Random Testing

• Random testing, in some form, is common for both hardware or software testing.

• It is sometimes assumed that an “average” fault can represent most faults. In reality some faults are easy to find, while some faults are very hard to find.

• For highly reliable, the real challenge is in finding hard-to-test bugs.

• “Detectability-profile” concept is introduced here.
Random Testing: Outline

• Random Testing (RT): advantages and tradeoffs
• RT vs pseudorandom testing (PR)
• Coverage and detectability profile (DP)
• Hardware and software DPs
• Connection between coverage and faults found?
• High and low testability faults during early & late testing
• Implications of an asymmetric DP
Random Testing

• Extensively used for both hardware and software
• Ideally each input is selected randomly. PR (Pseudorandom) schemes approximate random.
• Generally quite effective for moderate coverage.
  ▪ Coverage hard to determine a priori.
  ▪ Ineffective for random-pattern-resistant faults.
  ▪ Coverage tools: Random (functional) followed by Structural testing.
Random Testing: Advantage

- No test generation using structural information needed.
- Test set-up using comparison:

- Alternative: Is response reasonable?

Q: For software testing, how do you know the expected response?
Pseudorandom (PR) Testing

• Unlike true random, reproducible.
• Will not repeat until all combinations applied.
• Generation: usually just-in-time (not stored).
  ▪ Autonomous linear feedback shift register (ALFSR).
  ▪ Cellular automata etc possible.
• Some randomness properties satisfied, but not all.

Ex: Set of vectors with more 1s than 0s is not quite random.
Coverage Achieved

- Coverage grows fast in the beginning, saturates near end.
- Is it described by
  - \( C(L) = 1 - e^{-aL} \)?
  - No, doesn’t fit.
- It is controlled by distribution of detectability of faults.
- Detectability profile (Malaiya & Yang ’84):
- \( H = \{ h_1, h_2, \ldots h_N \} \)
  - \( N \): total possible vectors
  - \( h_k \): number of faults detected by exactly \( k \) vectors.
- Total faults \( M = \Sigma h_k \)
- \( h_1 \): number of least testable faults

Ex: Circuit with higher \( h_1 \) would be harder to test.
Detection Probability

• **Detection probability**: if there are \( N \) distinct possible vectors, and if a fault is detected by \( k \) of them, then its detection probability is \( \frac{k}{N} \)

• A fault with detection probability \( \frac{1}{N} \) would be hardest to test, since it is tested by only one specific test and none other.
Detectability Profiles: Ex

- **CECL Full adder**
  Inputs=4 (N=16), M=90
  \[ H = (h_1, h_2, h_3, h_4, h_5, h_6, h_8) = (1, 11, 2, 43, 21, 4, 8) \]

- **Schneider’s counterexample circuit:**
  Inputs= 4 (N=16), M=44
  \[ H = (h_1, h_2, h_3, h_{14}) = (23, 19, 1, 1) \]

*Schneider’s counterexample circuit* has 23 hard to test faults. A random vector has probability 1/16 to detect any one of them.
Coverage with L random vectors

- \( h_k \) out of \( M \) defects detectable by exactly \( k \) vectors: detection probability \( k/N \)
- \( P\{\text{a defect with dp } k/N \text{ not detected by a vector}\} = \left(1 - \frac{k}{N}\right) \)
- \( P\{\text{a defect with dp } k/N \text{ not detected by } L \text{ vectors}\} = \left(1 - \frac{k}{N}\right)^L \)
- Of \( h_k \) faults, expected number not covered is \( \left(1 - \frac{k}{N}\right)^L h_k \)
- Expected test coverage with \( L \) vectors

\[
C(L) = 1 - \sum_{k=1}^{N} \left(1 - \frac{k}{N}\right)^L \frac{h_k}{M}
\]
Coverage Obtained by L Vectors

• For PR tests (McClusky 87)

\[
C(L) = 1 - \sum_{k=1}^{N-L} \frac{(N-L) C_k}{N} \frac{h_k}{M} \\
\approx 1 - \sum_{k=1}^{N} \left(1 - \frac{k}{N}\right)^L \frac{h_k}{M} \quad \text{(for Random)}
\]

• For large L, terms with only low k (i.e. faults that are hard to test) have an impact. Thus only lower elements of H need to be estimated.

• For CECL Full Adder,

\[
C(15) = 1 - [4.2 + 16.4 + 0.9 + 6.3 + 0.84 + 0.03 + 0 + \ldots] \cdot 10^{-3}
\]

Pseudorandom (PR): a vector cannot repeat, unlike in true Random.

More in Appendix 1
Detectability Profile: software

- Regardless of initial profile, after some initial testing, the profile will become asymmetric.
- In the early development phases, inspection and early testing are likely to remove most easy to test bugs, while leaving the almost all hardest to test bugs still in.
Detectability Profile: software

- Adam’s Data for a large IBM software product


Notice: Fewer bugs with higher detection rates
Detectability Profile: Software

- Software detectability profile is exponential
- Justification: Early testing will find & remove easy-to-test faults.
- Testing methods need to focus on hard-to-find faults.

As testing time progresses, more of the faults are clustered to the left.
Implications of Asymmetrical DP

- Faults are not alike, and an “average” fault does not represent a hard-to-test fault.
- A fault injected artificially typically does not represent a hard-to-test fault.
- Faults found early during testing are not a good sample of faults that will be found later during testing.
- See Appendix II for details.
References

Appendix I

Ex: Coverage for CECL adder

- 16 vectors (0000 to 1111), 90 potential defects (transistor level)
- Shows coverage obtained for faults with detectability.
- A fault with det prob 1/16 has probability 0.0625 to be detected with 1 test, while fault with det prob 8/16 has 0.5 (i.e. 50%) of getting tested by it.
- With 20 vectors, 28% of faults with det prob 1/16 will still be left undetected, while those with det prob 8/16 will almost certainly be found.
Ex: C(L) and components for CECL Full Adder

<table>
<thead>
<tr>
<th>CECL full adder</th>
<th>N</th>
<th>16</th>
<th>M</th>
<th>90</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Hk</th>
<th>1</th>
<th>11</th>
<th>2</th>
<th>43</th>
<th>21</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>k =&gt;</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Coverage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L</th>
<th>0</th>
<th>0.0625</th>
<th>0.1250</th>
<th>0.1875</th>
<th>0.2500</th>
<th>0.3125</th>
<th>0.3750</th>
<th>0.5000</th>
<th>0.2736</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0.0625</td>
<td>0.1250</td>
<td>0.1875</td>
<td>0.2500</td>
<td>0.3125</td>
<td>0.3750</td>
<td>0.5000</td>
<td>0.2736</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0.2758</td>
<td>0.4871</td>
<td>0.6459</td>
<td>0.7627</td>
<td>0.8464</td>
<td>0.9046</td>
<td>0.9688</td>
<td>0.7652</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0.4755</td>
<td>0.7369</td>
<td>0.8746</td>
<td>0.9437</td>
<td>0.9764</td>
<td>0.9909</td>
<td>0.9990</td>
<td>0.9263</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>0.6202</td>
<td>0.8651</td>
<td>0.9556</td>
<td>0.9866</td>
<td>0.9964</td>
<td>0.9991</td>
<td>1.0000</td>
<td>0.9710</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0.7249</td>
<td>0.9308</td>
<td>0.9843</td>
<td>0.9968</td>
<td>0.9994</td>
<td>0.9999</td>
<td>1.0000</td>
<td>0.9865</td>
</tr>
</tbody>
</table>

After 20 vectors:

| covered | 0.72 | 10.24 | 1.97 | 42.86 | 20.99 | 4.00 | 8.00 |
| remaining | 0.28 | 0.76 | 0.03 | 0.14 | 0.01 | 0.00 | 0.00 |
Coverage of partitions

The plot in the next slide for CECL Full Adder shows that
• Faults with high detection probability get covered soon,
• while those with low detection probability are resistant to random testing.
Coverage of partitions
Shift in profile with progress in testing

Next slide for CECL Full Adder

• Assume that a fault is removed from consideration when found
• X-axis is k (k=1 hardest to find)

• Plot shows that at the beginning there are nearly 50% of the faults with det prob k/N = 4/16.
• After 20 vectors, more than 60% of the remaining faults have det prob 2/16.
• “Low hanging fruit” get picked quicker.
Shift in profile with progress in testing

![Graph showing undetected faults before and after 20 vectors.](chart.png)
Appendix II
Implications of Asymmetrical DP

• Fault seeding
• Fault sampling
• Fault exposure ratio
Implications: Fault Seeding

- A program has x defects. We want to estimate x.
- Seed j new faults.
- Do some testing. Let faults found be j₁ seeded faults and x₁ original faults.
- Assuming \( \frac{j_1}{j} = \frac{x_1}{x} \) we get \( x = x_1 \frac{j}{j_1} \)

However, in reality the x faults include harder faults to test,

\[
\frac{j_1}{j} > \frac{x_1}{x} \quad \text{hence} \quad x > \frac{x_1 j}{j_1}
\]
Implications: Estimation by Inspection Sampling

• Software with $x$ bugs is inspected by two separate teams that finds $x_1$ and $x_2$ bugs respectively, of which $x_3$ are shared.

• Assuming $\frac{x_1}{x} = \frac{x_3}{x_2}$ we get

\[ x = \frac{x_1 x_2}{x_3} \]

• However actually since $x$ includes more harder to test faults,

\[ \frac{x_3}{x_2} > \frac{x_1}{x} \quad hence \quad x > \frac{x_1 x_2}{x_3} \]
Implications: fault exposure ratio

Let $N(t)$ be the number of bugs at time $t$ during testing, then if $a$ is a parameter,

$$\frac{dN(t)}{dt} = -aN(t)$$

If $a$ is constant, then $N(t) = N(0)e^{-at}$ [expo SRGM]

However in random testing $a$ should decline as faults get harder to find.

If testing is intelligent, then $a$ can rise, which can give rise to Logarithmic SRGM.

Don’t worry about this here, we will come to it when we will study software reliability.