Reading Assignment

- This lecture: 5.1, 5.2
- Next lecture: 5.2, 5.3
Outline

Applications of Fault Simulation

General Fault Simulation Techniques
Fault Simulation

- Simulate a circuit in the presence of faults.
- Discover faults detected by a given test $T$ (sets of test vectors).
Fault Simulation

- Simulate a circuit in the presence of faults.
- Discover faults detected by a given test $T$ (sets of test vectors).
Fault simulation helps to determine the \textit{fault coverage} of $T$.
- The ratio of the number of faults detected by $T$ to the total number of simulated faults.

Fault coverage only refers to a particular fault model.
- 100% coverage may still fail to detect faults outside of the universe of the fault model.
- Experience shows that a test with high coverage for SSFs also achieves a high \textit{defect coverage}, i.e. with respect to the universe of physical faults.

A proper fault coverage is necessary to achieve a specific \textit{defect level} – probability of shipping a bad product.
Test Evaluation

- Fault simulation helps to determine the fault coverage of $T$.
  - The ratio of the number of faults detected by $T$ to the total number of simulated faults.
- Fault coverage only refers to a particular fault model.
  - 100% coverage may still fail to detect faults outside of the universe of the fault model.
  - Experience shows that a test with high coverage for SSFs also achieves a high defect coverage, i.e. with respect to the universe of physical faults.
- A proper fault coverage is necessary to achieve a specific defect level – probability of shipping a bad product.
Fault simulation helps to determine the *fault coverage* of $T$.
- The ratio of the number of faults detected by $T$ to the total number of simulated faults.

Fault coverage only refers to a particular fault model.
- 100% coverage may still fail to detect faults outside of the universe of the fault model.
- Experience shows that a test with high coverage for SSFs also achieves a high *defect coverage*, i.e. with respect to the universe of physical faults.

A proper fault coverage is necessary to achieve a specific *defect level* – probability of shipping a bad product.
Assume there are $n$ possible faults that each may happen independently with a probability $p$.

- Fraction of good products: the yield $Y = (1 - p)^n$.

Assume a fault coverage of $d$, i.e. $dn$ faults will be detected.

- Probability of accepting a product: $(1 - p)^{dn} = Y^d$.

Defect Level $= \frac{\text{Accepted bad products}}{\text{Accepted products}} = \frac{Y^d - Y}{Y^d} = 1 - Y^{1-d}$

(Fig. 5.1, Abramovici et al., 1990)
Defect Level

Assume there are $n$ possible faults that each may happen independently with a probability $p$.

- Fraction of good products: the yield $Y = (1 - p)^n$.

Assume a fault coverage of $d$, i.e. $dn$ faults will be detected.

- Probability of accepting a product: $(1 - p)^{dn} = Y^d$.

Defect Level = \frac{\text{Accepted bad products}}{\text{Accepted products}} = \frac{Y^d - Y}{Y^d} = 1 - Y^{1-d}
Defect Level

Assume there are \( n \) possible faults that each may happen independently with a probability \( p \).

- Fraction of good products: the yield \( Y = (1 - p)^n \).
- Assume a fault coverage of \( d \), i.e. \( dn \) faults will be detected.
  - Probability of accepting a product: \( (1 - p)^{dn} = Y^d \).
- Defect Level = \( \frac{\text{Accepted bad products}}{\text{Accepted products}} = \frac{Y^d - Y}{Y^d} = 1 - Y^{1-d} \)
Different test generation algorithms differ in how they modify $T$ if the coverage is not sufficient.
A test vector is introduced per iteration to cover a fault not detected previously.

Fault simulation helps to discover faults detectable under the test vector.

Figure 5.3

Fault simulation used in the selection of target faults for test generation (Abramovici et al., 1990)
A test vector is introduced per iteration to cover a fault not detected previously.

Fault simulation helps to discover faults detectable under the test vector.
Fault Location

- Before testing, fault simulation can be used to construct fault dictionaries.
  - Store the output response corresponding to every simulated fault $f$.
  - Help to locate the fault using the actual response.
- For storage efficiency, only the signature (hashing) of the precomputed output responses are stored.
  - There may exist multiple signatures matching the actual response.
- Post-test diagnosis first isolates faults consistent with the actual response and then simulates only them to identify the actual fault.
Fault Location

- Before testing, fault simulation can be used to construct *fault dictionaries*.
  - Store the output response corresponding to every simulated fault \( f \).
  - Help to locate the fault using the actual response.
- For storage efficiency, only the signature (hashing) of the precomputed output responses are stored.
  - There may exist multiple signatures matching the actual response.
- *Post-test diagnosis* first isolates faults consistent with the actual response and then simulates only them to identify the actual fault.
Fault Location

- Before testing, fault simulation can be used to construct *fault dictionaries*.
  - Store the output response corresponding to every simulated fault $f$.
  - Help to locate the fault using the actual response.
- For storage efficiency, only the signature (hashing) of the precomputed output responses are stored.
  - There may exist multiple signatures matching the actual response.
- *Post-test diagnosis* first isolates faults consistent with the actual response and then simulates only them to identify the actual fault.
Use fault simulation to simulate system behavior with presence of faults.

Critical for high-reliability systems when faults may drastically affect the operation.

- A fault can induce races and hazards not present in the fault-free circuit.
- A faulty circuit may oscillate or enter a deadlock (hang-up) state.
- A fault can inhibit the proper initialization of a sequential circuit.
- A fault can transform a combinational circuit into a sequential one or a synchronous circuit into an asynchronous one.
Reliable System Simulation

- Use fault simulation to simulate system behavior with presence of faults.
- Critical for high-reliability systems when faults may drastically affect the operation.
  - A fault can induce races and hazards not present in the fault-free circuit.
  - A faulty circuit may oscillate or enter a deadlock (hang-up) state.
  - A fault can inhibit the proper initialization of a sequential circuit.
  - A fault can transform a combinational circuit into a sequential one or a synchronous circuit into an asynchronous one.
Outline

Applications of Fault Simulation

General Fault Simulation Techniques
General Fault Simulation Techniques

- Serial fault simulation
  - Construct a model $N_f$ for the circuit $N$ with fault $f$.
  - Simulate $N_f$ using any logic simulator.
  - Simple, able to handle any kind of faults, but not efficient.

- Improvements
  - Simulate WITHOUT constructing $N_f$ explicitly.
  - Simulate a set of faults simultaneously.
  - e.g. parallel, deductive, and concurrent fault simulation.
General Fault Simulation Techniques

- **Serial fault simulation**
  - Construct a model $N_f$ for the circuit $N$ with fault $f$.
  - Simulate $N_f$ using any logic simulator.
  - Simple, able to handle any kind of faults, but not efficient.

- **Improvements**
  - Simulate WITHOUT constructing $N_f$ explicitly.
  - Simulate a set of faults simultaneously.
  - e.g. parallel, deductive, and concurrent fault simulation.
Parallel Fault Simulation

- **Inputs**
  - The good circuit $N$.
  - A set of $F$ faults to be simulated.
  - A test vector.

- **Outputs**
  - 1-bit per fault indicating whether the test vector detects it.
  - Idea: exploit bit-wise instructions on host.
    - For a word size of $W$ bits, simulate the good circuit and $W - 1$ bad circuits together.
    - Compare the outputs when simulation completes.
    - Need to run logic simulation $\lceil \frac{F}{W-1} \rceil$ times.
  - Challenges: how to model faults in bit-wise instructions?
Parallel Fault Simulation

- **Inputs**
  - The good circuit $N$.
  - A set of $F$ faults to be simulated.
  - A test vector.

- **Outputs**
  - 1-bit per fault indicating whether the test vector detects it.

- **Idea:** exploit bit-wise instructions on host.
  - For a word size of $W$ bits, simulate the good circuit and $W - 1$ bad circuits together.
  - Compare the outputs when simulation completes.
  - Need to run logic simulation $\lceil \frac{F}{W-1} \rceil$ times.

- **Challenges:** how to model faults in bit-wise instructions?
Parallel Fault Simulation

- **Inputs**
  - The good circuit $N$.
  - A set of $F$ faults to be simulated.
  - A test vector.

- **Outputs**
  - 1-bit per fault indicating whether the test vector detects it.

- **Idea:** exploit bit-wise instructions on host.
  - For a word size of $W$ bits, simulate the good circuit and $W - 1$ bad circuits together.
  - Compare the outputs when simulation completes.
  - Need to run logic simulation $\lceil \frac{F}{W-1} \rceil$ times.

- **Challenges:** how to model faults in bit-wise instructions?
Example

Consider three lines $A$, $B$, $O$ where $O = \overline{AB}$ (AND).

Assume we are simulating with a word size of 4 bits.
- Each bit corresponding to the value at the line when a given fault (or no fault) is presented.
- bit 0: good circuit
- bit 1: bad circuit with some fault leading to $A$
- bit 2: bad circuit with some fault leading to $B$
- bit 3: bad circuit with $O$ s-a-0

Assume $A = 1101$, $B = 1011$.
- We should have $O = 0001$.

$O = AB\overline{I}_O + S_O I_O$ (all operations are bit-wise)
- $I_O = 1000$: indicate whether we have the given fault at $O$.
- $S_O = 0xxx$: indicate the stuck-at value for the fault. Note that I put $x$ there to highlight the actual fault at $O$. For simulation, you should replace it with either 0 or 1.

Same method applies when propagating a signal from a stem to branches.
Consider three lines $A$, $B$, $O$ where $O = AB$ (AND).

Assume we are simulating with a word size of 4 bits.

- Each bit corresponding to the value at the line when a given fault (or no fault) is presented.
  - bit 0: good circuit
  - bit 1: bad circuit with some fault leading to $A$
  - bit 2: bad circuit with some fault leading to $B$
  - bit 3: bad circuit with $O$ s-a-0

Assume $A = 1101$, $B = 1011$.

We should have $O = 0001$.

$$O = A \overline{B} I_O + S_O l_O$$ (all operations are bit-wise)

- $l_O = 1000$: indicate whether we have the given fault at $O$.
- $S_O = 0xxx$: indicate the stuck-at value for the fault. Note that I put $x$ there to highlight the actual fault at $O$. For simulation, you should replace it with either 0 or 1.

Same method applies when propagating a signal from a stem to branches.
Example

- Consider three lines $A$, $B$, $O$ where $O = AB$ (AND).
- Assume we are simulating with a word size of 4 bits.
  - Each bit corresponding to the value at the line when a given fault (or no fault) is presented.
    - bit 0: good circuit
    - bit 1: bad circuit with some fault leading to $A$
    - bit 2: bad circuit with some fault leading to $B$
    - bit 3: bad circuit with $O$ s-a-0
- Assume $A = 1101$, $B = 1011$.
  - We should have $O = 0001$.
- $O = AB\overline{I_O} + S_O I_O$ (all operations are bit-wise)
  - $I_O = 1000$: indicate whether we have the given fault at $O$.
  - $S_O = 0xxx$: indicate the stuck-at value for the fault. Note that I put $x$ there to highlight the actual fault at $O$. For simulation, you should replace it with either 0 or 1.
- Same method applies when propagating a signal from a stem to branches.
Example

- Consider three lines $A$, $B$, $O$ where $O = AB$ (AND).
- Assume we are simulating with a word size of 4 bits.
  - Each bit corresponding to the value at the line when a given fault (or no fault) is presented.
  - bit 0: good circuit
  - bit 1: bad circuit with some fault leading to $A$
  - bit 2: bad circuit with some fault leading to $B$
  - bit 3: bad circuit with $O$ s-a-0
- Assume $A = 1101$, $B = 1011$.
  - We should have $O = 0001$.
- $O = AB\overline{I_O} + S_O l_O$ (all operations are bit-wise)
  - $l_O = 1000$: indicate whether we have the given fault at $O$.
  - $S_O = 0xxx$: indicate the stuck-at value for the fault. Note that I put $x$ there to highlight the actual fault at $O$. For simulation, you should replace it with either 0 or 1.
- Same method applies when propagating a signal from a stem to branches.
Example

- Consider three lines $A$, $B$, $O$ where $O = AB$ (AND).
- Assume we are simulating with a word size of 4 bits.
  - Each bit corresponding to the value at the line when a given fault (or no fault) is presented.
  - bit 0: good circuit
  - bit 1: bad circuit with some fault leading to $A$
  - bit 2: bad circuit with some fault leading to $B$
  - bit 3: bad circuit with $O$ s-a-0
- Assume $A = 1101$, $B = 1011$.
  - We should have $O = 0001$.
- $O = AB\overline{I_0} + S_OI_O$ (all operations are bit-wise)
  - $I_O = 1000$: indicate whether we have the given fault at $O$.
  - $S_O = 0xxx$: indicate the stuck-at value for the fault. Note that I put $x$ there to highlight the actual fault at $O$. For simulation, you should replace it with either 0 or 1.
- Same method applies when propagating a signal from a stem to branches.
Data Structure

▶ An array containing a word (e.g. $W = 32$) for each line.

▶ An array of $W$ elements indicating fault positions.
  
  - $I_Z$ is generated from this array when the value on line $Z$ is computed during simulation.
  
  - $S_Z$ could be part of this array and is reused for every line.

(Fig. 5.4, 5.5, Abramovici et al., 1990)
An array containing a word (e.g. \( W = 32 \)) for each line.

An array of \( W \) elements indicating fault positions.

- \( I_Z \) is generated from this array when the value on line \( Z \) is computed during simulation.
- \( S_Z \) could be part of this array and is reused for every line.
Discussions for Parallel Fault Simulation

- Three-valued parallel fault simulation is possible by using two words per line.
- Cannot simulate circuits directly when non-Boolean operations are involved.
- Not efficient for multivalued logic.
- Cannot take full advantage of event-driven simulations.
Discussions for Parallel Fault Simulation

- Three-valued parallel fault simulation is possible by using two words per line.
- Cannot simulate circuits directly when non-Boolean operations are involved.
  - Not efficient for multivalued logic.
  - Cannot take full advantage of event-driven simulations.
Discussions for Parallel Fault Simulation

- Three-valued parallel fault simulation is possible by using two words per line.
- Cannot simulate circuits directly when non-Boolean operations are involved.
- Not efficient for multivalued logic.
- Cannot take full advantage of event-driven simulations.
Discussions for Parallel Fault Simulation

- Three-valued parallel fault simulation is possible by using two words per line.
- Cannot simulate circuits directly when non-Boolean operations are involved.
- Not efficient for multivalued logic.
- Cannot take full advantage of event-driven simulations.
Fault simulation has numerous applications when one need to deal with faults in a system.

Efficient fault simulation algorithms avoid to construct the faulty circuits explicitly and attempt to simulate many faults simultaneously.