Timing Variation Detection, Control and Tolerance

Yinhe Han

Key Lab. Of System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences

2009-6-5
Outline

- Timing Variation
- Timing Variation Detection
- Timing Variation Control
- Timing Variation Tolerance
Timing Variation Sources

- Timing Variation ≠ Delay fault
- Process Variation
- Transistor Aging
  - Negative bias temperature instability (NBTI)
  - TDDB, Hot Carrier Injection, Temperature
- Soft Error
  - Single Event Transient (SET)
  - Single Event Upset (SEU)
Soft Error

- **SEU (Single Event Upset)**
  - Unintentional bit-flip in storage cells

- **SET (Single Event Transient)**
  - Transient voltage pulse propagating in combinational logics
Soft Error Detection

- Double-DFFs Detection
Aging Delay Detection

- Targeting Aging Delay
Unified detection scheme for Soft Error and Aging Delay

- Unified detection scheme \(\leq\) Unified fault model
- Stability Violation Check: Signal transitions occur in Stable Period.
How Soft Errors cause Stability Violation

ONLY the Stability Violation occurring in “vulnerable window” --- within which the flip-flops are updated --- could cause failures.
Stability Checker

- Basic operating principle
  - Step 1: Precharge S1 and S2 to “HIGH”
  - Step 2: Monitor state (evaluation)
    - No stability violation
      - $\overline{S_1 \text{ NOR } S_2} = 1$
    - Otherwise
      - $\overline{S_1 \text{ NOR } S_2} = 1$

When to precharge and evaluation is an essential design consideration!
Allocating precharge and evaluation time

CASE 1

Likely to catch normal tran- False Alarm!

CASE 2

Still NOT good!

CASE 3

1) is masked (logic or latch window)
2) cause SV--- Propagation Detectable
3) is stabilized before the start of Eval.

(n-1)T
Partitioning the detection phases

Still Open (XOR Protection)

Propagation Detectable Period

Benign Period And precharge can be scheduled here

Precharge → PDP → Evaluation
Comprehensive Solution

tpd: propagation delay of the combinational logic
tcd: contamination delay (a.k.a. short-path delay)
tcq: flip-flop’s clock-to-q time
T_{GB}: “conservative” setup time requirement
T_{DS}: expected maximum width of SET
Experiments

- Using 65nm PTM
- Hspice Simulation
- Overhead Analysis
  - Area
  - Power
  - Performance
  - Design complexity
Simulation Signal States

- **Guard Band**
- **Detection Slack**
- **Normal Transitions**
- **Fault Transitions**
- **Aging delay**
- **SEU fault**

- **Voltage**
  - CLK
  - CLKS
  - XOR
  - So
  - S1
  - S2
  - A1
  - B1
  - X

- **Time**
  - 0
  - 500p
  - 1n
  - 1.5n
  - 2n
  - 2.5n
  - 3n
  - 3.5n
  - 4n
Aging Delay Effect Control

- Input Vector Control
  - First applied in Leakage Power
  - Recently, in aging delay effect control by Yu Cao.

- Internal Node Control
  - Enhance the controllability of internal node

- Gate Size and Modification
  - Dr. Wang and Prof. Xie at DATE 2009
Multiple Input Vectors Control

- Using multiple input vectors, not only single vector to control aging delay
- Multiple vectors are applied in a uniform way during standby mode

![Diagram of control logic](image)
Multiple Vectors Combination

- Delay increase determined by duty cycle:

\[ \Delta T_{p(i)} = c_i \times \alpha_i^n \times t^n \]

\[ \Delta 1 + \Delta 2 + \Delta 3 + \Delta 4 + \Delta 5 + \Delta 6 \]
Timing-Critical Paths Selection

- Prepare: Using STA to get a candidate set \( D_{p(i)} \)
  \[
  D_{p(i)} \times (1 + 20\%) \geq D_{\text{max}} \times (1 + 10\%)
  \]

- First Step: Pruning TPs based on the rising or falling transition

CASE 1

CASE 2

Different Aging Delay
## Critical Gates Identification

### Second Step: Critical Gates Identification

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Num(CS)</th>
<th>Num(PR)</th>
<th>R%</th>
</tr>
</thead>
<tbody>
<tr>
<td>c880</td>
<td>80</td>
<td>18</td>
<td>77.5%</td>
</tr>
<tr>
<td>c1908</td>
<td>693</td>
<td>27</td>
<td>96.1%</td>
</tr>
<tr>
<td>c3540</td>
<td>1621</td>
<td>273</td>
<td>83.1%</td>
</tr>
<tr>
<td>c5315</td>
<td>1244</td>
<td>598</td>
<td>51.9%</td>
</tr>
<tr>
<td>s298</td>
<td>3</td>
<td>2</td>
<td>33.3%</td>
</tr>
<tr>
<td>s820</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>s1196</td>
<td>25</td>
<td>8</td>
<td>46.6%</td>
</tr>
<tr>
<td>s9234</td>
<td>2280</td>
<td>322</td>
<td>85.8%</td>
</tr>
</tbody>
</table>
Multi-Object Optimization Model

- **Formalizing the Optimization Object**

\[ D_{p(i)} = \phi_l + \sum_{j=1}^{l} \varphi_{j(k)} \times \alpha_{j(k)}^{n} \]

\[ \Delta T_{p(i)} = c_i \times \alpha_{i}^{n} \times t^{n} \]

Then:

\[ \min \max(D_{p(1)}, D_{p(2)}, \ldots, D_{p(m)}) \]
Two Contrains

- **Constraint-1: No Timing Violation Problem**
  \[ D_{p(i)} - D_{\text{max}} \times \gamma \leq 0 \]

- **Constraint-2: Logic Conflicts among Critical Gates**
  \[ 1 - (\alpha_a + \alpha_b) \leq \alpha_c \]
  \[ \alpha_c \leq 1 - \max(\alpha_a, \alpha_b) \]
  \[ \frac{\alpha_a + \alpha_b}{2} + \frac{\alpha_a - \alpha_b}{2} + \alpha_c \leq 1 \]
  \[ \frac{\alpha_a + \alpha_b}{2} + \frac{\alpha_b - \alpha_a}{2} + \alpha_c \leq 1 \]
Vectors Generation

- ATPG-like Process to Generate Input Vectors

![Diagram of ATPG-like process on C17]

N1 \[x|x|x\] → N10 [x|x|0] → N16 [0.7|0|1] → N19 [x|x|x] → N22 [x|x|x]

N2 [0|1|1] → N10 [x|x|0] → N16 [0.7|0|1] → N19 [x|x|x] → N23 [x|x|x]

N3 [0|1|1] → N11 [0.7|0|1] → N16 [0.7|0|1] → N19 [x|x|x] → N23 [x|x|x]

N6 [0.3|1|0] → N11 [0.7|0|1] → N19 [x|x|x] → N23 [x|x|x]

N7 [x|x|x] → N2 [0|1|1] → N10 [x|x|0] → N19 [x|x|x] → N23 [x|x|x]
# Experimental Results

<table>
<thead>
<tr>
<th>Circuit</th>
<th>M-IVC(proposed)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$D_{ori}$(ps)</td>
<td>$D_{final}$(ps)</td>
<td>+</td>
<td>$N_{iv}$</td>
</tr>
<tr>
<td>c880</td>
<td>1651</td>
<td>1683</td>
<td>2%</td>
<td>3</td>
</tr>
<tr>
<td>c1908</td>
<td>2201</td>
<td>2236</td>
<td>1.6%</td>
<td>3</td>
</tr>
<tr>
<td>c3540</td>
<td>3391</td>
<td>3588</td>
<td>5.8%</td>
<td>6</td>
</tr>
<tr>
<td>c5315</td>
<td>2962</td>
<td>3142</td>
<td>6%</td>
<td>5</td>
</tr>
<tr>
<td>s298</td>
<td>630</td>
<td>630</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>s820</td>
<td>812</td>
<td>812</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>s1196</td>
<td>2133</td>
<td>2186</td>
<td>2.6%</td>
<td>5</td>
</tr>
<tr>
<td>s1238</td>
<td>1730</td>
<td>1740</td>
<td>0.6%</td>
<td>7</td>
</tr>
<tr>
<td>s9234</td>
<td>3148</td>
<td>3202</td>
<td>5.7%</td>
<td>19</td>
</tr>
</tbody>
</table>
Tolerate Timing Variation

- Using Sensor to detect the timing variation

Diagram showing:
- Stability checker
- OR
- Output Latch
- Sensor 1
- Sensor 2
- Aging alarm
- Guard Band
- Upstream Flip-Flops
- Downstream Flip-Flops
- Combinational Logic
- Timing Non-Critical Signals
Path-Grain Dynamic Timing Adaptation

- Using Fine Timing Adaptation to tolerate the timing variation
A Case of timing unbalance

Analysis data from FPU of UltraSPARC T1 processor
Adaptive Clock Assignment

- UAFF
- FAFF
- BAFF
- GFF

CLK
FCLK
BCLK
FFs

CLK
BCLK
FCLK
Low Power Design using MicroFix
Power Reduction

Using 32nm high-performance models

Using 32nm low-power models

(a)

(b)

(c)

(d)