Automatic Configuration Generation for Application-Dependent Testing of FPGA Interconnects

Mehdi Baradaran Tahoori
Stanford CRC
November, 2002

Outline
- Problem Statement
- Theory
  - Single-term functions
  - Generalization to Entire FPGA
- Satisfiability
  - Background
  - SAT formulation for FPGA testing
- Results
- Complete Test set
- Results
- Summary

Problem Statement
- Test FPGA routing resources
  - For a specific mapped design
- Different from
  - Design verification
  - Manufacturing testing

Motivation
- Mapped designs not fully testable
  - No structural testing performed
    - No scan chains, BIST, test points, ...
  - FPGA users rely on manufacturing test
- Why routing resources?
  - 80% of FPGA transistors
  - Well-known techniques for logic blocks
    - [Abromovic][Renovell]

Applications
- Application-Specific FPGA
  - Weak parts working only for a specific design
    - Not good for all designs
    - Xilinx Easy-path
  - System-level testing
    - Mission-critical applications
    - Fault-tolerant applications

Previous Work
- Partitioning [Das & Touba]
  - Decomposition into fanout-free logic cones
    - One fanout branch at a time
    - Transparent logic + FF in every CLB
  - 2-18 test configurations
    - Depending on the size of circuit
    - Lose some bridging faults by partitioning
  - Xilinx Easy-path
    - One net at a time
    - Hundreds of configurations
    - Lose lots of bridging faults
Approach
- Keep configuration of routing resources
- Keep sequential behavior of design unchanged
  - Not adding or removing flip-flops
- Implement testable functions in logic blocks

Advantages
- Timing of original design preserved
  - Not changing critical paths
- At-speed testing
- No extra place-and-route for test configurations
- Fast reconfiguration time
  - Partial reconfiguration of only logic blocks
- Configuration storage overhead minimized
  - Storing only differences [Huang]
- No fault missed due to partitioning
- Works for all SRAM-based FPGAs and CPLDs

Outline
- Problem Statement
- Theory
  - Single-term functions
  - Generalization to Entire FPGA
- Satisfiability
  - Background
  - SAT formulation for FPGA testing
- Results
- Complete Test set
- Results
- Summary

Single-term Function
- Logic function with only one minterm or maxterm
- Activating Input:
  - input values for single minterm (maxterm)
- All sensitized faults are detectable
- Test vector = activating input
- \( \text{A1, B0, C1, D1, F0, A_{BF}, B_{BF}, C_{BF}, D_{BF}} \)

Example
- Network of single-term function with activating inputs
- All sensitized faults detectable
### Necessary Condition
- A combinational logic network \( N \)
- Test Vector \( V \)
- All sensitized (multiple) faults are detectable
  - For all net \( n \) with value \( r_n \)
  - \( \forall n \) stuck-at \( r_n \) detectable
- Then
  - All logic functions are single-term
  - Input of each function = activating input

### Proof
- Consider a function \( F \) in \( N \)
  - \( v = m \) Inputs of \( F \)
  - Faulty input \( v' \)
  - All multiple faults detectable
    - \( F(v) \neq F(v') \)
  - For all \( 2^m-1 \) possible values for \( v' \)
    - \( F \) is single-term
    - \( v = \) activating input of \( F \)

### Sequential Circuit
- Sequential network of single-term functions
  - Function \( F_i \) followed by flip-flop \( F \)
  - Preset value of \( F \)
  - Number of test cycles = Maximum sequential depth
  - Longest sequential path from PI to PO
  - Preset values of FFs in FPGAs
  - Independently programmable for each FF

### Stuck-at Faults
- First Configuration:
  - All LUTs implement AND
  - Preset values of FFs: 1
  - Input Vectors: 1
  - All stuck-at 0 faults detectable
- First Configuration:
  - All LUTs implement OR
  - Preset values of FFs: 0
  - Input Vectors: 0
  - All stuck-at 1 faults detectable

### Bridging Faults
- Fault list
- All pair-wise faults for inputs of each LUT
- \( N \)-input LUTs
- \( \log N \) configurations with single-term functions
- Activating inputs: Walsh-Rademacher codes

### Configuration Generation
- Backtracks
- Implication of fanout stem and branches
- NP-complete in general

---

**Mehdi Baradaran Tahoori**

Page 3

November 2002
Problem Definition

- For the first configuration
  - Partition all the nets into two groups
  - s.t for each LUT with N inputs
    - N/2 inputs in first group, N/2 in second group
  - Assign 0(1) to all nets in first group
  - Assign 1(0) to all nets in second group
  - For the next configurations (2 to \( \log_2 N \))
    - Repeat this procedure recursively
    - For each group obtained in the previous step
  - Implication in algorithm
  - Fanout stem and all branches in same group

Approaches

- Develop specific ATPG algorithm
  - Heuristics
- Convert problem to another domain
  - Use pre-existing solutions in domain
    - Graph model
    - Satisfiability (SAT)

Outline

- Problem Statement
- Theory
  - Single-term functions
  - Generalization to Entire FPGA
- Satisfiability
  - Background
  - SAT formulation for FPGA testing
  - Results
  - Complete Test set
  - Results
  - Summary

Satisfiability

- A boolean function \( \mathcal{F}(x_1, \ldots, x_n) \)
  - Typically in CNF (Product of Sums)
    - Each sum called clause
  - Find assignment to \( x_1, \ldots, x_n \) s.t. \( \mathcal{F}(x_1, \ldots, x_n) = 1 \)
  - Unsatisfiable \( \iff \forall x_1, \ldots, x_n, \mathcal{F}(x_1, \ldots, x_n) = 0 \)
- NP-complete
  - Other NP-complete problems reducible to SAT

Problem Solving by SAT

- Convert original problem to a SAT problem
  - Original problem \( \Rightarrow \) boolean function \( \mathcal{F} \)
- Use SAT-solver for \( \mathcal{F} \)
  - SAT
    - Variable assignment \( \Rightarrow \) original solution
    - UNSAT : no solution for original problem
  - Issues
    - Complexity of generation of \( \mathcal{F} \)
    - Size of \( \mathcal{F} \) in terms of size of original problem
    - Complexity of solving SAT
      - Number of clauses, variable dependencies...

SAT Formulation

- SAT function for one LUT
  - M inputs
  - Exactly M/2 of inputs 0, M/2 1
  - \( S_i^0 = 1 \iff \) exactly \( k \) out of \( n \) inputs are 1
  - \( S_i^0(X, \ldots, X) = \prod x_{i,k} = n \)
  - \( S_i^0(X, \ldots, X) = 0 \)
  - \( S_i^0(X, \ldots, X) + \sum_i S_i^0(X, \ldots, X) , 0 < k < n \)
- SAT formulation for each LUT
  - \( S_{M/2}^{M/2} = 1 \)
Example

- 4-input LUT
  \[ S(x_1, x_2, x_3, x_4) = (x_1 + x_2 + x_3) x_2 + x_2 + x_4) + (x_1 + x_2 + x_3) x_2 + x_2 + x_4) \]
- Possible answers
  \((x_1, x_2, x_3, x_4) = (0011), (0101), (1100), \ldots\)

Example

- One variable for each net
- SAT function:
  \[ \mathcal{F} = S(x_1, x_2, x_3, x_4) S(x_1, x_2, x_3, x_4) S(x_1, x_2, x_3, x_4) S(x_1, x_2, x_3, x_4) \]

Example (cont)

- Possible solution
  \(<x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}> = <0, 0, 1, 1, 0, 1, 1, 0, 0, 1>\)
- Two groups: \(G_1 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}, G_2 = \{x_8, x_9, \ldots\}\)

Test Configurations

- First Configuration
  - Each LUT
    \( \geq 2 \) inputs in \( G_1 \), \( 2 \) inputs \( G_2 \)
  - Assign \( 0, 1 \) to \( G_1 \), \( 1, 0 \) to \( G_2 \)
  - Activating inputs \( \Rightarrow \) LUTs configuration

- Second Configuration
  - Partition \( G_1 \) and \( G_2 \) recursively
  \( \geq G_{ij}, G_{ij}, G_{ij}, G_{ij} \)
  - Each LUT
    - Exactly one input in each \( G_j \)

Second Configuration

- SAT function for each LUT: \( S(x, y) = (x+y) \)
- \( G_1\) partitioning: \( \mathcal{F}_1 = S(x_1, x_2, x_3) S(x_4, x_5, x_6) S(x_7, x_8, x_9) S(x_{10}, x_{11}) \)
- \( G_2\) partitioning: \( \mathcal{F}_2 = S(x_1, x_2, x_3) S(x_4, x_5, x_6) S(x_7, x_8, x_9) S(x_{10}, x_{11}) \)
- Overall SAT function: \( \mathcal{F} = \mathcal{F}_1 \mathcal{F}_2 \)
  - Disjoint variable sets for \( \mathcal{F}_1 \) and \( \mathcal{F}_2 \)
Automatic Configuration Generation for Application-Dependent Testing of FPGA Interconnects

Mehdi Baradaran Tahooori
Outline
- Problem Statement
- Theory
  - Single-term functions
  - Generalization to Entire FPGA
- Satisfiability
  - Background
  - SAT formulation for FPGA testing
- Results
- Complete Test set
  - Results
  - Summary

Complete Test Set
- Fault list
  - All bridging faults for n nets
    \( \Rightarrow \) Number of faults = \( n(n-1)/2 \)
  - All opens and stuck-at faults
  - Conventional bus testing for M lines
    \( \Rightarrow \) Assumes full controllability and observability
  - Example: 6 lines, 3 vectors

Complete Test Set (cont)
- Single-term functions
  - Offer full controllability and observability
- Consider one partition \( |P_i| = n \)
- \( \log_2(n+2) \) test configurations
- Bus test vectors \( \Rightarrow \) LUT test configurations
- Cover all bridging faults, opens, stuck-at

Example (cont)
- 12 nets, 4 configurations

<table>
<thead>
<tr>
<th>Circuit</th>
<th>CLBs</th>
<th>Nets</th>
<th>Faults</th>
<th>Configs</th>
</tr>
</thead>
<tbody>
<tr>
<td>term1</td>
<td>50</td>
<td>88</td>
<td>3082</td>
<td>7</td>
</tr>
<tr>
<td>9symmi</td>
<td>110</td>
<td>79</td>
<td>3081</td>
<td>7</td>
</tr>
<tr>
<td>apex7</td>
<td>120</td>
<td>115</td>
<td>6555</td>
<td>7</td>
</tr>
<tr>
<td>busc</td>
<td>156</td>
<td>151</td>
<td>11325</td>
<td>8</td>
</tr>
<tr>
<td>exm2</td>
<td>188</td>
<td>205</td>
<td>20910</td>
<td>8</td>
</tr>
<tr>
<td>alu2</td>
<td>195</td>
<td>153</td>
<td>11628</td>
<td>8</td>
</tr>
<tr>
<td>2large</td>
<td>196</td>
<td>186</td>
<td>17205</td>
<td>8</td>
</tr>
<tr>
<td>vda</td>
<td>272</td>
<td>225</td>
<td>25200</td>
<td>8</td>
</tr>
<tr>
<td>dma</td>
<td>288</td>
<td>213</td>
<td>22578</td>
<td>8</td>
</tr>
<tr>
<td>alu4</td>
<td>323</td>
<td>255</td>
<td>32385</td>
<td>9</td>
</tr>
<tr>
<td>k2</td>
<td>440</td>
<td>404</td>
<td>81406</td>
<td>9</td>
</tr>
<tr>
<td>bnre</td>
<td>462</td>
<td>352</td>
<td>67776</td>
<td>9</td>
</tr>
<tr>
<td>dfsm</td>
<td>506</td>
<td>420</td>
<td>87990</td>
<td>9</td>
</tr>
<tr>
<td>z03</td>
<td>702</td>
<td>608</td>
<td>184528</td>
<td>10</td>
</tr>
</tbody>
</table>
### Real FPGAs
- FPGA with $M$ LUTs
  - Maximum number of nets $n_{max} = 5 \times M$
    - One net for each input (4) and output (1)
    - For any mapped design
  - Upper bound on number of configurations
    - $\lceil \log_2 (5M+2) \rceil$
    - Complete test set

### Upper Bounds
- Xilinx Virtex II FPGAs

<table>
<thead>
<tr>
<th>Device Name</th>
<th>LUTs</th>
<th>Max Nets</th>
<th>Faults</th>
<th>Max Configs</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2V40</td>
<td>516</td>
<td>2580</td>
<td>3.3 M</td>
<td>12</td>
</tr>
<tr>
<td>XC2V88</td>
<td>1024</td>
<td>5120</td>
<td>13.1 M</td>
<td>13</td>
</tr>
<tr>
<td>XC2V250</td>
<td>3072</td>
<td>15360</td>
<td>118 M</td>
<td>14</td>
</tr>
<tr>
<td>XC2V500</td>
<td>6144</td>
<td>30720</td>
<td>472 M</td>
<td>15</td>
</tr>
<tr>
<td>XC2V1000</td>
<td>10240</td>
<td>51200</td>
<td>1316 M</td>
<td>16</td>
</tr>
<tr>
<td>XC2V1500</td>
<td>15360</td>
<td>76800</td>
<td>2949 M</td>
<td>17</td>
</tr>
<tr>
<td>XC2V2000</td>
<td>21504</td>
<td>107520</td>
<td>5780 M</td>
<td>17</td>
</tr>
<tr>
<td>XC2V3000</td>
<td>28672</td>
<td>143000</td>
<td>10276 M</td>
<td></td>
</tr>
<tr>
<td>XC2V4000</td>
<td>46080</td>
<td>230400</td>
<td>28542 M</td>
<td></td>
</tr>
<tr>
<td>XC2V6000</td>
<td>67584</td>
<td>337920</td>
<td>57095 M</td>
<td></td>
</tr>
<tr>
<td>XC2V8000</td>
<td>93184</td>
<td>465920</td>
<td>108540 M</td>
<td></td>
</tr>
</tbody>
</table>

### Summary
- Single-term functions
  - FPGA interconnect testing
- SAT formulation
  - Systematic conversion of FPGA testing into SAT
    - Testing for bridging fault
  - Experimental results on various benchmarks
    - Very fast
- Complete test set
  - All possible bridging, open and stuck-at faults
  - Less than 20 configurations for largest FPGA

### Conclusions
- Single-term functions
  - Offer full controllability and observability
  - Best for FPGA testing
    - Better than Transparent logic
- SAT formulation completely practical
  - Very few configurations, very fast
- Complete testing using single-term functions
  - Practical

### References