Literature Survey

- Choose one research article.
- Read it and its references.
  - Possibly references of references.
- Explain the work there.
- You may work with other students to read the same article.
  - But all the writings should be your own.
- Progress Report Due: 10:00am 3/27 Chicago time
- Final Survey Paper Due: 10:00am 5/01 Chicago time
- Check course webpage for details.
Literature Survey

▶ Choose one research article.
▶ Read it and its references.
  ▶ Possibly references of references.
▶ Explain the work there.
▶ You may work with other students to read the same article.
  ▶ But all the writings should be your own.
▶ Progress Report Due: 10:00am 3/27 Chicago time
▶ Final Survey Paper Due: 10:00am 5/01 Chicago time
▶ Check course webpage for details.
Choose one research article.
Read it and its references.
   Possibly references of references.
Explain the work there.
You may work with other students to read the same article.
   But all the writings should be your own.
Progress Report Due: 10:00am 3/27 Chicago time
Final Survey Paper Due: 10:00am 5/01 Chicago time
Check course webpage for details.
Choose one research article.
Read it and its references.
  Possibly references of references.
Explain the work there.
You may work with other students to read the same article.
  But all the writings should be your own.
Progress Report Due: 10:00am 3/27 Chicago time
Final Survey Paper Due: 10:00am 5/01 Chicago time
Check course webpage for details.
Literature Survey

- Choose one research article.
- Read it and its references.
  - Possibly references of references.
- Explain the work there.
- You may work with other students to read the same article.
  - But all the writings should be your own.
- Progress Report Due: 10:00am 3/27 Chicago time
- Final Survey Paper Due: 10:00am 5/01 Chicago time
- Check course webpage for details.
Reading Assignment

- This lecture: 4.2 – 4.4
- Next lecture: 4.5, 4.6
Outline

Fault Equivalence and Fault Dominance

Fault Modeling for Sequential Circuits
Fault Equivalence

- Two faults are *functionally equivalent* if the logic functions of the system when they present are the same.
  - \( f \) and \( g \) are functionally equivalent iff \( \forall x, Z_f(x) = Z_g(x) \).

- A test *distinguishes* between two faults if the outputs under them are different.
  - \( t \) distinguishes \( f \) and \( g \) iff \( Z_f(x) \neq Z_g(x) \).
  - Tests distinguishing \( f \) and \( g \) are the solutions to \( Z_f(x) \oplus Z_g(x) = 1 \).

- There is no test that can distinguish between two functionally equivalent faults.

- It is sufficient to consider only one fault from each group of functionally equivalent faults for fault analysis.
  - Reduce the number of faults one has to consider.
Fault Equivalence

- Two faults are *functionally equivalent* if the logic functions of the system when they present are the same.
  - $f$ and $g$ are functionally equivalent iff $\forall x, Z_f(x) = Z_g(x)$.
- A test *distinguishes* between two faults if the outputs under them are different.
  - $t$ distinguishes $f$ and $g$ iff $Z_f(x) \neq Z_g(x)$.
  - Tests distinguishing $f$ and $g$ are the solutions to $Z_f(x) \oplus Z_g(x) = 1$.
- There is no test that can distinguish between two functionally equivalent faults.
- It is sufficient to consider only one fault from each group of functionally equivalent faults for fault analysis.
  - Reduce the number of faults one has to consider.
Fault Equivalence

- Two faults are *functionally equivalent* if the logic functions of the system when they present are the same.
  - \( f \) and \( g \) are functionally equivalent iff \( \forall x, \ Z_f(x) = Z_g(x) \).
- A test *distinguishes* between two faults if the outputs under them are different.
  - \( t \) distinguishes \( f \) and \( g \) iff \( Z_f(x) \neq Z_g(x) \).
  - Tests distinguishing \( f \) and \( g \) are the solutions to \( Z_f(x) \oplus Z_g(x) = 1 \).
- There is no test that can distinguish between two functionally equivalent faults.
  - It is sufficient to consider only one fault from each group of functionally equivalent faults for fault analysis.
    - Reduce the number of faults one has to consider.
Two faults are *functionally equivalent* if the logic functions of the system when they present are the same.

\[ f \text{ and } g \text{ are functionally equivalent iff } \forall x, Z_f(x) = Z_g(x). \]

A test *distinguishes* between two faults if the outputs under them are different.

\[ t \text{ distinguishes } f \text{ and } g \text{ iff } Z_f(x) \neq Z_g(x). \]

Tests distinguishing \( f \) and \( g \) are the solutions to \( Z_f(x) \oplus Z_g(x) = 1 \).

There is no test that can distinguish between two functionally equivalent faults.

It is sufficient to consider only one fault from each group of functionally equivalent faults for fault analysis.

\[ \text{Reduce the number of faults one has to consider.} \]
Consider an $n$-input AND gate.

- There are $2n + 2$ single stuck faults.
  - $2n$ at inputs and 2 at output.
- The s-a-0 faults at inputs are equivalent to the output s-a-0 fault.
- So we only need to consider $n + 2$ single stuck faults.
  - More than 33% reduction.
Consider an $n$-input AND gate.

- There are $2n + 2$ single stuck faults.
  - $2n$ at inputs and 2 at output.

- The s-a-0 faults at inputs are equivalent to the output s-a-0 fault.

- So we only need to consider $n + 2$ single stuck faults.
  - More than 33% reduction.
Consider an $n$-input AND gate.

- There are $2n + 2$ single stuck faults.
  - $2n$ at inputs and 2 at output.
- The s-a-0 faults at inputs are equivalent to the output s-a-0 fault.
- So we only need to consider $n + 2$ single stuck faults.
  - More than 33% reduction.
Consider an $n$-input AND gate.

There are $2n + 2$ single stuck faults.

- $2n$ at inputs and 2 at output.

The s-a-0 faults at inputs are equivalent to the output s-a-0 fault.

So we only need to consider $n + 2$ single stuck faults.

- More than 33% reduction.
More on Equivalence Fault Collapsing

(Fig. 4.12, Abramovici et al., 1990)

- Equivalence fault collapsing also applies to OR/NOR/NAND gates.
  - Black dots: s-a-1
  - White dots: s-a-0
Fault Location

- Goal of testing may include fault location in addition to fault detection.
  - Need to distinguish among detectable faults as much as possible.
- A complete location test set distinguishes between every pair of distinguishable faults.
- Fault detection is a special case of fault location.
  - Denote the fault-free circuit with the empty fault $\Phi$.
  - All undetectable faults are functionally equivalent to $Z_{\Phi}$.
  - A complete location test set must be able to distinguish between detectable and undetectable faults.
- In practice, it is usually computationally prohibitive to obtain a complete location test set.
  - We are interested in functionally equivalence under a test set $T$, i.e. we treat $f$ and $g$ equivalent iff $Z_f(t) = Z_g(t) \forall t \in T$. 
Fault Location

- Goal of testing may include fault location in addition to fault detection.
  - Need to distinguish among detectable faults as much as possible.

- A complete location test set distinguishes between every pair of distinguishable faults.

- Fault detection is a special case of fault location.
  - Denote the fault-free circuit with the empty fault $\Phi$.
  - All undetectable faults are functionally equivalent to $Z_\Phi$.
  - A complete location test set must be able to distinguish between detectable and undetectable faults.

- In practice, it is usually computationally prohibitive to obtain a complete location test set.
  - We are interested in functionally equivalence under a test set $T$, i.e. we treat $f$ and $g$ equivalent iff $Z_f(t) = Z_g(t) \forall t \in T$. 
Fault Location

- Goal of testing may include fault location in addition to fault detection.
  - Need to distinguish among detectable faults as much as possible.
- A complete location test set distinguishes between every pair of distinguishable faults.
- Fault detection is a special case of fault location.
  - Denote the fault-free circuit with the empty fault $\Phi$.
  - All undetectable faults are functionally equivalent to $Z_{\Phi}$.
  - A complete location test set must be able to distinguish between detectable and undetectable faults.
- In practice, it is usually computationally prohibitive to obtain a complete location test set.
  - We are interested in functionally equivalence under a test set $T$, i.e. we treat $f$ and $g$ equivalent iff $Z_f(t) = Z_g(t) \forall t \in T$. 

ECE 586 – Fault Detection in Digital Circuits
Spring 2015
Fault Location

- Goal of testing may include fault location in addition to fault detection.
  - Need to distinguish among detectable faults as much as possible.
- A complete location test set distinguishes between every pair of distinguishable faults.
- Fault detection is a special case of fault location.
  - Denote the fault-free circuit with the empty fault $\Phi$.
  - All undetectable faults are functionally equivalent to $Z_\Phi$.
  - A complete location test set must be able to distinguish between detectable and undetectable faults.
- In practice, it is usually computationally prohibitive to obtain a complete location test set.
  - We are interested in functionally equivalence under a test set $T$, i.e. we treat $f$ and $g$ equivalent iff $Z_f(t) = Z_g(t) \forall t \in T$. 

\[ Z_f(t) = Z_g(t) \forall t \in T \]
Consider two faults $g$ and $f$.

We say $f$ dominates $g$ iff for any test $t$ detecting $g$, $Z_g(t) = Z_f(t)$.

- So any test detecting $g$ can be used to detect $f$.
- It is not necessarily true any test detecting $f$ can be used to detect $g$.

Let $T_g$ and $T_f$ be all the tests detecting $g$ and $f$ respectively.

- $f$ dominates $g$ iff they are functionally equivalence under $T_g$.
- It implies $T_g \subseteq T_f$.

However, $T_g \subseteq T_f$ does not imply $f$ dominates $g$.

- In other words, even any test detecting $g$ can detect $f$, one cannot say $f$ dominates $g$. 
Fault Dominance

- Consider two faults $g$ and $f$.
- We say $f$ dominates $g$ iff for any test $t$ detecting $g$, $Z_g(t) = Z_f(t)$.
  - So any test detecting $g$ can be used to detect $f$.
  - It is not necessarily true any test detecting $f$ can be used to detect $g$.
- Let $T_g$ and $T_f$ be all the tests detecting $g$ and $f$ respectively.
  - $f$ dominates $g$ iff they are functionally equivalence under $T_g$.
  - It implies $T_g \subseteq T_f$.
- However, $T_g \subseteq T_f$ does not imply $f$ dominates $g$.
  - In other words, even any test detecting $g$ can detect $f$, one cannot say $f$ dominates $g$. 

ECE 586 – Fault Detection in Digital Circuits
Consider two faults \( g \) and \( f \).

We say \( f \) dominates \( g \) iff for any test \( t \) detecting \( g \),
\[
Z_g(t) = Z_f(t).
\]

So any test detecting \( g \) can be used to detect \( f \).

It is not necessarily true any test detecting \( f \) can be used to
detect \( g \).

Let \( T_g \) and \( T_f \) be all the tests detecting \( g \) and \( f \) respectively.

\( f \) dominates \( g \) iff they are functionally equivalence under \( T_g \).

It implies \( T_g \subseteq T_f \).

However, \( T_g \subseteq T_f \) does not imply \( f \) dominates \( g \).

In other words, even any test detecting \( g \) can detect \( f \), one
cannot say \( f \) dominates \( g \).
Consider two faults $g$ and $f$.

We say $f$ dominates $g$ iff for any test $t$ detecting $g$, $Z_g(t) = Z_f(t)$.

- So any test detecting $g$ can be used to detect $f$.
- It is not necessarily true any test detecting $f$ can be used to detect $g$.

Let $T_g$ and $T_f$ be all the tests detecting $g$ and $f$ respectively.

- $f$ dominates $g$ iff they are functionally equivalence under $T_g$.
- It implies $T_g \subseteq T_f$.

However, $T_g \subseteq T_f$ does not imply $f$ dominates $g$.

- In other words, even any test detecting $g$ can detect $f$, one cannot say $f$ dominates $g$. 
Dominance Fault Collapsing

\[ \text{x} \rightarrow \bullet \quad \text{\texttt{OR}} \quad \bullet \rightarrow \text{z} \]

\[ \text{y} \rightarrow \bullet \quad \text{\texttt{OR}} \quad \bullet \rightarrow \text{c} \]

\( z \ s-a-0 \) dominates \( y \ s-a-1 \).

- Dominance fault collapsing: a test set detecting \( y \ s-a-1 \) will detect \( z \ s-a-0 \).
  - Reduce number of faults we need to consider as long as we don’t need to distinguish them (fault location).
  - Also applies to OR/NOR/NAND gates.
- A fault model is preferred if its faults are generally dominated by faults from other fault models.
  - So we may detect more faults than we target at explicitly.
  - The single stuck-fault model appears to be the best.

(Fig. 4.12, Abramovici et al., 1990)
Dominance fault collapsing

- $z$ s-a-0 dominates $y$ s-a-1.
- Dominance fault collapsing: a test set detecting $y$ s-a-1 will detect $z$ s-a-0.
  - Reduce number of faults we need to consider as long as we don’t need to distinguish them (fault location).
  - Also applies to OR/NOR/NAND gates.
- A fault model is preferred if its faults are generally dominated by faults from other fault models.
  - So we may detect more faults than we target at explicitly.
  - The single stuck-fault model appears to be the best.
Dominance Fault Collapsing

$z$ s-a-0 dominates $y$ s-a-1.

Dominance fault collapsing: a test set detecting $y$ s-a-1 will detect $z$ s-a-0.

- Reduce number of faults we need to consider as long as we don’t need to distinguish them (fault location).
- Also applies to OR/NOR/NAND gates.

A fault model is preferred if its faults are generally dominated by faults from other fault models.

- So we may detect more faults than we target at explicitly.
- The single stuck-fault model appears to be the best.
Let $f$ be $z_2$ s-a-0 and $g$ be $y_1$ s-a-1.

Any test detect $g$ (10) will detect $f$, i.e. $T_g \subseteq T_f$.

But $f$ does not dominate $g$ since $Z_f(10) \neq Z_g(10)$.

However, such cases are not easy to be identified.
Let $f$ be $z_2 \text{ s-a-0}$ and $g$ be $y_1 \text{ s-a-1}$.

Any test detect $g$ (10) will detect $f$, i.e. $T_g \subseteq T_f$.

But $f$ does not dominate $g$ since $Z_f(10) \neq Z_g(10)$.

However, such cases are not easy to be identified.
Fault Collapsing without Dominance

Let $f$ be $z_2$ s-a-0 and $g$ be $y_1$ s-a-1.

Any test detect $g$ (10) will detect $f$, i.e. $T_g \subseteq T_f$.

But $f$ does not dominate $g$ since $Z_f(10) \neq Z_g(10)$.

However, such cases are not easy to be identified.

(Abramovici et al., 1990)
Let $f$ be $z2$ s-a-0 and $g$ be $y1$ s-a-1.

Any test detect $g$ (10) will detect $f$, i.e. $T_g \subseteq T_f$.

But $f$ does not dominate $g$ since $Z_f(10) \neq Z_g(10)$.

However, such cases are not easy to be identified.
Fault Equivalence and Fault Dominance

Fault Modeling for Sequential Circuits
Sequential Circuit Testing

- Considerably more difficult than testing combinational circuits.
  - Need a test sequence instead of a test vector.
  - Need to consider initial state.
  - The response is a sequence.
- Let $R(q, T)$ be the response of the sequential circuit $N$ to the test sequence $T$ starting in the initial state $q$.
  - Let $R_f(q, T)$ be the response in the presence of the fault $f$.
- A test sequence $T$ strongly detects the fault $f$ iff
  - $R(q_1, T) \neq R_f(q_2, T)$ for every pair of initial states $q_1$ and $q_2$. 
Sequential Circuit Testing

- Considerably more difficult than testing combinational circuits.
  - Need a test sequence instead of a test vector.
  - Need to consider initial state.
  - The response is a sequence.
- Let $R(q, T)$ be the response of the sequential circuit $N$ to the test sequence $T$ starting in the initial state $q$.
  - Let $R_f(q, T)$ be the response in the presence of the fault $f$.
- A test sequence $T$ strongly detects the fault $f$ iff
  - $R(q_1, T) \neq R_f(q_2, T)$ for every pair of initial states $q_1$ and $q_2$. 
Sequential Circuit Testing

- Considerably more difficult than testing combinational circuits.
  - Need a test sequence instead of a test vector.
  - Need to consider initial state.
  - The response is a sequence.
- Let $R(q, T)$ be the response of the sequential circuit $N$ to the test sequence $T$ starting in the initial state $q$.
  - Let $R_f(q, T)$ be the response in the presence of the fault $f$.
- A test sequence $T$ strongly detects the fault $f$ iff
  - $R(q_1, T) \neq R_f(q_2, T)$ for every pair of initial states $q_1$ and $q_2$. 
Example Sequential Circuit

(Fig. 4.9, Abramovici et al., 1990)
Example Test Sequence

$T = 10111$ strongly detects $\beta$ but not $\alpha$.

However, to strongly detect a fault is not practical.

- Need to list responses for all initial state.
- No guarantee on when the difference appears – the outputs must be compared per test vector.
- The difficulty is mostly due to the unknown initial states.

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Output sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fault-free</td>
</tr>
<tr>
<td>$A$</td>
<td>01011</td>
</tr>
<tr>
<td>$B$</td>
<td>11100</td>
</tr>
<tr>
<td>$C$</td>
<td>00011</td>
</tr>
<tr>
<td>$D$</td>
<td>11001</td>
</tr>
</tbody>
</table>

(Fig. 4.9, Abramovici et al., 1990)
Example Test Sequence

- $T = 10111$ strongly detects $\beta$ but not $\alpha$.
- However, to strongly detect a fault is not practical.
  - Need to list responses for all initial state.
  - No guarantee on when the difference appears – the outputs must be compared per test vector.
  - The difficulty is mostly due to the unknown initial states.

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Fault-free</th>
<th>$\alpha$ (a s—a-1)</th>
<th>$\beta$ (b s—a-0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>01011</td>
<td>01010</td>
<td>01101</td>
</tr>
<tr>
<td>$B$</td>
<td>11100</td>
<td>11100</td>
<td>11101</td>
</tr>
<tr>
<td>$C$</td>
<td>00011</td>
<td>00010</td>
<td>01010</td>
</tr>
<tr>
<td>$D$</td>
<td>11001</td>
<td>10010</td>
<td>11010</td>
</tr>
</tbody>
</table>

(Fig. 4.9, Abramovici et al., 1990)
One may apply an initialization sequence $T_I$ first to bring the possibly faulty circuit to a known state, and then apply another sequence $T'$ where the output is compared to the fault-free one.

- The fault-free circuit should be brought to a known state by $T_I$ so that only one output sequence under $T'$ need to be generated (and stored).
- However, a fault may prevent proper initialization.

The concepts of fault equivalence and fault dominance can be extended to sequential circuits though they are not practical for testing.
Discussions

- One may apply an initialization sequence $T_I$ first to bring the possibly faulty circuit to a known state, and then apply another sequence $T'$ where the output is compared to the fault-free one.
  - The fault-free circuit should be brought to a known state by $T_I$ so that only one output sequence under $T'$ need to be generated (and stored).
  - However, a fault may prevent proper initialization.

- The concepts of fault equivalence and fault dominance can be extended to sequential circuits though they are not practical for testing.
Fault equivalence and fault dominance can be utilized to reduce the number of faults one need to consider for fault detection.

To test sequential circuits is considerably more difficult than to test combinational circuits.