Research Article

Dynamic CMOS Load Balancing and Path Oriented in Time Optimization Algorithms to Minimize Delay Uncertainties from Process Variations

Kumar Yelamarthi and Chien-In Henry Chen

1 School of Engineering and Technology, Central Michigan University, Mt Pleasant, MI 48859, USA
2 Department of Electrical Engineering, Wright State University, Dayton, OH 45435, USA

Correspondence should be addressed to Kumar Yelamarthi, kumar.yelamarthi@cmich.edu

Received 2 June 2009; Revised 19 October 2009; Accepted 3 December 2009

Academic Editor: Ethan Farquhar

Copyright © 2010 K. Yelamarthi and C.-I. H. Chen. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The complexity of timing optimization of high-performance circuits has been increasing rapidly in proportion to the shrinking CMOS device size and rising magnitude of process variations. Addressing these significant challenges, this paper presents a timing optimization algorithm for CMOS dynamic logic and a Path Oriented IN Time (POINT) optimization flow for mixed-static-dynamic CMOS logic, where a design is partitioned into static and dynamic circuits. Implemented on a 64-b adder and International Symposium on Circuits and Systems (ISCAS) benchmark circuits, the POINT optimization algorithm has shown an average improvement in delay by 38% and delay uncertainty from process variations by 35% in comparison with a state-of-the-art commercial optimization tool.

1. Introduction

The performance improvement of microprocessors has been driven traditionally by dynamic logic and microarchitectural improvements [1] and can be further enhanced through circuit design and topology organization. Dynamic logic is an effective logic style in terms of timing and area when compared to its static counterpart due to (1) the absence of requirement for design implementation in complementary PMOS logic, and (2) the use of a clock signal in its implementation of combinational logic circuits. In general, CMOS dynamic logic uses fast NMOS transistors in its pull-down network. Its delay is dependent on the number and size (width) of transistors in the NMOS critical path. This paper presents an NMOS transistor sizing optimization for a faster operation.

Static logic is slower because it has twice the loading, higher thresholds, and actually uses slow PMOS transistors for computation. Dynamic logic has been predominantly used in microprocessors, and their usage has increased the timing performance significantly over static CMOS circuits [1, 2]. However, timing optimization of dynamic logic is challenging due to several issues such as charge sharing, noise-immunity, leakage, and environmental and semiconductor process variations. Also, with dynamic circuits consuming more power over static CMOS, an optimal balance of delay and power can be achieved at the architectural level through effective partitioning of design into a mixed-static-dynamic circuit style [3].

Process variations introduce design uncertainties at each step of process development, design, manufacturing, and test. The ratio of these process variations to the nominal values has been increasing with the shrinking device size towards 32 nm [4], causing an impending requirement to account for process variations during timing optimization. They need to be taken into account during the design phase to make sure that performance analysis provides an accurate estimation [5].

One of the challenges in timing optimization of CMOS logic is delay uncertainty (\(\Delta\)) from process variations, \(\Delta = T_{\text{max}} - T_{\text{min}}\), where \(T_{\text{max}}\) and \(T_{\text{min}}\) are the maximum and minimum delays of a timing path. In the 180 nm CMOS
technology, these process variations have caused about 30% variation in chip frequency, along with 20X variation in current leakage [6]. The magnitude of intradie channel length variations has been estimated to increase from 35% of total variations 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% [7]. In CMOS 65 nm process, the parameters that affect timing the most are device length, threshold voltage, device width, mobility, and oxide thickness [8]. For process variation sensitive circuits such as SRAM arrays and dynamic logic circuits, these process variations may result in functional failure and yield loss [7].

Addressing the challenges of timing optimization and delay uncertainty from process variations, this paper presents a timing optimization algorithm for dynamic circuits, and a timing optimization flow for mixed-static-dynamic CMOS logic. The proposed algorithm and flow are validated through implementation on several benchmark circuits and a 64-b binary adder in 130 nm CMOS process. This paper is an extension of our previous work [9] and is organized as follows. Section 2 presents previous work on transistor sizing (width) optimization for timing and process variation minimization. Section 3 presents the proposed transistor sizing optimization for CMOS dynamic logic. Section 4 presents implementation and results of several ISCAS benchmark circuits. Based on this timing optimization algorithm for dynamic circuits, Section 5 presents a timing optimization flow for mixed-static-dynamic CMOS logic and its implementation and results of several ISCAS benchmark circuits. Finally, conclusion is presented in Section 6.

2. Previous Work

Methods of automating transistor sizing for timing optimization were proposed in [3, 10–16], but many of them focus on static CMOS circuits and technologies using multiple threshold voltages. Timed Logic Synthesizer (TILOS) [10] presented an algorithm of iteratively sizing transistors by a certain factor in the critical path. The algorithm is not a deterministic approach, as it does not guarantee convergence in timing optimization. MINFLOTRANSIT [11] is another algorithm proposed for transistor sizing based on iterative relaxation method but requires iterative generation of directed acyclic graphs for every step of timing optimization. Computation of “Logical Effort” is the other method proposed for timing optimization [16]. However, it has two limitations. First, it requires estimation of input capacitance, of which circuits with complex branches or multiple paths have difficulty in accurate estimate. Second, it optimizes timing at the cost of increased area [17].

Methods to mitigate the effect of process variations in CMOS circuits were proposed in [6, 7, 18–23]. These methods deal with statistical variations and are not optimal for designs with large number of parameter variations [24]. A technique called Adaptive Body Biasing (ABB) was presented in [6, 22] to compensate for variation tolerance. The ABB technique is implemented in postsilicon where each die receives a unique bias voltage, reducing the variance of frequency variation. However, this method does not minimize intradie variations, as each block in the design requires a unique bias voltage. Another limitation is the increasing leakage power, caused by the reduction of threshold voltage. Programmable keepers were proposed to compensate for process variations in [23]. This method works for designs with large number of parallel stacks (similar to the NOR gates). However, it requires additional hardware to program the keeper transistors for other designs.

Research has shown that intradie variations primarily impact the mean delay, and interdie variations impact the variance of delay [18]. As timing optimization should consider both interdie and intradie variations, both mean and variance should be accounted for. In addition to optimizing path delay, other parameters affected by process variations that need to be considered and reduced are delay uncertainty ($\Delta = T_{\text{max}} - T_{\text{min}}$) and sensitivity ($\delta = \sigma/\mu$), where $T_{\text{max}}$ and $T_{\text{min}}$ are the maximum and minimum delays, $\mu$ is the mean delay, and $\sigma$ is the standard deviation of delay distribution.

3. Transistor Sizing Optimization of Dynamic Circuits

The delay of dynamic circuit is highly dependent on the number and size (width) of transistors in the critical path. Increasing width of transistors in a path will increase the discharging current and reduce the output pull-down path delay. However, increasing width of transistors to reduce one path delay may increase the capacitive load of channel-connected transistors on other paths and substantially increase their delays. This complexity increases along with the number of paths present in the circuit. A 2-b Weighted Binary-to-Thermometric Converter (WBTC) that is used in high-performance binary adders [25] shown in Figure 1 is used as an example to explain the path delay optimization complexity while considering process variations.

Figure 1 highlights two timing paths: path-A ($T_{26}$,$T_7$,$T_8$,$T_{12}$,$T_{18}$,$T_{32}$) and path-B ($T_{28}$,$T_6$,$T_9$,$T_{11}$,$T_{15}$,$T_{16}$,$T_{31}$). A test was performed to optimize path-A by gradually increasing widths of $T_7$, $T_8$, $T_{12}$, and $T_{18}$. It was observed that the delay of path-A reduced by 4%, but delay of path-B increased by 9.3%. This is a result of transistors on path-B being channel-connected to transistors on path-A. For instance, $T_4$ and $T_{14}$ are channel-connected to $T_7$ and $T_8$, and $T_{15}$ and $T_{16}$ are channel-connected to $T_{12}$ and $T_{18}$. Increasing widths of $T_7$, $T_8$, $T_{12}$, and $T_{18}$ in path-A causes the capacitive load of $T_4$, $T_{11}$, $T_{15}$, and $T_{16}$ to increase and therefore increase delay of path-B. This circuit example illustrates that increasing widths of transistors on one critical path increases capacitive load and delay of the other critical paths.

Conventionally, a path delay is denoted by the mean ($\mu$) of its delay distribution, which accounts only for intradie variations. As interdie variations are equally important, its standard deviation ($\sigma$) is as important and should be considered as well. Consider the delay distribution of two paths of WBTC shown in Figure 2. Path-B has a high mean delay, while path-A has a high standard deviation. Typically, path-B would be chosen as the critical path for timing optimization as it has the highest mean delay ($\mu$). Optimizing the design by increasing width of transistors on path-B may...
Consider the circuit with a series of $n$ channel-connected NMOS transistors, $T_1, T_2, \ldots, T_n$, in Figure 4. While $T_n$ conducts the discharge current of the load capacitance $C_L$, $T_1$ conducts the discharge current from a total capacitive load, $C_{\text{total}} = C_L + \cdots + C_3 + C_2 + C_1$, which is substantially large when $n$ increases. So, the discharge time of $T_1$, the transistor near Gnd, is longer than $T_n$, the transistor near output. Accordingly, accounting for the discharge times increasing from $T_n$ to $T_1$, the transistor sizes are made progressively larger, starting from a minimum-size transistor at $T_n$ to reduce the total discharge time of the pull-down path (out, $T_n, \ldots, T_2, T_1$). The width of the next to the last transistor is scaled up by a factor. In the proposed LBMP algorithm we assign a weight (used for transistor sizing) in the range of 0.05–0.5 to each transistor relative to its distance from the output for this reason. For instance, the 2-b WBTC in Figure 1 is comprised of seven transistor stacks relative to their distance from the output. Stack-1, closest to the output, includes transistors $T_3, T_{10}, T_{16}, T_{21}, T_{25}$, and $T_{27}$. Stack-2 includes transistors $T_6, T_{13}, T_{18}, T_{23}$, and $T_{26}$. Stack-3 includes transistors $T_9, T_{15}, T_{20}$, and $T_{24}$. Stack-4 includes transistors $T_{12}, T_{17}$, and $T_{22}$. Stack-5 includes transistors $T_1, T_8, T_{14}$, and $T_{19}$. Stack-6 includes transistors $T_4$ and $T_{11}$. Stack-7 farthest from the output includes transistors $T_0$ and $T_7$. Accordingly, transistors in stacks 1–7 are assigned weights of 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, and 0.5, respectively. For designs with different number of stacks, weights of transistors reduce the mean delay ($\mu$) but may not reduce its standard deviation ($\sigma$) as well. However, by considering both the mean and the standard deviation (i.e., $\mu + \sigma$), path-A would be chosen as the critical path to be optimized. As both interdie and intradie variations are equally important, the proposed timing optimization algorithm ranks critical paths based on the sum of the path mean delay and its standard deviation, $(\mu + \sigma)$. The Load Balance of Multiple Paths (LBMPs) timing optimization algorithm proposed for transistor sizing of dynamic circuits while considering process variations is presented in Figure 3.
Identify all timing paths and assign weights to each transistor

Perform initial transistor sizing with a fixed ratio ‘r’

Induce process variations and obtain delay profiles of all paths

Choose top 20% critical paths based on (μ+σ)

Identify all transistors in the critical paths (set-x)

Increase size of transistors in set-x
new size = old size (1+(repeats/(1+repeats))×weight)

Identify all first order connections (set-y) to set-x

Choose set-z from set-y based on the criteria that, none of them are in critical paths

For each transistor in set-z, if transistor is in critical paths of previous iteration
Decrease size by
New size = old size (1−(repeats/(1+repeats))×weight)

Decrease size by new size = old size (1−(repeats/(1+repeats))×weight)

Induce process variations and obtain delay profiles of all paths

Solution acceptable

Yes

No

STOP

Figure 3: LBMP Transistor sizing algorithm.

Figure 4: Transistor sizes are made progressively larger.

As the delay of critical path is dependent on the capacitive load of channel-connected transistors, reducing this capacitive load reduces the overall delay. The 1st-order connection transistors in set-x are identified and grouped to set-y. Then, transistors in set-x are excluded from set-y to form set-z. For each transistor in set-z, it is checked if the transistor is present in set-x of previous iteration. If so, its width is decreased and calculated by (2) and (3). If not, its width is decreased and calculated by (4). Once new transistor widths are determined, simulations are performed to locate the new critical paths. This algorithm is repeated until a convergence of an optimal solution is obtained:

\[
\text{TempNew} = \text{Old Size} \left(1 - \frac{\text{Repeats}}{1 + \text{Repeats}}\right) \times \text{Weight},
\]

(2)

\[
\text{New Size} = \frac{\text{Old Size} + \text{TempNew}}{2},
\]

(3)

\[
\text{New Size} = \text{Old Size} \left(1 - \frac{\text{Repeats}}{1 + \text{Repeats}}\right) \times \text{Weight}.
\]

(4)


A 2-b Weighted Binary-to-Thermometric Converter (WBTC) used in high-performance binary adders was shown in Figure 1 [25]. This circuit is used as an example to illustrate the complexity of transistor sizing optimization. With less than 50 transistors, the 2-b WBTC has 34 timing paths, and of which path delays change dramatically with different transistor sizes.

The timing paths of the 2-b WBTC are shown in Table 1, and transistor repeat and weight profiles are shown in
Prior to optimization, the worst-case delay of 2-b WBTC was 355 psec from path-1. The top 20% critical paths are path-1, 2, 5, 8, 26, and 29. Widths of all transistors in these critical paths are initially increased by a ratio of 1.1 to their initial values. For example, the sizes of transistors ($T_{22}$, $T_{11}$, $T_4$, and $T_9$) in path-1 are increased to $160 \times 1.1 = 176$ nm, $160 \times 1.1^2 = 193$ nm, $160 \times 1.1^3 = 213$ nm, $160 \times 1.1^4 = 234$ nm, respectively. After initial transistor sizing, process variations are considered in simulations in which delay distribution of each path is obtained. Then, transistor sizes are updated using (1)–(4), and simulations are performed to obtain a new critical path order. After several iterations of the LBMP timing optimization algorithm, the worst-case delay of 2-b WBTC is reduced and finally converged to 157 psec, accounting for a 55.77% improvement.

Efficiency of the LBMP algorithm is further illustrated through reduction in delay uncertainty ($\Delta$). Figures 5 and 6 show the normalized delay distribution of the 2-b WBTC before and after optimization, respectively. It is clearly evident that delay uncertainty has reduced and distribution has been narrowed significantly in the optimized design.

With major contributors towards delay uncertainty being gate length, channel width, capacitance, supply voltage, and threshold voltage [5], timing analysis was performed to categorize the impact of each. Figure 7 shows the reduction in delay uncertainty of 2-b WBTC from 14% to 8% due to variation in zero-bias junction capacitance.

Kinget in his work on device mismatch [27] has shown that variance in delay distribution of $\Delta V_T$ is dependent on device area, $W \times L$ as shown in (5), where $W$ is the transistor width, $L$ is the transistor channel length, and $A_{VT}$ is the proportionality constant (a technology dependant value). Experimental results of delay impact from variation in oxide thickness are shown in Figure 8 where increasing device area after transistor sizing reduces the delay uncertainty of the 2-b WBTC from 24% to 15%. The other research [5] shows that a drop in supply voltage degrades cell timing at a quadratic rate; a 5% drop in total rail-to-rail voltage may result in a 15% timing degradation. Figure 9 shows the delay uncertainty of the 2-b WBTC before and after optimization using the LBMP algorithm. It is observed that a 20% drop in total rail-to-rail voltage (from 1.0 V to 0.8 V) results in a 4% variation in timing (much less than 15%), which
variation in supply voltage:

\[ \sigma^2(\Delta V_T) = \frac{A_{VT}^2}{W \cdot L} \]  

further illustrates that the LBMP algorithm is less sensitive to variation in supply voltage:

Another benchmark used to validate the algorithm is a 4-b Unity Weight BTC (UWBTC) that is used in high-performance digital-to-analog converters, as shown in Figure 10. Along with an increase in the number of transistors, the number of timing paths to be considered is also increased to 83. Prior to optimization, the 4-b UWBTC had a worst-case delay of 152 psec. Through iterative optimization using the LBMP algorithm, the worst-case delay of 4-b UWBTC was reduced to 103 psec, an improvement of 33%. Furthermore, the LBMP algorithm was also implemented on several ISCAS benchmark circuits of which the ratio of reduction by 48%, power increase by 13%, and an area increase by 39.8%. The delay convergence profiles of these circuits are shown in Figure 11.

As delay in general can be reduced by increasing power consumption [28], power-delay product (PDP) is a key evaluation parameter to compare the design performance among different circuit structures. Table 5 shows the PDP of benchmark circuits before and after optimization.
Figure 10: 4-bit Unity Weight BTC.
optimal sizing of transistor widths, the proposed LBMP timing optimization algorithm has reduced the PDP by an average of 40.17%.

The other electronic performance measurement associated with timing optimization is delay sensitivity ($\partial$) due to process variations. Traditionally, CMOS device switching speed improves at a lower temperature due to increase in mobility. However, Negative Bias Temperature Instability (NBTI) effects may degrade the device switching speed over time via threshold voltage shifts in PMOS transistors [29, 30], even at a lower temperature. The delay sensitivities of several ISCAS benchmark circuits, due to process variations at different temperatures are reported in Figure 12. It is observed that all circuits after timing optimization have a very little difference in the delay sensitivity reduction for different temperatures. The LBMP timing optimization provides consistent delay sensitivity at different temperatures.
Table 5: Power Delay Product optimization results from the proposed algorithm.

<table>
<thead>
<tr>
<th>Design</th>
<th>Initial PDP (fWs)</th>
<th>Final PDP (fWs)</th>
<th>PDP Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCT-2</td>
<td>3.57</td>
<td>1.87</td>
<td>47.49</td>
</tr>
<tr>
<td>2-b WBTC</td>
<td>13.66</td>
<td>6.185</td>
<td>54.74</td>
</tr>
<tr>
<td>4-b UWBTC</td>
<td>6.61</td>
<td>5.11</td>
<td>22.57</td>
</tr>
<tr>
<td>74181-CLA</td>
<td>5.04</td>
<td>3.04</td>
<td>39.55</td>
</tr>
<tr>
<td>74181-E Mod</td>
<td>3.55</td>
<td>1.90</td>
<td>46.46</td>
</tr>
<tr>
<td>C2670-CLA</td>
<td>2.97</td>
<td>1.75</td>
<td>41.07</td>
</tr>
<tr>
<td>C3540-CC5</td>
<td>1.09</td>
<td>0.66</td>
<td>38.78</td>
</tr>
<tr>
<td>C3515-CalP2</td>
<td>7.47</td>
<td>4.57</td>
<td>38.78</td>
</tr>
<tr>
<td>C3515-GLC4_2</td>
<td>3.68</td>
<td>2.70</td>
<td>26.48</td>
</tr>
<tr>
<td>C3515-CB4</td>
<td>4.61</td>
<td>2.93</td>
<td>36.43</td>
</tr>
<tr>
<td>C7552-GGC34_4</td>
<td>16.42</td>
<td>6.19</td>
<td>62.26</td>
</tr>
<tr>
<td>C7552-GGC17</td>
<td>1.63</td>
<td>1.15</td>
<td>28.97</td>
</tr>
<tr>
<td>C7552-GGC20</td>
<td>1.09</td>
<td>0.67</td>
<td>38.70</td>
</tr>
<tr>
<td>Average (%)</td>
<td></td>
<td></td>
<td>40.17</td>
</tr>
</tbody>
</table>

At the architecture level, a common limitation in most design optimization flows is the limited accountability for process variations. Typically after placement and route, if a design fails to meet the timing constraints, optimization flow is reiterated. Even after several iterations, design may still not meet the timing constraint and miss the time-to-market window. The process variation-aware Path Oriented IN Time (POINT) optimization algorithm proposed in Figure 13 answers these challenges of timing optimization and also accounts for process variations. Utilizing the LBMP algorithm proposed in Section 3, the POINT optimization algorithm partitions the design to effectively utilize both dynamic and static CMOS logic to meet the timing constraints.

Initially, a high-level description of a design is input to Synopsys Design Compiler (SDC) [31] for synthesis and optimization. The optimized designs from SDC are considered as the initial case for POINT optimization flow. Following synthesis and optimization, static timing

5. Timing Optimization of Mixed-Static-Dynamic Circuits

Conventionally, synthesis tools perform design and optimization using static CMOS logic [31, 32]. It is not uncommon for the synthesis tools to not find an acceptable solution in terms of timing. This challenge can be answered through utilizing the advantage of fast timing in dynamic logic. Dynamic logic has smaller gate capacitances compared to their static CMOS counterparts, which accounts for a significant speedup [3, 33]. With static and dynamic logic having their respective advantages of low power and low delay, an optimal balance can be obtained by partitioning the design to use both static and dynamic logic in an effective manner.

At the architecture level, a common limitation in most design optimization flows is the limited accountability for process variations. Typically after placement and route, if a design fails to meet the timing constraints, optimization flow is reiterated. Even after several iterations, design may still not meet the timing constraint and miss the time-to-market window. The process variation-aware Path Oriented IN Time (POINT) optimization algorithm proposed in Figure 13 answers these challenges of timing optimization and also accounts for process variations. Utilizing the LBMP algorithm proposed in Section 3, the POINT optimization algorithm partitions the design to effectively utilize both dynamic and static CMOS logic to meet the timing constraints.
Perform synthesis and optimization

Perform static timing analysis

Find timing critical modules

Design custom dynamic CMOS circuits for timing critical modules

Optimize dynamic circuits using LBMP timing optimization algorithm

Solution converged?

Yes

No

Replace static timing critical modules with custom dynamic circuits

Design clock tree and perform timing constraint verification

Perform incremental static timing analysis

Solution converged?

Yes

No

Export netlist for placement and routing

Figure 13: POINT Optimization Algorithm.

Figure 14: ISCAS benchmark–C3540 (8-b ALU).

analysis (STA) is performed using Synopsys PrimeTime (SPT) to identify the critical paths. Also, the critical timing modules identified by the number of occurrences, and delay significance on the critical paths are reported and dynamic circuits of the same are designed. Using the LBMP algorithm, iterative transistor sizing optimization for timing is performed on these dynamic circuits.

With the updated design comprising of dynamic logic circuits, clock tree design and timing verification is performed. After the design is verified for clock signal timing constraints, incremental STA is performed to verify for timing convergence. The algorithm is iteratively repeated towards convergence of acceptable solution. Following the timing convergence through iterations, the final mixed-static-dynamic circuit design is exported for placement and route.

The POINT optimization algorithm is verified through implementation on several ISCAS benchmark circuits, including C3540, an 8-b ALU as shown in Figure 14 [34]. Initial synthesis and optimization was performed using SDC, and static timing analysis was performed using SPT...
Table 6: POINT Optimization Flow results.

<table>
<thead>
<tr>
<th>Design</th>
<th># Inputs</th>
<th># Outputs</th>
<th># Gates</th>
<th>Delay reduction (%)</th>
<th>Uncertainty reduction (%)</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>adder64</td>
<td>130</td>
<td>65</td>
<td>1491</td>
<td>61</td>
<td>63</td>
</tr>
<tr>
<td>c3540</td>
<td>50</td>
<td>22</td>
<td>1669</td>
<td>32</td>
<td>40</td>
</tr>
<tr>
<td>c5315</td>
<td>178</td>
<td>123</td>
<td>2406</td>
<td>29</td>
<td>32</td>
</tr>
<tr>
<td>c7552</td>
<td>207</td>
<td>108</td>
<td>3512</td>
<td>26</td>
<td>31</td>
</tr>
</tbody>
</table>

Average (%)

<table>
<thead>
<tr>
<th>Delay reduction (%)</th>
<th>Uncertainty reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>35</td>
</tr>
</tbody>
</table>

Figure 17: Timing Profiles using POINT Optimization Algorithm.

[35]. For the design in hierarchical format (synthesis and optimization was performed at block level, and design flatten option was disabled), the critical path delay was found to be 3.6 nanoseconds. The critical modules and the critical paths obtained from STA are highlighted in Figure 14. Based on the STA report, it is shown that the ALU Core-M5 with a delay of 1.24 nanoseconds is the timing critical module with the most number of worst-case paths. Figure 15 shows the schematic of UM5_6 from ALU Core-M5 with the critical paths highlighted; the submodules labeled CC5 and CC9 are timing critical with delays of 0.5 nanoseconds and 0.61 nanoseconds respectively.

With timing optimization being the primary goal in this stage, submodules CC5 and CC9 in M5/UM5_6 of C3540 are designed in dynamic logic, and timing optimization is performed using the LBMP algorithm. With dynamic circuits optimized using LBMP algorithm, the delay of CC5 was reduced from 0.5 nanoseconds to 0.07 nanoseconds, and delay of CC9 was reduced from 0.61 nanoseconds to 0.20 nanoseconds, respectively. After the first iteration of the POINT optimization flow, the critical path delay of C3540 was reduced from 3.6 nanoseconds to 2.8 nanoseconds. Further iterations of POINT optimization flow reduced the critical path delay from 3.6 nanoseconds to 2.4 nanoseconds, as shown in Figure 16. In addition to reducing the delay by 33%, the delay uncertainty due to process variations is also reduced by 40% as shown in Figure 17.

Similarly, the process variation-aware POINT optimization was implemented on other benchmark circuits and timing optimization results are presented in Table 6, where both delay and uncertainty from process variations were reduced by an average of 38% and 35%, respectively, over initial designs optimized with Synopsys Design Compiler.

6. Conclusion

In this paper a process variation-aware timing optimization of dynamic logic and a timing optimization flow for mixed-static-dynamic CMOS logic have been presented. Solutions addressing further design challenges are presented by (1) considering delay uncertainties from process variations and (2) developing a process variation-aware Path Oriented IN Time (POINT) optimization algorithm for mixed-static-dynamic logic.

Through implementation and verification of several benchmark circuits in 130 nm CMOS process, the process variation-aware timing optimization algorithm has shown by average a delay reduction by 47.8%, an uncertainty reduction by 48%, a power increase by 13%, and a power-delay-product reduction by 40%. Validated through implementation of mixed-static-dynamic logic on a 64-b adder and several ISCAS benchmark circuits, the POINT optimization algorithm has demonstrated an average improvement in delay reduction by 38% and delay uncertainty reduction from process variation by 35%.

7. Acknowledgment

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


Submit your manuscripts at http://www.hindawi.com