ReCycle: Pipeline Adaptation to Tolerate Process Variation

Abhishek Tiwari, Smruti R. Sarangi, Josep Torrellas
Outline

• Motivation
• ReCycle Idea
• Using ReCycle
• ReCycle System overview
• Results
Motivation

• Variation makes some stages slower than others
• Slowest stage determines clock period
• Wasted cycle time in fast stages
Existing Solutions

Circuit Level
• ABB and ASV
• Complicate manufacturing process
• Increase leakage
• Degrade lifetime reliability

Architecture Level
• VACA, Variable Latency Register Files
  – Target specific functional blocks
• X-Pipe, Razor
  – Redesigning latching elements
  – Substantial design effort and hardware overhead
Outline

• Motivation
• ReCycle Idea
• Using ReCycle
• ReCycle System overview
• Results
ReCycle Idea

• Transfer excess cycle time from fast stages to slower ones
• Clock the pipeline at the average stage delay
• Comprehensively apply to entire pipeline
  – Timing constraints
  – Pipeline loops
Effect of Process Variation

Pipeline Without Variation

Effect of Variation

Pipeline After Variation (No ReCycle)

Pipeline After Variation (With ReCycle)

Period

$T_i$ for all $i$

Max($T_i$)

Avg($T_i$)
Pipeline Stage Clock Skew

Clock to Initial Register

Clock to Final Register (No ReCycle)

Clock to Final Register (With ReCycle)

\[ D_{\text{min}} \]

\[ D_{\text{max}} \]

\[ T_{\text{setup}} \]

\[ T_{\text{hold}} \]

\[ T_{\text{skew}} \]
Timing Constraints

Setup constraint:
\[ \partial_i - \partial_f + T_{cp} \geq D_{\text{max}} + T_{\text{setup}} \]

Hold constraint
\[ \partial_f - \partial_i \leq D_{\text{min}} - T_{\text{hold}} \]

- Linear Program formulation
- Map to a graph
- Solve using shortest paths algorithm [Albrecht et al]
Determining $D_{\text{max}}$ and $D_{\text{min}}$

- **Before fabrication:** identify groups of slowest and fastest paths
- **After fabrication:** apply BIST vectors at a range of frequencies to these paths
Pipeline loops
Clock Distribution Network

![Diagram of Clock Distribution Network]

- Clock Gen.
- Clock Drivers
- Repeaters
- Signal Buffers
- Local Grid
- IntQ
Clock Distribution Network
Tunable Delay Buffers

![Diagram of Tunable Delay Buffers]
Outline

- Motivation
- ReCycle Idea
- Using ReCycle
- ReCycle System overview
- Results
Using ReCycle

- Enabling high-frequency, long pipelines
- Adding **Donor** stages
- Pushing slack to feedback paths
  - Power reduction
  - Improved reliability
- Salvaging chips rejected due to hold violations
Enabling High-Frequency Pipelines

- Intuitive analytical model:
  - All paths in a stage have same delay
  - Path delays across stages are uncorrelated
  - Stage delays are normally distributed \( \sim N(\mu, \sigma) \)
Pipeline Without ReCycle

• Cumulative distribution function of clock period $T_{CP}$

  $F_{nr_{CP}}(x) = P(T_{CP}\leq x) = P(T_1 \leq x \land \ldots \land T_N \leq x)$

  – Path delays are independent, therefore
    • $F_{nr_{CP}}(x) = P(T_1 \leq x) \ldots P(T_N \leq x)$
  – Let $F(x) = $ CDF of individual stage delay
    • $F_{nr_{CP}}(x) = (F(x))^N$
Pipeline With ReCycle

- \( F_{r_{CP}}(x) = P(T_{CP} \leq x) \)
  \[ = P\left(\frac{(T_1 + \ldots + T_N)}{N} \leq x\right) \]
  \[ = F(x) \]

- Thus, \( F^{nr}_{CP}(x) = \left(F_{r_{CP}}(x)\right)^N \)

  - Pipeline with ReCycle performs *exponentially* better as we add more stages (\(N\) increases)
Adding **Donor** Stages

- Empty stage added to the *critical* loop
- Generates additional slack that is *donated* to other stages in the loop
- Implementation:
  - Include a **Duplicate** register after the output registers of some pipeline stages
  - Duplicate registers are transparent in normal operation
  - Use pass-transistor multiplexing to enable and disable these latches
Donor Algorithm

• Identify critical loop using ReCycle algorithm
• Repeat for all duplicate registers in the loop:
  – Enable duplicate register
  – Measure IPC impact on workload
• Chose duplicate with highest performance
Pushing Slack to Feedback Paths

- Use ReCycle algorithm to squeeze out slack from logic datapaths and push it into feedback paths of each pipeline loop.

- Use this extra slack in feedback paths to:
  1. Eliminate repeaters: save power
  2. Improve routability: relax routing constraints
Overlapping Loops

Feedback paths of both loops 1 and 2 get slack of the shared stage $IntQ$
Salvaging Chips Rejected Due to Hold Violations

Hold constraint: \[ \partial_f - \partial_i \leq D_{\text{min}} - T_{\text{hold}} \]

- Process variation can speed up some critical paths such that \( D_{\text{min}} \leq T_{\text{hold}} \)
- Slowing down critical paths by reducing voltage degrades noise margins of slow critical paths
- ReCycle can adjust the latch skews \( \partial_f \) and \( \partial_i \) to meet the hold constraint
Outline

• Motivation
• ReCycle Idea
• Using ReCycle
• ReCycle System overview
• Results
Overall ReCycle System
Outline

• Motivation
• ReCycle Idea
• Using ReCycle
• ReCycle System overview
• Results
Results

• Timing Issues
• Frequency
• Adding Donor Stages
• Overall Performance and Power
• Eliminating Repeaters
Histogram of Critical Pipeline Loops

Influencing factors:

- Number of logical stages in the loop
- Number of physical stages in each logical stage
- Relative number of feedback paths
Average Slack Per Stage

All non-critical pipeline loops have unused slack
Skew Versus Correlation Length

Average skew is a measure of stage imbalance
Skew Versus Logic Depth

![Graph showing Skew Versus Logic Depth with bars for Max and Avg values.](image)
Results

• Timing Issues
• Frequency
• Adding Donor Stages
• Overall Performance and Power
• Eliminating Repeaters
Pipeline Frequency

![Pipeline Frequency Graph](image)

- Relative frequency
- 0.0, 0.4, 0.8, 1.2
- 0.1, 0.3, 0.5, 0.9

Legend:
- NoVar
- ReCycle
- Var
Effect of Scaling

ReCycle puts variation affected pipelines back on the roadmap to scaling
Results

• Timing Issues
• Frequency
• Adding Donor Stages
• Overall Performance and Power
• Eliminating Repeaters
Adding Donor Stages

![Graph showing the relationship between Extra Donor stages and Relative frequency, with data points for different categories like All applications, SPECint, SPECfp, and NoVar performance.](image)
Number of Donor Stages Needed

![Bar chart showing the number of donor stages needed for various benchmarks with error bars.](chart.png)
Results

• Timing Issues
• Frequency
• Adding Donor Stages
• Overall Performance and Power
• Eliminating Repeaters
Performance

![Bar chart showing relative BIPS for different conditions: Var, ReCycle, NoVar, ReCycle+ St Donor, ReCycle+ Dyn Donor. The x-axis represents the conditions, and the y-axis represents the relative BIPS ranging from 0 to 1.2.]
Dynamic Power

[Bar chart showing relative power for different conditions: Var, ReCycle, NoVar, ReCycle+St Donor, ReCycle+Dyn Donor]
Results

- Timing Issues
- Frequency
- Adding Donor Stages
- Overall Performance and Power
- Eliminating Repeaters
Fraction of Repeaters Eliminated

Receivers removed (%) vs. Useful logic per stage (FO4) for different values of $\phi$: 0.1, 0.3, 0.5, and 0.9.
Conclusions

- Increase pipeline freq without changing structure
- A comprehensive framework for dynamic pipeline adaptation using donor stages
- Low overhead