</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>26</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>10</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>27</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>11</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>28</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>12</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>29</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>13</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>30</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>14</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>31</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>15</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>32</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>16</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>33</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
<tr>
<td>17</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
<td>34</td>
<td>$T_{2b}$, $T_{18}$, $T_{11}$, $T_{22}$, $T_{26}$</td>
</tr>
</tbody>
</table>

Upon computation of the weight and repeat profiles, process variations are induced to obtain delay profiles using Cadence Spectre. From the delay data obtained, all transistors in the top 20% of critical paths are grouped to set-x, and their sizes are increased as in (2):

\[
\text{New Size} = \text{Old Size} \times \left(1 + \frac{\text{repeats}}{1 + \text{repeats}} \times \text{weight}\right) \quad (2)
\]

\[
\text{Temp New} = \text{Old Size} \times \left(1 - \frac{\text{repeats}}{1 + \text{repeats}} \times \text{weight}\right) \quad (3)
\]

\[
\text{New Size} = \frac{\text{Old Size} + \text{Temp New}}{2} \quad (4)
\]

\[
\text{New Size} = \text{Old Size} \times \left(1 - \frac{\text{repeats}}{1 + \text{repeats}} \times \text{weight}\right) \quad (5)
\]
Fig. 3. Process variation-induced flow for timing analysis.

For instance, if path-1 is one of the top 20% of critical paths, then all transistors in path-1 (T_0, T_4, T_11, T_22, and T_27) are included in set-x transistors.

As delay of a critical path is also dependent on the device channel load, the process variation-aware LBMP algorithm minimizes the channel load by reducing the size of transistors in the interacting paths. All the 1st order connections (transistors with device channels connected) to set-x transistors are grouped to set-y. Given the 2-b WBTC in Fig. 1, the 1st order connection transistors that are channel-connected to T_0 and T_4, T_7, T_14, T_19, T_24, and T_27. Accordingly, set-y for path-1 is comprised of T_0, T_1, T_4, T_7, T_8, T_11, T_14, T_15, T_17, T_19, T_20, T_22, T_24, T_26, and T_27. Next, transistors in set-y that are not in set-x are grouped to set-z. The set-z for path-1 in the 2-b WBTC is comprised of T_1, T_7, T_8, T_14, T_15, T_17, T_19, T_20, T_24, and T_26. Each transistor in set-z is then checked if it is present in the critical path of the previous iteration. If so, its size is decreased by (3) and (4). If not, its size is decreased as in (5).

**IV. TRANSISTOR SIZING EXAMPLE USING LBMP ALGORITHM**

Fig. 1 shows a 2-b weighted binary-to-thermometric-converter (WBTC) used in parallel adders [23], [24]. The weight and repeat profiles of all transistors in the 2-b WBTC based on the LBMP algorithm are computed and shown in Table II. Prior to implementing the process variation-aware LBMP algorithm, the initial sizes of all the transistors are obtained by performing sizing based on [14] to reduce the energy-delay-product. Accordingly, a worst-case delay of the 2-b WBTC was found to be 244 ps from path-1. The top 20% critical paths are path-1, 2, 5, 8, 26, 29. The set-x transistors and their initial sizes on these critical paths are T_0 (311 nm), T_4 (283 nm), T_7 (311 nm), T_11 (283 nm), T_15 (212 nm), T_16 (176 nm), T_17 (234 nm), T_22 (234 nm), T_23 (193 nm), and T_26 (193 nm).

The set-z transistors are increased by (2) to T_0 (454 nm), T_4 (383 nm), T_7 (454 nm), T_11 (389 nm), T_15 (239 nm), T_16 (183 nm), T_17 (274 nm), T_22 (274 nm), T_23 (209 nm), and T_26 (208 nm). The set-z transistors and their sizes are T_1 (257 nm), T_8 (257 nm), T_12 (234 nm), T_13 (193 nm), T_14 (257 nm), T_18 (193 nm), T_19 (257 nm), T_20 (212 nm), T_21 (176 nm), T_24 (212 nm), T_25 (176 nm), and T_26 (76 nm). In order to reduce the device channel load on paths-2, 5, 8, 26, and 29, sizes of transistors in set-z are reduced by (5) to T_1 (195 nm), T_8 (195 nm), T_12 (202 nm), T_13 (180 nm), T_14 (195 nm), T_18 (177 nm), T_19 (195 nm), T_20 (184 nm), T_21 (168 nm), T_24 (190 nm), T_25 (168 nm), and T_26 (171 nm).

After updating the transistor sizes, process variations are induced and spice-level simulations are performed to obtain a new critical path order. After repetitive iterations of the process variation-aware LBMP algorithm, the worst-case path delay converged to 157 ps, an improvement of 35.65%. In addition, the process variation-aware LBMP algorithm reduced delay uncertainty as in (1) from 112 ps to 57 ps, accounting for a 49.1% improvement. Table III shows the 2-b WBTC delay and uncertainty convergence profiles over 10 iterations of the process variation-aware LBMP algorithm.

A comparison of applying the LBMP algorithm to the 2-b WBTC with and without consideration of process variation is shown in Table IV. When process variations are not considered, the nominal delay was 161.37 ps. However, when process variations are considered, the mean delay from the delay distribution...
was 144 ps. By accounting for process variation in the optimization flow, not only the delay reduced from 161.37 ps to 144 ps, but also the area reduced from 2,054 \( \mu \text{m}^2 \) to 1,695 \( \mu \text{m}^2 \). This accounts for an improvement in delay and area by 10.8\% and 17.4\%, respectively, further highlighting the significance of accounting for process variations in timing optimization flow.

With the major contributors towards delay uncertainty being gate length, channel width, capacitance, supply voltage, and threshold voltage, timing analysis was performed to categorize the impact of each. Fig. 4 shows the improvement in delay uncertainty of the optimized 2-b WBTC design from variation in zero-bias junction capacitance. After implementing the process variation-aware LBMP algorithm on the 2-b WBTC, the delay uncertainty from variation in zero-bias junction capacitance reduced from 14\% to 8\%.

Statistical analysis was also performed on the design using t-test to identify if the normalized delay distributions before and after applying the LBMP algorithm are statistically similar. The null hypothesis assumed was “Delay distributions of both the cases are statistically similar.” The alternate hypothesis assumed was “Delay distributions of both cases are not statistically similar.” Results obtained from these t-tests are presented in Table V. For test-1, it was found that t-Stat value was less than the critical value (t \( \text{stat} = -5.48 \) \text{Critical two-tail} = -2.845). For a 95\% confidence interval, based on these test results, the null hypothesis of “Delay distributions of both the cases are statistically similar” is rejected. Also, from the four tested performed, it was clear that the average variance in normalized delay distribution of the post-optimized design (0.0115) is lower than the average variance of the pre-optimized design (0.0632). Based on these statistical tests, it is inferred that the LBMP optimized design results in a lower delay uncertainty from process variations.

Using the Monte Carlo parameter generator in Fig. 3, gate length, and channel width were varied in the 3\% regions, and delay distributions were obtained using Cadence Spectre. Delay distributions for both pre- and post-optimized designs are normalized and their contour plots were generated as in Figs. 5 and 6 respectively. The x-axis and y-axis in these plots represent gate length, channel width, and different colors represent normalized delay variations. From the delay profile of pre-optimized design in Fig. 5, it is evident that delay varies with both gate length and channel width. From the delay profile of post-optimized design in Fig. 6, it is evident that delay varies only with gate length, but not channel width. With balancing of device channel load, sensitivity to variations in channel width of the design is significantly reduced.

Another benchmark used to validate the LBMP algorithm is a 4-b Unity Weight BTC (UWBTC). Along with an increase in the number of transistors, the number of timing paths to be considered is also increased to 83. Prior to implementing the process variation-aware LBMP algorithm, the 4-b UWBTC had a worst-case delay of 114 ps. Through iterative optimizations, the worst-case delay reduced to 103 ps, an improvement of 9.65\%. The process variation-aware LBMP algorithm was also implemented on several benchmarks, and circuits with a low ratio of number of paths to number of transistors. All the benchmarks used were initially optimized using sizing algorithm in [14], and the resulting designs are considered as the initial case for the LBMP algorithm. Implemented and verified on 130 nm CMOS process, results obtained are shown in Table VI.
TABLE VI
LBMP TIMING OPTIMIZATION RESULTS

<table>
<thead>
<tr>
<th>Design</th>
<th># PI</th>
<th># PO</th>
<th># Paths</th>
<th># Trs</th>
<th>Initial Delay (ps)</th>
<th>Final Delay (ps)</th>
<th>Delay Improvement (%)</th>
<th>Area Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCT-2</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>173</td>
<td>109</td>
<td>36.99</td>
<td>-53.12</td>
</tr>
<tr>
<td>2-b WBTC</td>
<td>4</td>
<td>6</td>
<td>34</td>
<td>28</td>
<td>244</td>
<td>157</td>
<td>35.65</td>
<td>-65.52</td>
</tr>
<tr>
<td>4-b UBTC</td>
<td>4</td>
<td>15</td>
<td>83</td>
<td>83</td>
<td>114</td>
<td>103</td>
<td>9.64</td>
<td>-12.66</td>
</tr>
<tr>
<td>74181–CLA</td>
<td>10</td>
<td>6</td>
<td>18</td>
<td>24</td>
<td>152</td>
<td>103</td>
<td>32.23</td>
<td>-53.10</td>
</tr>
<tr>
<td>74181–E Mod</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>174</td>
<td>110</td>
<td>36.78</td>
<td>-54.25</td>
</tr>
<tr>
<td>C2670-CLA</td>
<td>24</td>
<td>1</td>
<td>15</td>
<td>39</td>
<td>330</td>
<td>206</td>
<td>37.57</td>
<td>-58.53</td>
</tr>
<tr>
<td>C3540-CC5</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>96.3</td>
<td>77</td>
<td>20.04</td>
<td>-43.37</td>
</tr>
<tr>
<td>C3540-CC8</td>
<td>7</td>
<td>3</td>
<td>17</td>
<td>35</td>
<td>351</td>
<td>216</td>
<td>38.46</td>
<td>-62.17</td>
</tr>
<tr>
<td>C3540-CC9</td>
<td>8</td>
<td>3</td>
<td>22</td>
<td>47</td>
<td>252</td>
<td>202</td>
<td>19.84</td>
<td>-32.15</td>
</tr>
<tr>
<td>C3540-UM12-7</td>
<td>9</td>
<td>1</td>
<td>24</td>
<td>50</td>
<td>253</td>
<td>178</td>
<td>29.64</td>
<td>-33.46</td>
</tr>
<tr>
<td>C5315-M4-CatP2</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>14</td>
<td>168</td>
<td>137</td>
<td>18.45</td>
<td>-9.25</td>
</tr>
<tr>
<td>C5315-M6-GLC4_2</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>165</td>
<td>122</td>
<td>26.06</td>
<td>-12.22</td>
</tr>
<tr>
<td>C5315-M6-CB4</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>175</td>
<td>134</td>
<td>23.42</td>
<td>-32.37</td>
</tr>
<tr>
<td>C7552-M5-GLC5_1</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>140</td>
<td>95</td>
<td>32.14</td>
<td>-62.84</td>
</tr>
<tr>
<td>C7552-M5-CGC34_4</td>
<td>9</td>
<td>4</td>
<td>14</td>
<td>9</td>
<td>272</td>
<td>161</td>
<td>40.8</td>
<td>-51.87</td>
</tr>
<tr>
<td>C7552-M5-CGC17</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>102</td>
<td>84</td>
<td>17.64</td>
<td>-21.33</td>
</tr>
<tr>
<td>C7552-M5-CGC20</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>100</td>
<td>78</td>
<td>22</td>
<td>-19.36</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>28%</strong></td>
<td><strong>-39%</strong></td>
</tr>
</tbody>
</table>

Fig. 6. Impact of delay variation in gate length and channel width post-optimization.

V. MIXED-STATIC-DYNAMIC 64-BIT ADDER

With static and dynamic CMOS logic having perspective advantages of power and speed, a design performance can be improved through a balanced use of both logic styles. A 64-b adder shown in Fig. 7 is used here as a test case for timing optimization. This adder is divided into two blocks operating in parallel. Block-1 is comprised of a 64-b Carry Convergent Tree (CCT) and a Carry Generator (CG) as in Fig. 8. Block-2 is comprised of eight 8-b carry-select adders; each 8-b carry-select adder is designed with four 2-b Thermometric Adders (TA) [8] connected in a ripple fashion. The Block-1 of 64-b adder computes the seven intermediate carry outputs (C8, C16, C24, C32, C40, C48, C56) that are used as select lines for carry-select adders in Block-2. Upon receiving the intermediate carry inputs from Block-1, Block-2 selects the corresponding pre-computed partial sum to be the end result. The 2-b TA consists of Binary to Thermometric Converters and a Final Sum (FS) block [8] (comprised of a Thermometric-to-Abacus Converter with add-1 logic, a Thermometric-to-Abacus Converter with add-0 logic, Abacus-to-Binary Converter, and multiplexers).

The 64-b adder is partitioned into a MSD circuit style and designed in four combinations as shown in Table VII. The 64-b adder designed with CCT, CG, and WBTC using dynamic CMOS and FS using static CMOS had the least delay of 632 ps and PDP of 84.17 pJ. Changing the 2-b WBTC to static CMOS reduced the power from 133.19 to 125.34 mW, accounting for an improvement of 5.8%. However, delay increased from 632 to 1462.33 ps, accounting for a 131.38% increase. Here, changing the 2-b WBTC from dynamic to static CMOS logic increased the delay by 180 ps. With the 8-b carry select adders made of four 2-b WBTC’s connected in ripple mode, a slight increase in the delay of 2-b WBTC significantly increased the delay of 64-b adder. Furthermore, changing the CG to static CMOS reduced the power from 133.19 to 125.02 mW, accounting for 6.3% improvement. However, delay increased
from 632 to 1646.5 ps, accounting for a 160.52% increase (the delay increase is primarily from the 2-b WBTC). With CCT and WBTC in dynamic CMOS logic, and CG and FS in static CMOS logic, the 64-b adder consumes an average power of 133.45 mW. But, the delay is increased from 632 ps to 862.4 ps, accounting for a 36.45% increase.

A comparison of implementing the LBMP algorithm to CCT blocks and 2-b WBTC of the 64-b adder with and without consideration of process variation during timing optimization is shown in Table VIII. When the CCT blocks and 2-b WBTC are optimized without considering process variations, the worst-case delay of the 64-b adder in case-1 was 686 ps. By considering process variations in the LBMP algorithm, delay reduced from 686 to 632 ps, and power-delay-product reduced from 91.6 pJ to 84.17 pJ. This accounts for an improvement in delay and power-delay-product by 8%. Similarly, considering process variations in case-4 reduced the delay and power-delay-product by 3% further highlighting the significance of considering for process variations in the timing optimization flow.

VI. PROCESS VARIATION-AWARE POINT OPTIMIZATION

At the architecture level, one common limitation in most design optimization flows is lack of accounting for process variations. After placement and route, if a design fails to meet the timing constraints, the optimization flow is re-iterated. After several design iterations, the design might still not meet the timing constraint, and miss the time-to-market window. The proposed process variation-aware Path Oriented In Time (POINT) optimization flow in Fig. 9 answers these challenges by accounting for process variations during iterative timing optimization.

In Section V, it was shown that MSD logic results in better timing over conventional static CMOS logic alone. The process variation-aware POINT optimization performs timing optimization through effective use of both static and dynamic CMOS logic styles.

Initially, a high-level description of a design is input to Cadence Encounter [25]. Following synthesis and optimization, static timing analysis (STA) is performed to identify the timing critical modules, and the amount of delay to be reduced. Accordingly, the critical modules are identified based on the number and significance of critical delay paths traversing through them. After 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.

Later, the timing critical modules in static CMOS logic are replaced with performance optimized dynamic CMOS logic. With the updated design comprising dynamic CMOS logic blocks, both clock tree design and timing verification are performed. After the design is timing-verified with clock signal timing constraints, incremental STA is performed to verify for timing convergence. After several iterations, if the optimized design still does not meet the timing, new timing critical modules are identified, and the design is further optimized. Following the timing convergence through iterations, the final MSD circuit design is exported for placement and route.

A test case used to validate the POINT optimization flow is an architectural-level description of ISCAS c7552, a 32-b adder, and magnitude comparator as in Fig. 10 [26]. Synthesis, optimization, and STA were performed using Cadence Encounter. For the design in hierarchical format, the worst-case path delay
Fig. 9. Process variation-aware POINT optimization flow.

was found to be 3.01 nsec (synthesis and optimization was performed at block level, and design flatten option was disabled). From the STA report, the critical module in terms of delay was found to be M5/UM5_1/CC_1/CYG34_0 with a delay of 960 ps as highlighted in Fig. 10. As, it is not desirable to replace the entire module M5 with dynamic logic, only sub-modules that appear in most of the critical paths were selected, from the Cadence Encounter timing report, and are replaced by dynamic CMOS logic. The first two sub-modules selected from M5 are CGC20 with a delay of 559 ps and CGC17 with a delay of 390 ps. In addition, other sub-modules chosen from M5 are GLC9_0/GLC5_1 with a delay of 720 ps and CGC34_4 with a delay of 660 ps.

After the timing-critical sub-modules were selected, dynamic CMOS logic were designed and process variation-aware timing optimization was performed. Through implementation of the LBMP algorithm, delays of CGC20, CGC17, GLC9_0/GLC5_1, and CGC34_4 in M5 were reduced by 22%, 17%, 32%, and 40%, respectively. After the static CMOS logic of these sub-modules are replaced by the LBMP-optimized dynamic CMOS logic, post-optimization incremental STA was performed. After POINT optimization, the c7552 was tested at various corners of operation and its delay and delay uncertainty from process variations and was reduced by 20.59% and 22.02%, respectively.

Similarly, the process variation-aware POINT optimization was implemented on other benchmark circuits and results obtained are presented in Table IX. Applying the POINT optimization to five benchmark circuits, delay and uncertainty reduced by an average of 16.94% and 13.14%, respectively, over designs optimized with Cadence Encounter.

VII. CONCLUSION

In this paper, challenges in timing optimization of CMOS logic have been presented. It was first shown that importance and complexity in timing optimization of dynamic CMOS logic increases with the number of timing paths and the magnitude of process variations. To gain insights in the area of process variations and timing optimization, this paper presented: 1) a process variation-aware Load Balance of Multiple Paths algorithm for timing optimization of dynamic CMOS logic; and 2) a process variation-aware Path Oriented IN Time optimization flow for MSD CMOS logic. Through implementation on ISCAS benchmark and custom circuits, the process variation-aware LBMP algorithm has demonstrated an average improvement in delay and uncertainty by 28% and 32%, respectively, while sacrificing an area of 39%. Also, through implementation on five benchmark circuits, the process variation-aware POINT optimization flow demonstrated an average improvement in delay and uncertainty by 16% and 13% over results obtained with Cadence Encounter.
ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for insightful comments to improve the quality of presentation.

REFERENCES


Kumar Yelamarthi (S’02–M’04) received the Ph.D. degree in electrical engineering from Wright State University, Dayton, OH, in 2008. He is currently an Assistant Professor of Electrical Engineering at Central Michigan University, Mt. Pleasant, MI. His research interest is in the area of timing optimization, computer-aided design, semiconductor process variations, multidisciplinary VLSI design, and engineering education. He has served as a technical reviewer for several IEEE/ASME/ASEE international conferences and journals, and has written over 35 publications in both technical and educational fields.

Dr. Yelamarthi is a member of Tau Beta Pi engineering honor society, and Omicron Delta Kappa national leadership honor society.

Chien-In Henry Chen (S’89–M’89) received the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 1989.

Since joining Wright State University, Dayton, OH, in 1989, he has worked primarily in design and test of digital, mixed-signal IC and system-on-a-chip (SoC), specifically VLSI and FPGA for signal processing, communications, and digital microwave receivers, where he is currently a Professor. His research was supported by DoD, federal agencies, industrial companies, and the State of Ohio. He has written over 100 publications in IEEE transactions and journals, international journals, and IEEE conference proceedings.

Dr. Chen was a plenary speaker in the 6th VLSI Design/CAD Symposium and served as a Guest Editor of VLSI Design Journal in 2002. He has been a technical committee member of IEEE International ASIC/SOC Conference, IEEE International Instrumentation and Measurement Technology Conference, Annual Conference of the IEEE Industrial Electronics, and IEEE International Symposium on Circuits and Systems. He has consulted for a number of U.S. semiconductor companies.