

# CSC313 High Integrity Systems/ CSCM13 Critical Systems

Course Notes  
Additional Material  
Chapter 5: Fault Tolerance

Anton Setzer  
Dept. of Computer Science, Swansea University

<http://www.cs.swan.ac.uk/~csetzer/lectures/critsys/current/index.html>

December 8, 2016

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques
- 5 (g) Hardware Fault Tolerance
- 5 (h) Software Fault Tolerance
- 5 (i) Fault Tolerant Architectures
- 5 (j) Example: The Space Shuttle

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques
- 5 (g) Hardware Fault Tolerance
- 5 (h) Software Fault Tolerance
- 5 (i) Fault Tolerant Architectures
- 5 (j) Example: The Space Shuttle

No Additional Material

For this subsection no additional material has been added yet.

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques
- 5 (g) Hardware Fault Tolerance
- 5 (h) Software Fault Tolerance
- 5 (i) Fault Tolerant Architectures
- 5 (j) Example: The Space Shuttle

## Classification by Nature

Faults can be classified by their nature: random vs. systematic faults:

- ▶ **(i) Random faults.**
  - ▶ Random faults are faults in components of a system, which occur with a certain probability.
  - ▶ We can predict random faults by collecting statistical data from large numbers of samples of similar components.
  - ▶ Random faults are usually hardware faults.
    - ▶ Reason for random faults are that any hardware is subject to environmental influences, which might affect its correct operation.
    - ▶ E.g. radioactivity, radiation, humidity, warmth, cold, wear and tear.

## (b) Types of Faults

- ▶ Faults can be characterised by the following criteria:
  - ▶ Nature (random/systematic).
  - ▶ Duration.
  - ▶ Extent.

## Classification by Nature

- ▶ In software reliability engineering one considers as well software errors as random.
- ▶ Because random faults can be predicted well, they are more easy to tolerate.

## Classification by Nature

- ▶ **(ii) Systematic Faults**
  - ▶ Systematic faults are faults which are not random.
    - ▶ Either a component has it, or it doesn't have it.
  - ▶ Software faults are usually considered to be systematic.
  - ▶ Three kinds of systematic faults:
    - ▶ Mistakes in the specification of a system.
    - ▶ Mistakes in the implementation of software.
    - ▶ Mistakes in the design of hardware.
  - ▶ Difficult to tolerate.
    - ▶ E.g. if two programmers write the same program, it might be that both make the same systematic mistake.

## Classification by Duration

- ▶ **(ii) Transient faults**
  - ▶ Appear, and vanish again.
  - ▶ Typical example are effects of radioactive particles hitting a semi conductor of a memory chip.
    - ▶ If it happens, the state of a few bits is changed.
    - ▶ But there is no lasting damage to the chip.
  - ▶ Although infrequent and not lasting, one needs to take steps to correct this error before a system error is caused.

## Classification by Duration

Faults can be classified by their duration:

- ▶ **(i) Permanent faults**
  - ▶ Remain in existence indefinitely, until corrective action is taken.
  - ▶ Software faults are always permanent.
  - ▶ Many hardware component faults are permanent.

## Classification by Duration

- ▶ **(iii) Intermittent faults**
  - ▶ Appear, disappear, and then reappear after some time.
  - ▶ Results of
    - ▶ poor solder joints, corrosion on connector contacts.  
At some times connections are possible, at others not.
    - ▶ electromagnetic radiation.
    - ▶ **Electromagnetic compatibility (EMC)** is the ability of a system to work correctly in the presence of (electromagnetic) interference from other electrical equipment, and not to interfere with other equipment or other parts itself.

## Classification by Duration

- ▶ Problems of electromagnetic radiation occur
  - ▶ within wires of a digital circuits
  - ▶ between computers and other sources of noise (usually caused by direct electromagnetic radiation or by coupling through common power lines).
  - ▶ Particular problems close to car engines, jet engines, nuclear power reactors, high-power electric motors.
  - ▶ Mobile phones and CD/DVD players cause nowadays problems (e.g. not allowed in planes).
- ▶ **Software errors**, especially those caused by race conditions, often appear to be intermittent, but are **always permanent**.

5 (a) Introduction

5 (b) Types of Faults

5 (c) Fault Models

5 (d) Fault Coverage

5 (e) Redundancy

5 (f) Fault Detection Techniques

5 (g) Hardware Fault Tolerance

5 (h) Software Fault Tolerance

5 (i) Fault Tolerant Architectures

5 (j) Example: The Space Shuttle

## Classification by Extent

Faults can be classified by their extent:

- ▶ **(i) Localised faults** affect only a single hardware or software module.
- ▶ **(ii) Global faults** have effects that permeate through the entire system.

## (c) Fault Models

- ▶ In order to analyse the effect of **hardware faults** on the entire system, one uses fault models.
- ▶ These are **not perfect representations** of what is physically actually happening.
- ▶ However, they help to design **test procedures**, to simulate **fault conditions**, and to develop **fault tolerant** systems.
  - ▶ Testing of safety critical software needs to take into account that safety requirements are met even if one or two hardware components fail.

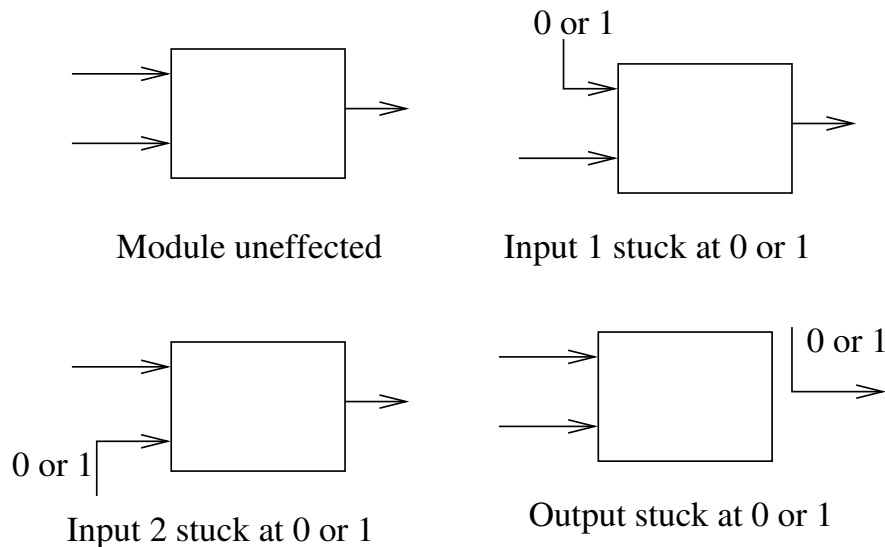
# Fault Models

- ▶ We consider 3 fault models:
  - ▶ **Single-stuck-at fault model.**
  - ▶ **Bridging fault model.**
  - ▶ **Stuck-open fault model.**

# Single-Stuck-At Model

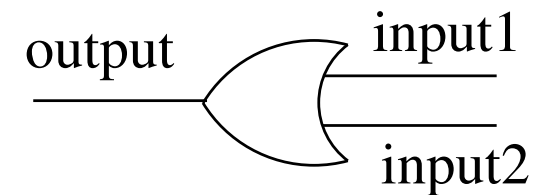
- ▶ **Single-stuck-at model** assumes that a fault within a module causes it
  - ▶ to respond as if one of its inputs or outputs is stuck at logic 0 or 1.
  - ▶ and such that the basic functionality of the circuit is otherwise unaffected.

## Example (Single-Stuck-At Model)



## Example: Or-Gate

- ▶ Assume for instance an or gate with inputs input1, input2 and output output:



- ▶ If input1 is stuck at 0 then the output is computed as follows:

$$\text{output} = 0 \vee \text{input2} = \text{input2}$$

## Example: Or-Gate

- ▶ If input1 is stuck at 1, then the output is constant 1:

$$\text{output} = 1 \vee \text{input2} = 1$$

- ▶ This is identical to the situation where the output is stuck at 1.

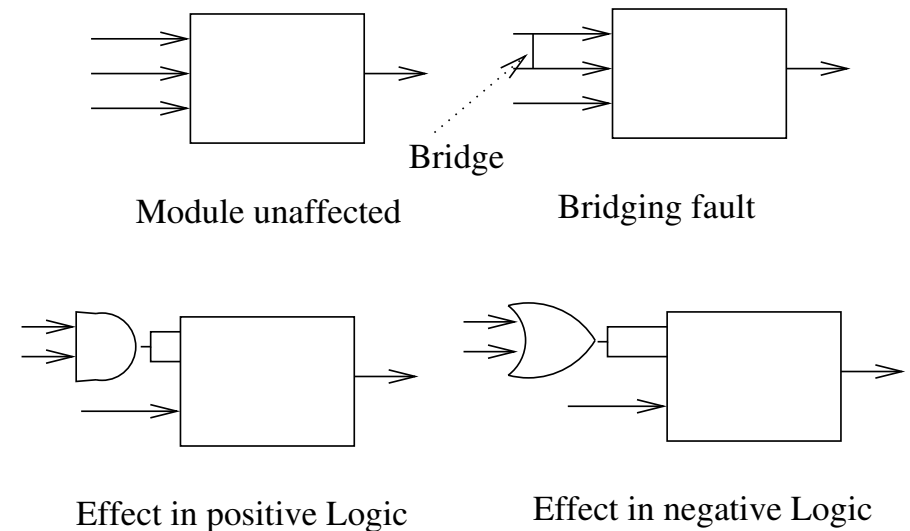
## Analysis of Single-Stuck-At Model

- ▶ Majority of faults arising from broken tracks, open or short-circuit components and shorts between tracks can be represented by the single-stuck-at model.
- ▶ Single-stuck-at model cannot represent accurately transient or intermittent faults.
- ▶ **Complexity of testing:**
  - ▶ For a circuit with N nodes there are only 2N single-stuck-at faults: one for each node stuck at 0, one for each node stuck at 1.
  - ▶ Exhaustive testing feasible.

## Bridging Model

- ▶ A **bridging** or **short-circuit fault** occurs when two or more nodes in a circuit are accidentally joined together to form a permanent fault.
- ▶ In positive logic, i.e.
  - ▶ 1 represented by power on,
  - ▶ 0 represented by power off,
 a bridging fault between two inputs has the effect of both inputs being ANDed together (see next slide).
- ▶ In negative logic, i.e.
  - ▶ 1 represented by power off,
  - ▶ 0 represented by power on,
 a bridging fault between two inputs has the effect of both inputs being ORed together (see next slide).

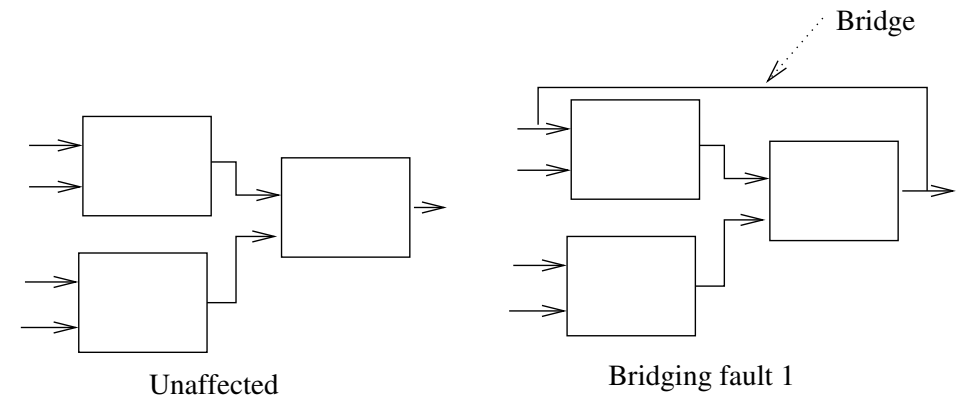
## Simple Example (Bridging Fault)



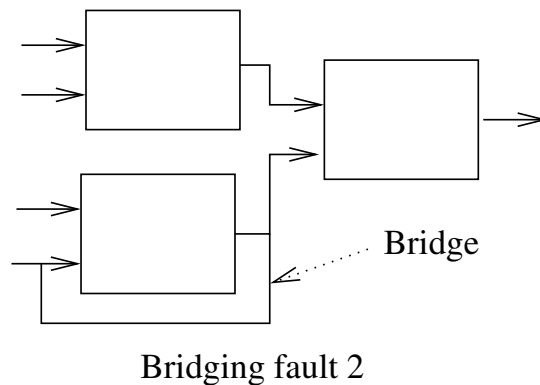
## More Complex Bridging Faults

- ▶ Bridges between inputs and outputs might result in converting combinatorial circuits into sequential ones, and might result in instability or oscillation.

## Complex Example (Bridging Fault)



## Complex Example (Bridging Fault)



## Analysis of Bridging Model

- ▶ Bridging faults behave usually different from single-stuck-at faults.
- ▶ **Complexity of testing** more complex:
  - ▶ For a circuit with  $N$  nodes there are  $\binom{N}{M}$  bridging faults between  $M$  nodes at the same time.
    - ▶ Especially, there are  $\binom{N}{2} = \frac{N \cdot (N-1)}{2}$  bridging faults between two nodes.
    - ▶ Makes exhaustive testing impossible in most cases.

## The Stuck-Open Model

- ▶ **Stuck-open** fault occurs if in a CMOS gate, both output transistors are turned off because of an internal open- or short-circuit.
- ▶ Therefore output is neither pulled to high nor to low.
- ▶ Depending on the exact fault, the gate will alternate between
  - ▶ driving the output
  - ▶ or maintaining its previous output.
- ▶ Length of time it maintains its output depends on the gate and the nature of the fault.
- ▶ Therefore circuit gets a complex sequential characteristic.

## Use of Fault Models

- ▶ Testing for bridging faults usually infeasible as well. Usually restriction to testing for single occurrences of single-stuck-at faults.
- ▶ Fault models can be used for developing strategies for tolerating faults.
- ▶ **Limitations:**
  - ▶ Hardware design faults (especially wrong logic design) are usually not covered by those fault models.
  - ▶ Software faults are usually not covered by these models.
    - ▶ In software reliability engineering, fault models for software are developed.

## Use of Fault Models

- ▶ Exhaustive testing of circuits is not feasible except for simple combinatorial circuits.
- ▶ Using fault models, test vectors can be developed which test for faults occurring by one of the above fault models.
  - ▶ Testing only feasible by assuming single failures.
  - ▶ E.g. a circuit with  $N$  nodes can have  $3^N - 1$  multiple stuck-at faults, which is infeasible to test.
    - Why  $3^N - 1$  faults?
      - ▶ Each of the nodes can be error free, stuck at 1 and stuck at 0, giving  $3^N$  possibilities.
      - ▶ The only case when the circuit is correct is when all nodes are error free. Excluding it we get  $3^N - 1$  faulty cases.

5 (a) Introduction

5 (b) Types of Faults

5 (c) Fault Models

5 (d) Fault Coverage

5 (e) Redundancy

5 (f) Fault Detection Techniques

5 (g) Hardware Fault Tolerance

5 (h) Software Fault Tolerance

5 (i) Fault Tolerant Architectures

5 (j) Example: The Space Shuttle



## (d) Fault Coverage

- ▶ **Fault coverage** is the fraction of possible faults that can be avoided, removed, detected or tolerated.
  - ▶ Usually it is difficult to give a numerical estimate, except when good fault models can be used.
- ▶ **Fault removal coverage** is the fraction of faults found during the testing phase of system development.
  - ▶ Testing vectors aim at 100% fault removal coverage for the faults in the underlying fault model.
  - ▶ However, fault models never include all possible faults.
  - ▶ Especially, most models cover only single faults and don't cover transient or intermittent faults.
  - ▶ Therefore, fault removal coverage is never 100%.

5 (a) Introduction

5 (b) Types of Faults

5 (c) Fault Models

5 (d) Fault Coverage

5 (e) Redundancy

5 (f) Fault Detection Techniques

5 (g) Hardware Fault Tolerance

5 (h) Software Fault Tolerance

5 (i) Fault Tolerant Architectures

5 (j) Example: The Space Shuttle

## Fault Coverage

- ▶ **Fault detection coverage** is the ability of a system to detect faults during operation.
  - ▶ Using fault models, fault detection coverage can be estimated, but we have the same limitations as above.
- ▶ **Fault tolerance coverage** is the ability of a system to tolerate faults.

## No Additional Material

For this subsection no additional material has been added yet.

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques**
- 5 (g) Hardware Fault Tolerance
- 5 (h) Software Fault Tolerance
- 5 (i) Fault Tolerant Architectures
- 5 (j) Example: The Space Shuttle

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques
- 5 (g) Hardware Fault Tolerance**
- 5 (h) Software Fault Tolerance
- 5 (i) Fault Tolerant Architectures
- 5 (j) Example: The Space Shuttle

## No Additional Material

For this subsection no additional material has been added yet.

## No Additional Material

For this subsection no additional material has been added yet.

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques
- 5 (g) Hardware Fault Tolerance
- 5 (h) Software Fault Tolerance**
- 5 (i) Fault Tolerant Architectures
- 5 (j) Example: The Space Shuttle

## No Additional Material

For this subsection no additional material has been added yet.

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques
- 5 (g) Hardware Fault Tolerance
- 5 (h) Software Fault Tolerance
- 5 (i) Fault Tolerant Architectures**
- 5 (j) Example: The Space Shuttle

## No Additional Material

For this subsection no additional material has been added yet.

- 5 (a) Introduction
- 5 (b) Types of Faults
- 5 (c) Fault Models
- 5 (d) Fault Coverage
- 5 (e) Redundancy
- 5 (f) Fault Detection Techniques
- 5 (g) Hardware Fault Tolerance
- 5 (h) Software Fault Tolerance
- 5 (i) Fault Tolerant Architectures
- 5 (j) Example: The Space Shuttle

## No Additional Material

For this subsection no additional material has been added yet.