### Design Techniques for Diversity

**Subhasish Mitra**  
**RATS**  
10. 25. 99.  
Center for Reliable Computing  
Stanford University

#### Outline
- Motivation  
- Design diversity metric  
- Error latency  
- Two-level synthesis  
- Multi-level synthesis  
- Conclusions

#### Background
- Hardware redundancy techniques  
  - Duplex  
  - Two implementations  
  - Comparator at the output

#### Common-Mode Failures
- CMFs  
  - Affect multiple copies simultaneously  
  - Single cause  
- Diversity  
  - Antidote for CMFs

#### Design Diversity
- Independent design [Avizienis 85]  
- Software  
  - N-version programming  
- Hardware  
  - Function and its dual [Tamir 84]  
  - Three different processors  
  - Boeing 777  
- Evaluate diversity  
  - Diversity metric [Mitra 99]

#### D: Design Diversity Metric
- Two implementations \( N_1 \) and \( N_2 \)  
- Faults \( f_i \) in \( N_1 \), \( f_j \) in \( N_2 \)  
- \( d_{ij} \)  
  - Probability that \( N_1 \) and \( N_2 \) produce  
  - Correct outputs or  
  - Non-identical erroneous outputs  
  
\[
D = \sum_{(f_i, f_j)} P(f_i, f_j) d_{ij}
\]
Design Techniques for Diversity

### Error Latency
- Mission time $T$
- Faults $f_1$ in $N_1$, $f_2$ in $N_2$
- Error latency [Shedletsky 75]
  - $\#$ input vectors
  - $f_1$, $f_2$ active
  - Incorrect output observed
  
  $$\min[T, \frac{1}{1 - d_{1,2}}]$$

### Diversity Effects on CMFs
- CMFs in duplex systems
- Pick worst $f_2$
- Least error latency

### Simulation Results

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Modules</th>
<th>Error latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z3xp1</td>
<td>Identical</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Different</td>
<td>1711</td>
</tr>
<tr>
<td>clip</td>
<td>Identical</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Different</td>
<td>372</td>
</tr>
<tr>
<td>inc</td>
<td>Identical</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Different</td>
<td>1645</td>
</tr>
<tr>
<td>rd84</td>
<td>Identical</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Different</td>
<td>301</td>
</tr>
</tbody>
</table>

- Diversity helpful for CMFs

### Problem
- Given implementation $N_1$
- Synthesize implementation $N_2$
  - Worst case error latency maximized
- Synthesis steps
  - Two-level minimization
  - Multi-level transformations

### Conventional Synthesis Flow
- Two-level minimization
  - Minimize $\#$ cubes
  - Minimize $\#$ literals
- Multi-level transformations
  - Minimize literal count

### Definitions
- Fault equivalence
  - $f_1$ equivalent to $f_2$
  - Test set of $f_1 = \text{Test set of } f_2$
- Fault dominance
  - $f_1$ dominates $f_2$
  - Test set of $f_1 \supseteq \text{Test set of } f_2$
Design Techniques for Diversity

Two-level Single Output Circuit
- Any internal stuck-at fault
- Dominated by or equivalent to
  - Input or Output stuck-at fault

Example
- E/0 dominates A/0, B/0, C/0, D/0
- E/1 dominates A/1, B/1, C/1, D/1

Two-Level Minimization
- Single output function
- Any stuck-at fault \( f_1 \) in \( N_1 \)
- Worst case fault in \( N_2 \)
  - Output stuck-at fault
  - Test set determined by function
    - Same for all implementations
- Any implementation for \( N_2 \)
  - Same worst case error latency

Good News
- Previous observation
  - Not true for multi-output functions
  - Fanout structure matters
    - Multi-output prime implicants (MOPI) sharing
- Next target
  - Multi-output functions

Why Sharing Matters? Case 1

Why Sharing Matters? Case 2
- Fault detected on \( Z_2 \)
  - ‘1’ on \( Z_1 \) and \( Z_3 \)
Design Techniques for Diversity

What Do We Learn?
- Second case better than the first
  - Higher worst-case latency possible
- Create diversity in MOPI sharing
- Find the best MOPI sharing

Does Complementation Help?
- Fault detected on $Z_2$
  - '0' on $Z_1$ or '1' on $Z_1$ independent of $p$
  - Similar for $Z_3$

Effect of Complementation
- Fault detected on $Z_2$
  - '0' on $Z_1$ and $Z_3$
  - Computation easier
- Similar for stuck-at 1 on $p$ output

Issues
- Start with MOPIs
  - Want area-efficient circuits
  - Related to reliability
- Each output function
  - Prime and Irredundant?
    - Some MOPIs cannot be used

Two-Level Synthesis
- Form complemented truth table
- Generate MOPIs
- Create covering table
- Perform covering
  - Cost function
    - Minimize literal count
    - Maximize expected worst case latency

Covering Table Reduction
- For each output function
  - Remove dominating columns
  - No MOPI discarded
Design Techniques for Diversity

### Covering Table Reduction
- **Primary goal**
  - Minimum literal count
    - Dominated rows removed
    - Equivalent rows not removed
  - Maximize worst case error latency
    - Dominated rows not removed
    - Equivalent rows not removed

### Multi-Level Synthesis
- **General synthesis flow**
  - Start with 2-level design
  - Apply multi-level transformations

### Testability Preservation
- **Network $K_1$ to $K_2$**
- **$T$ detects all faults in $K_1$**
- **$T$ detects all faults in $K_2$**
- **Worst case error latency**
  - Cannot increase
  - May decrease

### Multi-Level Transformations
- **Single-cube extraction**
- **Double-cube extraction**
- **Kernel extraction**
- **Resubstitution**

[Rajski 92][De Micheli 94]

### Single-Cube Extraction
- $Z/0$ dominates $C/0$, $Z/1$ dominates $C/1$
- Worst case error latency unchanged
- Can use this transform freely

### Double-Cube Extraction
- $Z/0, Z/1$ dominates all faults
- Worst case error latency unchanged
- Can use freely
Kernel Extraction

- ABC'D + ABC'D' to (C'D + CD')AB
  - Worst case error latency unchanged
  - Can use freely

- ABC'D + ABC'D + A'CD
  - (C'D + CD')AB + A'CD
  - Fanout point
    - Worst case error latency may decrease
    - Cannot use freely

Resubstitution

- Boolean network
- Expressions for same function
  - Replace with a single node
  - Affects fanout structure

Resubstitution Effects

- Decreased testability
  - Reconvergent fanout of odd parity
  - Worst case error latency can increase

- Increased testability
  - Fanout point driving disjoint outputs
  - Worst case error latency can decrease

- Cost function during synthesis

Summary

- Synthesis for diversity
- Two-level minimization
  - Fanout structure in multi-output circuits
    - MOPI sharing

- Multi-level transformations
  - Can use most transforms freely
  - Exceptions
    - Kernel extraction at fanout points
    - Resubstitution

References