DDRO: A Novel Performance Monitoring Methodology Based on Design-Dependent Ring Oscillators

Tuck-Boon Chan†, Puneet Gupta§, Andrew B. Kahng†‡ and Liangzhen Lai§

UC San Diego ECE† and CSE‡ Departments, La Jolla, CA 92093
UC Los Angeles EE§ Department, Los Angeles, CA 90095
Outline

• Performance Monitoring: An Introduction
• DDRO Implementation
• Delay Estimation from Measured DDRO Delays
• Experiment Results
• Conclusions
Performance Monitoring

• Process corner identification
  – Adaptive voltage scaling, adaptive body-bias

• Runtime adaptation
  – DVFS

• Manufacturing process tuning
  – Wafer and test pruning [Chan10]
Monitor Taxonomy

- **In-situ monitors:**
  - In-situ time-to-digital converter (TDC) [Fick10]
  - In-situ path RO [Ngo10, Wang08]

- **Replica monitors:**
  - One monitor: representative path [Liu10]
  - Many monitors: PSRO [Bhushan06]

- How many monitors?
- How to design monitors?
- How to use monitors?
Key Observation: Sensitivities Cluster!

- Each dot represents $\Delta$delay of a critical path under variations
- The sensitivities form natural clusters
  - Design dependent
  - Multiple monitors
    - One monitor per cluster
DDRO Contributions

- Systematic methodology to design *multiple* DDROs based on clustering
- Systematic methodology to leverage monitors to estimate chip delay
Outline

• Performance Monitoring: An Introduction
• DDRO Implementation
  – Delay model
  – Sensitivity Clustering
  – DDRO Synthesis
• Delay Estimation from Measured DDRO Delays
• Experiment Results
• Conclusions
Delay Model and Model Verification

- Assume a linear delay model for variations

\[ d = d_{\text{nom}} (1 + \sum V_j G_j) \]

- Linear model correlates well with SPICE results

![Graph showing comparison between estimation from delay model and path delay from SPICE](image)

Max error ≈ 13ps
Sensitivities and Clustering

• Extract delay sensitivity based on finite difference method
  \[ V_j = \frac{d_{G_j=1\sigma} - d_{nom}}{d_{nom}} \]

• Cluster the critical paths based on sensitivities
  – Use kmeans++ algorithm
  – Choose best k-way clustering solution in 100 random starts
  – Each cluster centroid = target sensitivity for a DDRO

• Synthesize DDROs to meet target sensitivities
DDRO Synthesis

• Gate module is the basic building block of DDRO
  – Consists of standard cells from qualified library
• Multiple cells are concatenated in a gate module
  – Inner cells are less sensitive to input slews and output load variation
  – Delay sensitivity is independent of other modules
ILP formulation

- Module sensitivity is independent of its location

\[
\text{RO sensitivity} = \sum (S_h \times \text{Module } h \text{ sensitivity})
\]

- Module number can only be integers
- Formulate the synthesis problem as integer linear programming (ILP) problem

Minimize: \[\text{RO sensitivity} - \text{Target sensitivity}\]

Subject to:
\[
\text{Delay}_{\text{min}} < \sum (S_h \times \text{Module } 1 \text{ delay}) < \text{Delay}_{\text{max}}
\]
\[
\sum S_h < \text{Stage}_{\text{max}}
\]
Outline

• Performance Monitoring: An Introduction
• DDRO Implementation
• Delay Estimation from Measured DDRO Delays
  – Sensitivity Decomposition
  – Path Delay Estimation
  – Cluster Delay Estimation
• Experiment Results
• Conclusions
Sensitivity Decomposition

• Based on the cluster representing RO
• User linear decomposition to fully utilize all ROs

\[ \text{Sens(path)} = 0.9 \times \text{Sens(RO1)} + 0.1 \times \text{Sens(RO2)} \]
Path Delay Estimation

- Given DDRO delay, use the sensitivity decomposition
- Apply margin for estimation confidence

Predicted path delay: \( d_{i_{\text{path}}} = d_{\text{nom}} \times \left( 1 + \sum b_k \frac{d_{k_{\text{ro}}} - d_{k_{\text{nom}}}}{d_{k_{\text{nom}}}} + u_i \right) \)

- Measured from RO
- Margin

Other variation components
- Sensitivity residue

Where: \( u_i = l_{i_{\text{path}}} + V_{res} G \)

- One estimation per path
Cluster Delay Estimation

• For run-time delay estimation, may be impractical to make one prediction per path
• Reuse the clustering
  – Assume a pseudo-path for each cluster
    \[ d_{x}^{\text{cluster}} = \max \{ d_{i}^{\text{path}}, \text{path} \ i \in \text{cluster} \ X \} \]
  – Use statistical method to compute the nominal delay and delay sensitivity of the pseudo-path
  – Estimate the pseudo-path delay
• One estimation per cluster
Outline

• Introduction
• Implementation
• Delay Estimation
• **Experiment Results**
• Conclusion
Sensitivity Extraction

- All variability data from a commercial 45nm statistical SPICE model
Experiment Setup

• Use Monte-Carlo method to simulate critical path delays and DDRO delays
• Apply delay estimation methods with certain estimation confidence
  – 99% in all experiments
• Compare the amount of delay over-prediction
  – Delay from DDRO estimation vs. Delay from critical paths
Linear Model Results
Global variation only

- Overestimation reduces as the number of cluster and RO increases
- The two estimation methods perform similarly
Linear Model Results
Global and local variations

With local variation, the benefit of having more ROs saturates

Local variation can only be captured by in-situ monitors
Conclusion and Future Work

• A systematic method to design multiple DDROs based on clustering
• An efficient method to predict chip delay
• By using multiple DDROs, delay overestimation is reduced by up to 25% (from 4% to 3%)
  – Still limited by local variations
• Test chip tapeout using 45nm technology
  – With an ARM CORTEX M3 Processor
Acknowledgments

• Thanks to Professor Dennis Sylvester, Matt Fojtik, David Fick, and Daeyeon Kim from University of Michigan
Thank you!
Test Chip

• Test chip tapeout using 45nm technology
  – With an ARM CORTEX M3 Processor
Gate-module

• The delay sensitivities for different input slew and output load combinations.
• Use 5 stages as trade-off between module area and stability.
SPICE Results

Global and local variations

AES

M0

MIPS
Process Tuning

• Circuit performance monitoring is potentially helpful as test structure for manufacturing process tuning

- How to exploit the performance monitors to make short-loop monitoring?

T. Chan, ICCAD 2010
# Existing Monitors

<table>
<thead>
<tr>
<th></th>
<th>Generic</th>
<th>Design-dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Many monitors</td>
<td>N/A</td>
<td>Representative path [Xie10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In-situ monitors [Fick10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Critical-path replica [Black00, Shaik11]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In-situ path RO [Ngo10, Wang08]</td>
</tr>
<tr>
<td>Multiple monitors</td>
<td>PSRO [Bhushan06]</td>
<td>This work</td>
</tr>
<tr>
<td></td>
<td>RO [Tetelbaum09]</td>
<td>TRC [Drake08]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Process monitors [Burns08, Philling09]</td>
</tr>
<tr>
<td>One monitor</td>
<td>PLL [Kang10]</td>
<td>Representative Path [Liu10]</td>
</tr>
</tbody>
</table>