Introduction to Discrete Event Simulation II

John Colley

13th February 2012

Southampton
Introduction

• Discrete Event Simulation Principles – Review
• The Testbench
• Simulation-based Verification
• Sign Off
• Summary
Discrete Event Simulation

COMPONENT VIEW

Components: A, B, C, D (processes)
Connections: C1, C2 (unidirectional)
Ports: IN, OUT

SIMULATOR API

GetValue(port)
HasChanged(port)
SetValue(OUT port, val, delay)
ScheduleEval(component, delay)
The Two-list Simulation Algorithm

$$t = 0$$

$$t = n$$

update list

evaluation list
Time Zero Initialisation: Evaluate all Components
Component evaluations call *SetValue, ScheduleEval*

![Diagram showing component evaluations and scheduled evaluations]

- **A** connected to **C1**
- **B** connected to **C1**
- **C** connected to **C2**
- **D** connected to **C2**

**Update list**:
- t = 50
- t = 30
- t = 20
- t = 0

**Evaluation list**:
- EC
- ED
- C2
- C1
Component evaluations call *SetValue, ScheduleEval*

- **A**: C1 new val
- **B**: ScheduleEval
- **C**: C2 new val
- **D**: D future eval

**Timeline:**
- **t = 0**: EC update list
- **t = 20**: C1
- **t = 30**: ED C2
- **t = 50**: EC
Global Time is Advanced to $t = 20$

```
Global Time is Advanced to t = 20
```

```
update list

<table>
<thead>
<tr>
<th>t</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>C1</td>
</tr>
<tr>
<td>30</td>
<td>C2</td>
</tr>
<tr>
<td>50</td>
<td>EC</td>
</tr>
</tbody>
</table>
```

```
evaluation list
```
Add to eval list each component on C1 fanout
B calls *ScheduleEval* with delay 40
Time advances to $t = 30$: two updates

- C1 at $t = 30$
- C2 at $t = 30$
Time advances to $t = 30$: \textit{two} updates, \textit{one} eval
The Simulation Testbench

Primary inputs

Primary outputs

A → C1 → B

C → C2 → D
The Simulation Testbench

In *principle*, just another component using the API

In *practice*, a complex model of the design environment:
- Constrained Random Test Generation
- Assertion Checking
- Functional Coverage Metrics
Simulation-based Verification

• As opposed to Formal Verification
  – Theorem Proving
  – Model Checking
• Structural Coverage
• Functional Coverage
• Assertion Checking
• Assertion Coverage
• Integrating Formal and Simulation-based Verification
Structural Coverage

Do the requirements-based test cases adequately exercise the structure of the source code?

DO 178B

- Statement Coverage
- Branch (Decision) Coverage
- MC/DC (Modified Condition/Decision Coverage)
  - Unique Cause
  - Masking
MC/DC
(Modified Condition/Decision Coverage)

$$Z = (A \text{ or } B) \text{ and } (C \text{ or } D)$$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
MC/DC
(Modified Condition/Decision Coverage)

\[ Z = (A \text{ or } B) \text{ and } (C \text{ or } D) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Unique Cause
MC/DC
(Modified Condition/Decision Coverage)

\[ Z = (A \text{ or } B) \text{ and (C or D)} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Masking
MC/DC
(Modified Condition/Decision Coverage)
Summary

• Masking MC/DC results in fewer tests and shorter simulation runs
• Unique-Cause MC/DC cannot deal with repeated conditions
• Unique-Cause MC/DC may detect more errors (NB not the actual DO 178B requirement)

Do the requirements-based test cases adequately exercise the structure of the source code?

DO 178B
DO 178 C

• Approved: December 2011
• Provision for Formal Methods
• Provision for Object-Oriented Code Development
  – As opposed to Structured Code Development
  – Does MC/DC have the same value?
Bug Rate vs Time

Tandem Computers ca. 1990

Sign Off
Bug Rate vs Time – *Coverage Directed* Verification

![Graph showing Bug Rate vs Time with Coverage Directed Verification.](image-url)

- Tandem Computers ca. 1990
- Sign Off

**Legend:**
- Bug Rate
- Time
- Coverage Directed Verification
Bug Rate vs Time – **Coverage Directed** Verification

- Sign Off
- Tandem Computers ca. 1990

[Graph showing bug rate decreasing with time, with two curves and a star indicating 'Sign Off' marker.]
Bug Rate vs Time – Coverage Directed Verification

**Tandem Computers ca. 1990**

- Sign Off
- Sign Off

Sharper “knee”

- 1
- 2
Functional Coverage

• *User-defined* Coverage Metrics
  – Typical Scenarios
  – Error Cases
  – Corner Cases

• High-Level Language Description

• Constrained Random Test Generation from Coverage Description

• “Scoreboarding”
### Functional Coverage Example

#### Transaction Coverage

<table>
<thead>
<tr>
<th>Transaction Type</th>
<th>Header1</th>
<th>Header2</th>
<th>Typical Value</th>
<th>Min. Value</th>
<th>Max. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Functional Coverage Example

### Transaction Coverage

<table>
<thead>
<tr>
<th>Transaction Type</th>
<th>Header1</th>
<th>Header2</th>
<th>Typical Value</th>
<th>Min. Value</th>
<th>Max. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td><img src="green" alt="Circle" /></td>
<td><img src="green" alt="Circle" /></td>
<td><img src="red" alt="Circle" /></td>
<td><img src="red" alt="Circle" /></td>
<td><img src="green" alt="Circle" /></td>
</tr>
<tr>
<td>B</td>
<td><img src="green" alt="Circle" /></td>
<td><img src="red" alt="Circle" /></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td><img src="green" alt="Circle" /></td>
<td>-</td>
<td><img src="green" alt="Circle" /></td>
<td><img src="green" alt="Circle" /></td>
<td><img src="green" alt="Circle" /></td>
</tr>
</tbody>
</table>

**Functional Coverage:** 8/11 (73%)
Assertion Checking: PSL
(Property Specification Language)

- LTL-based
- Unit of time is the Clock Cycle
- SEREs
  - Sequential Extended Regular Expressions

\{ a ; \text{not } a ; b \} \implies \{ c \}
Assertion Checking: PSL
(Property Specification Language)

• PSL is converted to \textit{Simulation Monitors}
  – First, convert to Buchi Automata (non-deterministic)
  – Second, generate deterministic automata
  – Third, generate HDL representation

• Simulate the Monitors together with the Design

\[
\{ \ a ; \ \text{not} \ a \ ; \ b \ \} \ \Rightarrow \ \{ \ c \ \}
\]
Assertion Coverage

• How effectively do the tests exercise the Assertions?
  – Vacuously Satisfied? ( a is always false)
  – Are the SEREs sensitised to detect assertion failure?

\{ a ; \text{ not } a ; b \} \implies \{ c \}

• Is the set of assertions
  – Necessary?
  – Sufficient?
Integrating Formal and Simulation-based Verification

• PSL SEREs can be verified using a model checker
  – Assertion Coverage principles still apply
  – Assertion Coverage results for simulation an model checking can be combined

• Are the assertions necessary/sufficient?
  – An open problem
  – Scope for using theorem proving?
Verification Sign Off

• Ultimately, an engineer will have to physically provide a signature
• How confident is the engineer that the design meets its specification?
  – The outcome of the verification process must be measurable
    • Bug rate
    • Coverage metrics
Summary

• Discrete Event Simulation
  – Two-List Algorithm for Deterministic Execution

• Coverage-Driven Verification
  – Structural (MC/DC)
  – Functional

• Assertion Checking/Coverage

• Combining Formal and Simulation-based Verification for
  – Earlier Sign Off with
  – Increased Confidence