Möbius Tool

LAB 03

Contacts

Maurizio Palmieri

Post Doc of the Department of Information Engineering, University of Pisa

Office: Largo Lucio Lazzarino 1 - 56122 Pisa (PI), Italy

Email: maurizio.palmieri@ing.unipi.it

Overview

- Tutorial on highly redundant fault-tolerant multiprocessor system
  - System description
  - Composition model
  - San models
  - Other elements
- Exercise

System Description 1/2

At the highest level, the system consists of multiple computers.

- The system is considered operational if at least 1 computer is operational.

Each computer is composed of

- 3 memory modules, of which 1 is a spare
- 3 CPU units, of which 1 is a spare
- 2 I/O ports, of which 1 is a spare
- 2 non-redundant error-handling chips

A computer is operational if:

- at least 2 memory modules are functioning
- at least 2 CPU units are functioning
- at least 1 I/O port is functioning
- the 2 error-handling chips are functioning.

System Description 2/2

Each memory module consists of:

- 41 RAM chips (2 are spare)
- 2 non-redundant interface chips.

➢A memory module is operational if at least 39 of its 41 RAM chips, and its 2 interface chips, are working.

Each CPU unit consists of

- 6 non-redundant chips.

➢A CPU unit is operational if all the 6 chips are working

Each I/O port consists of

- 6 non-redundant chips

➢An I/O port is operational if all the 6 chips are working

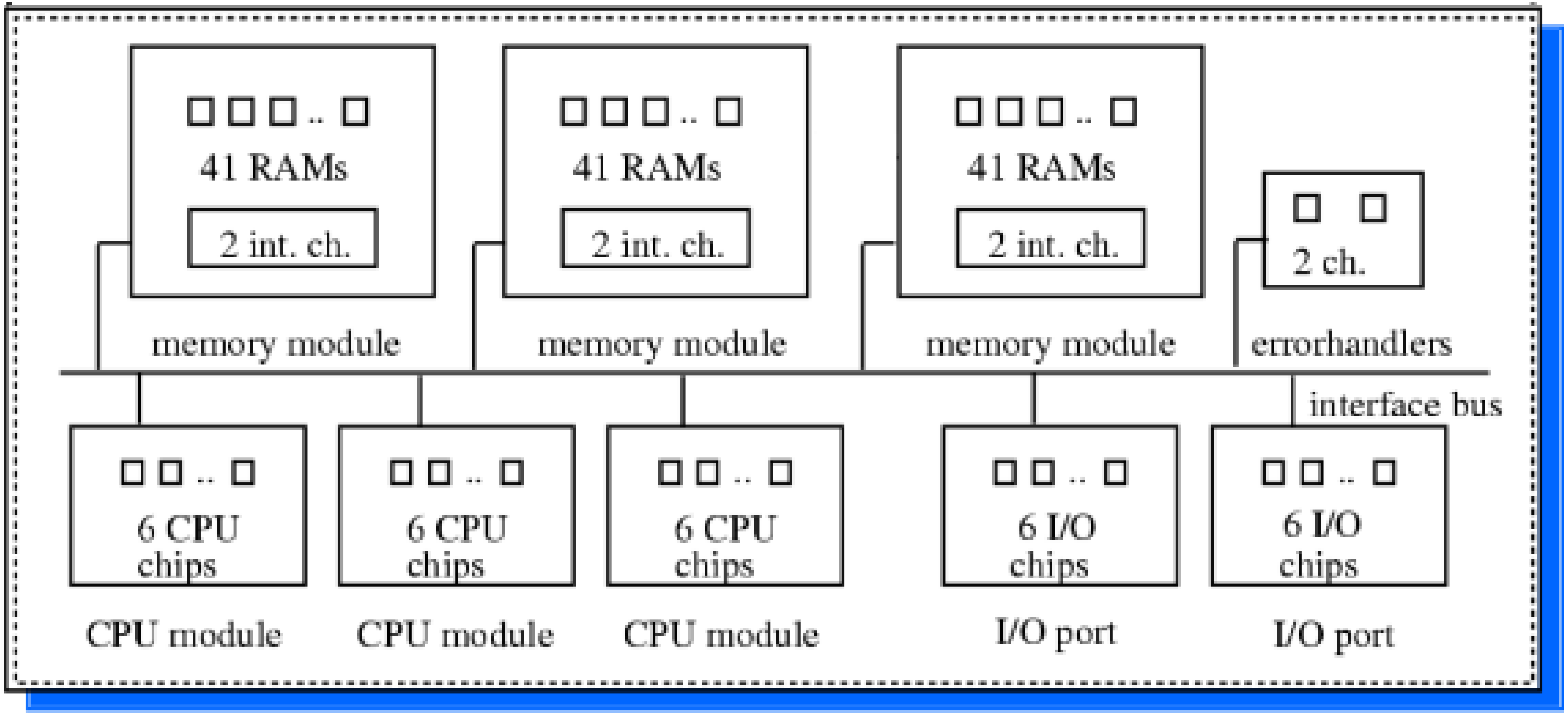
Fault Coverage

Where there is redundancy (available spares) at any level of system hierarchy, there is a coverage factor associated with the component failure at that level.

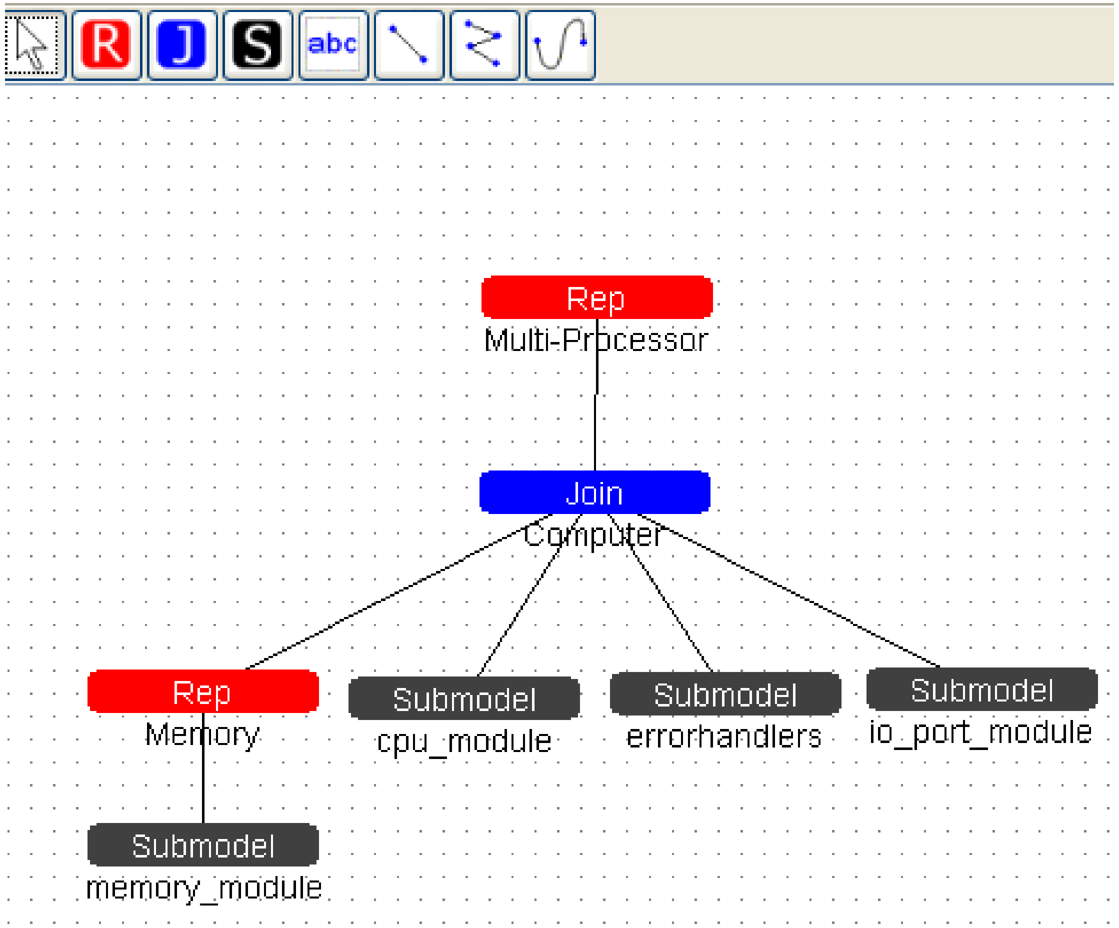
Redundant Component	Fault Coverage Probability
RAM Chip	0.998
Memory Module	0.95
CPU Unit	0.995
I/O Port	0.99
Computer	0.95

Finally, the failure rate of every chip in the system is assumed to be
100 failures per billion hours

Computer Scheme



Composition model of the system



Rep : Repetition of the same submodel

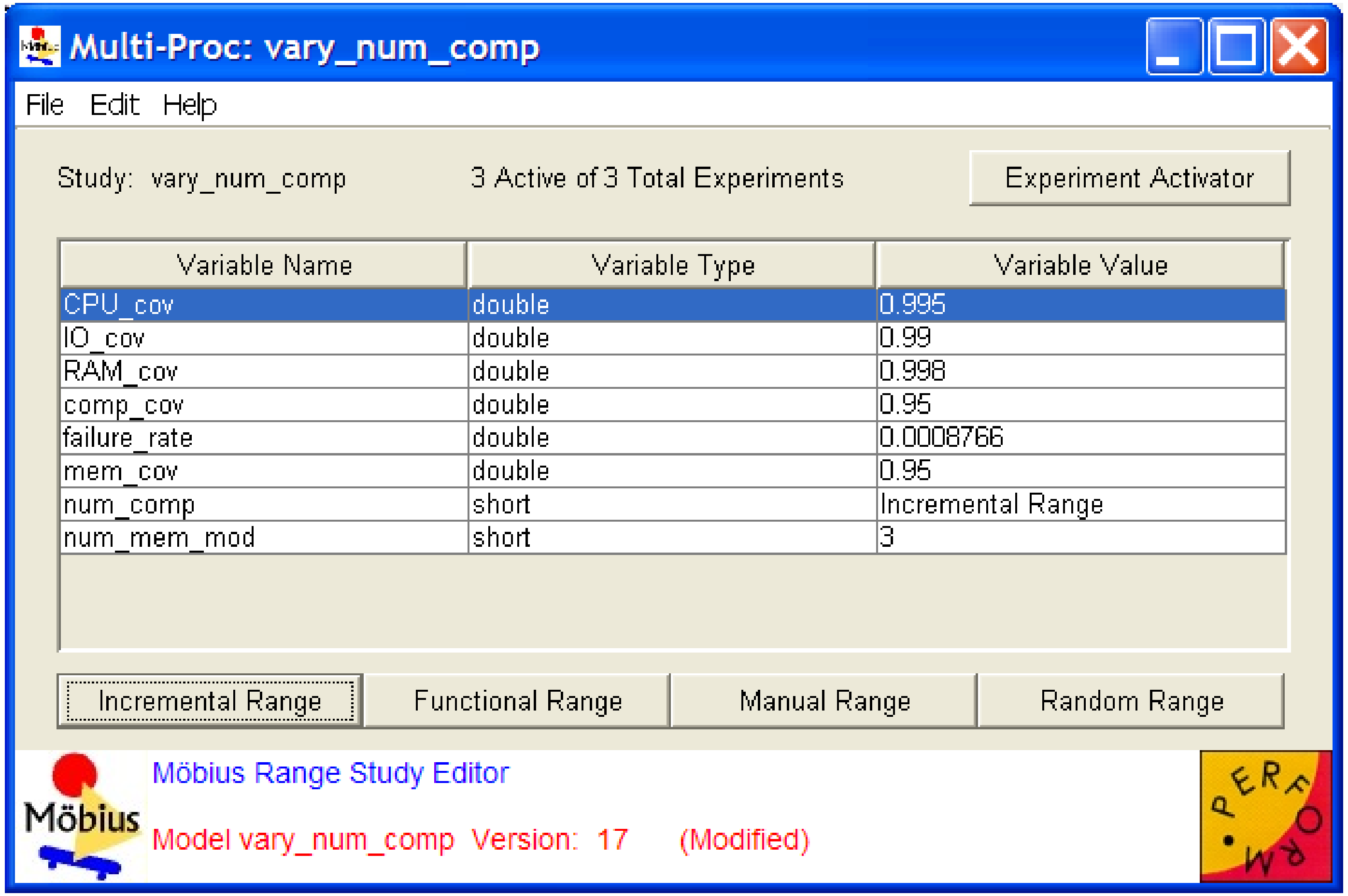
Join : Union of different submodel

A computer is a union of

- 3 different submodels
- the repetition of the memory submodule

The whole system is the repetition of a computer

Study model



Study: vary_num_comp

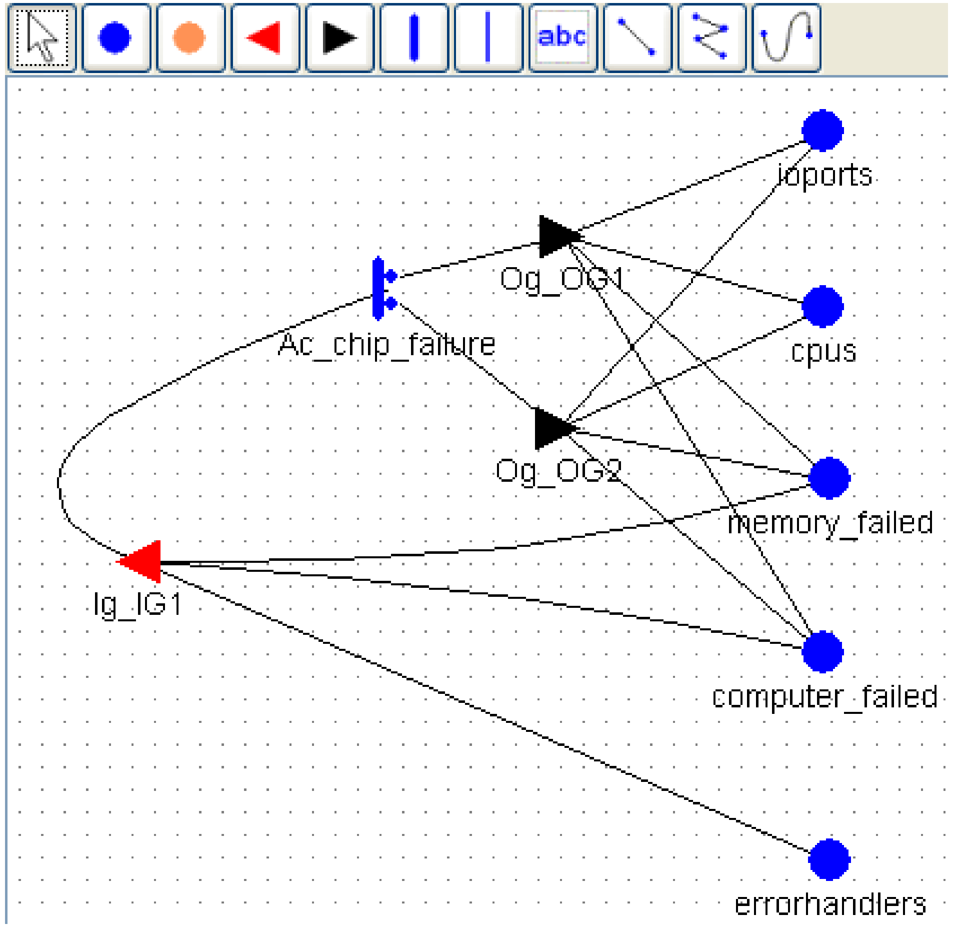
3 Active of 3 Total Experiments

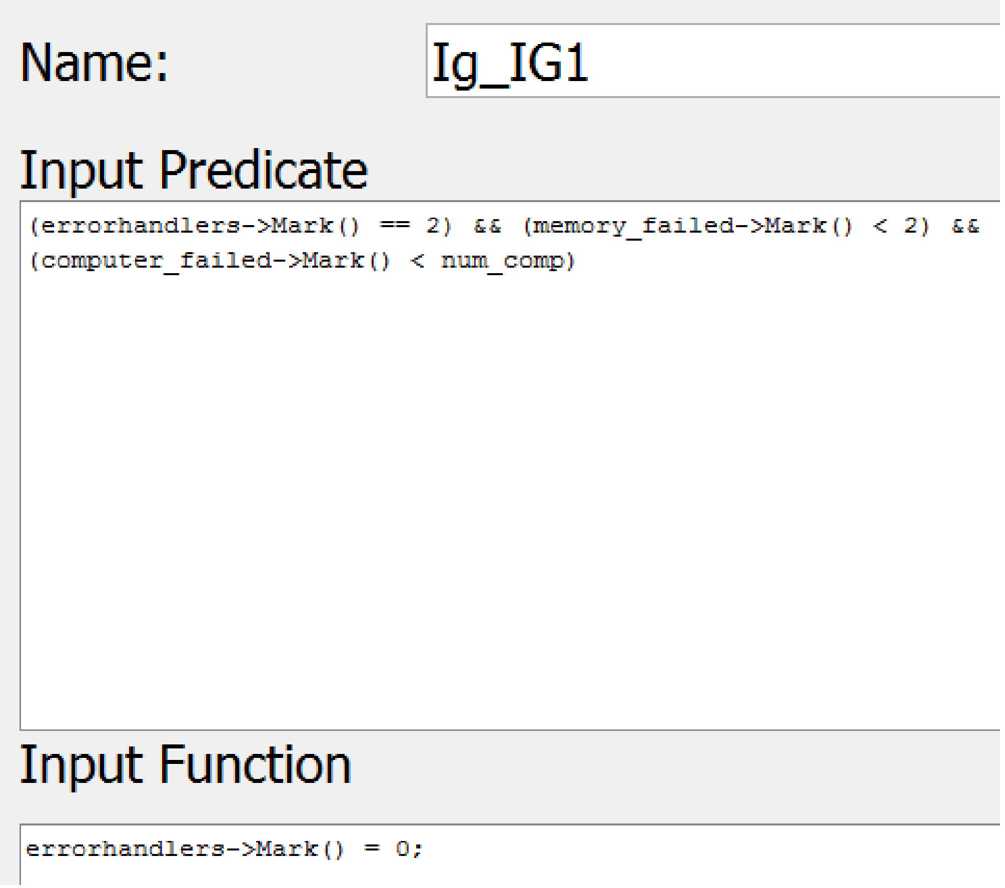
Variable Name	Variable Type	Variable Value
CPU_cov	double	0.995
IO_cov	double	0.99
RAM_cov	double	0.998
comp_cov	double	0.95
failure_rate	double	0.0008766
mem_cov	double	0.95
num_comp	short	Incremental Range
num_mem_mod	short	3

Möbius Range Study Editor

Model vary_num_comp Version: 17 (Modified)

Error handlers Model (Ig_IG1)





Name: Ig_IG1

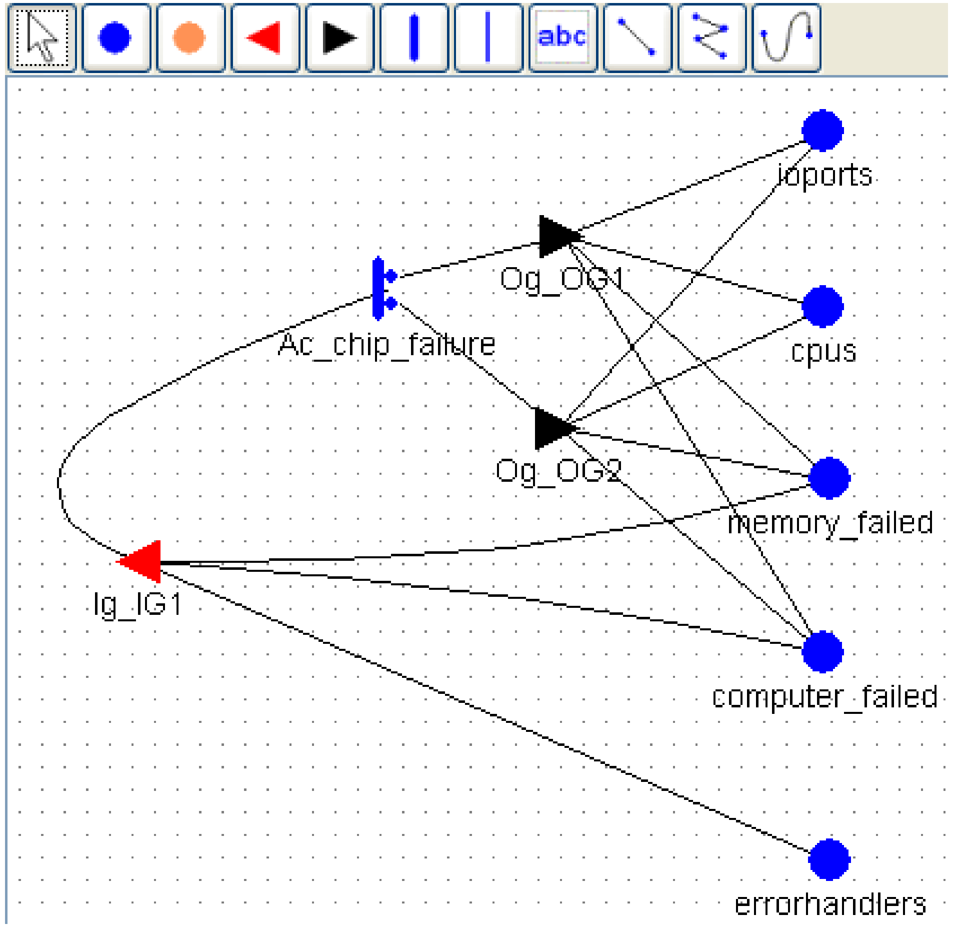
Input Predicate

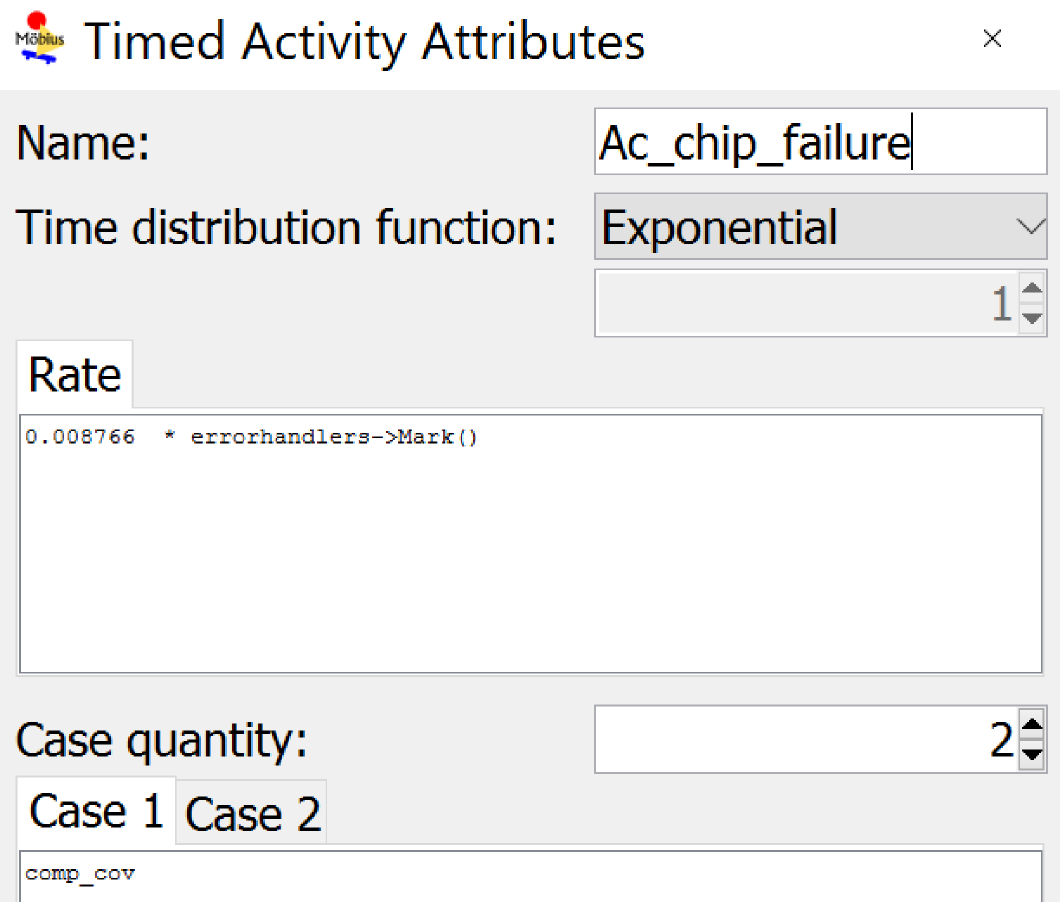
```
(errorhandlers->Mark() == 2) && (memory_failed->Mark() < 2) &&
(computer_failed->Mark() < num_comp)
```

Input Function

```
errorhandlers->Mark() = 0;
```

Error handlers Model(Ac_chip_failure)





Name: Ac_chip_failure

Time distribution function: Exponential

1

Rate

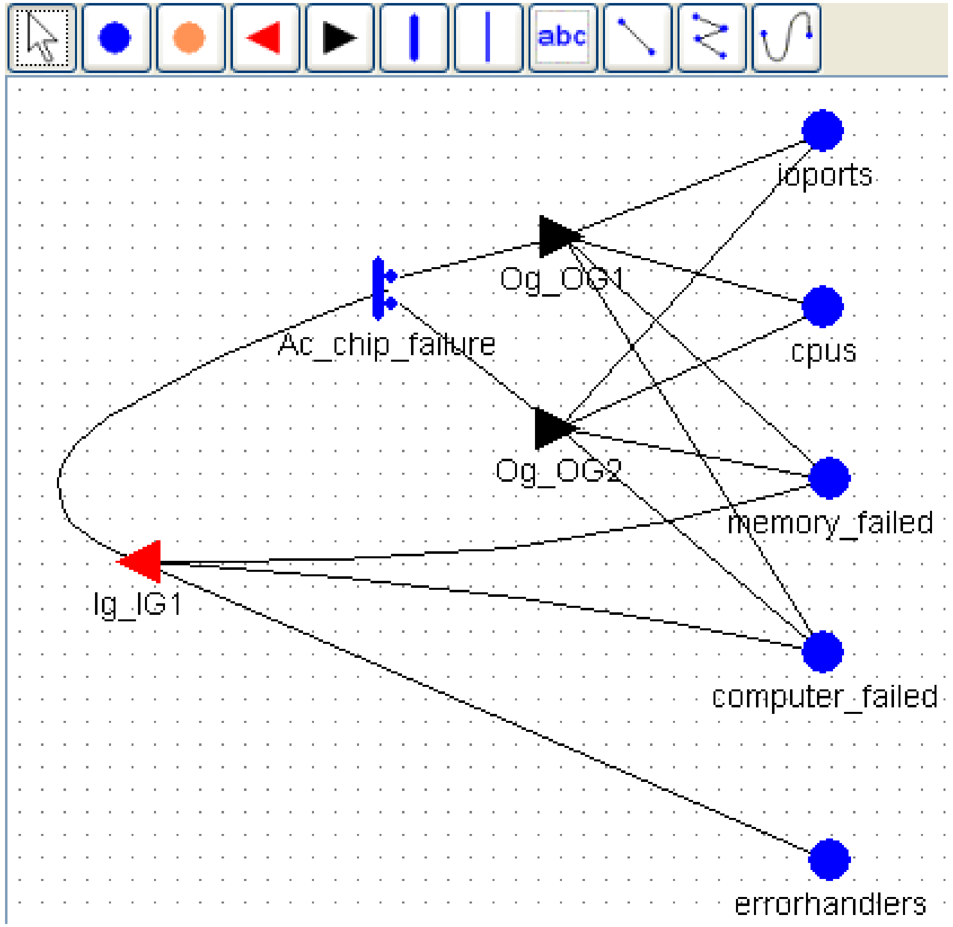
```
0.008766  * errorhandlers->Mark()
```

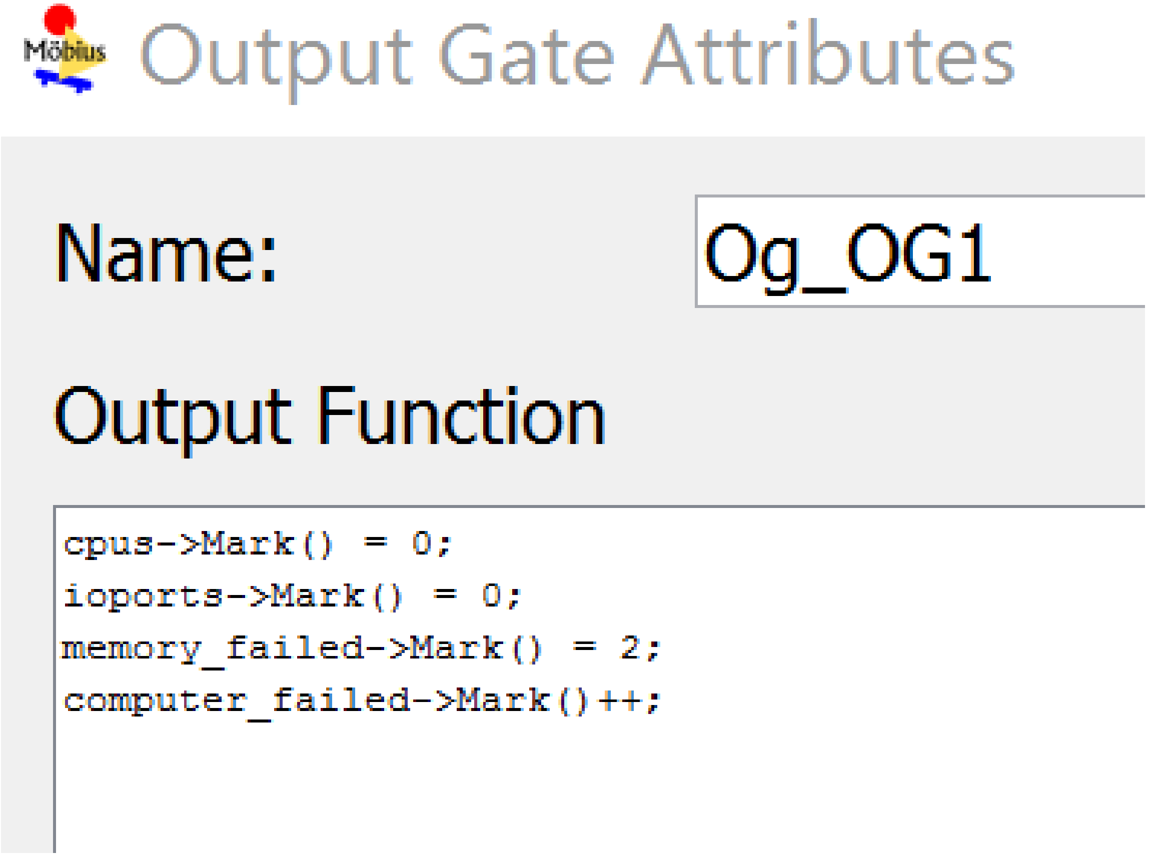
Case quantity: 2

Case 1 Case 2

```
comp_cov
```

Error handlers Model (Og_OG1)





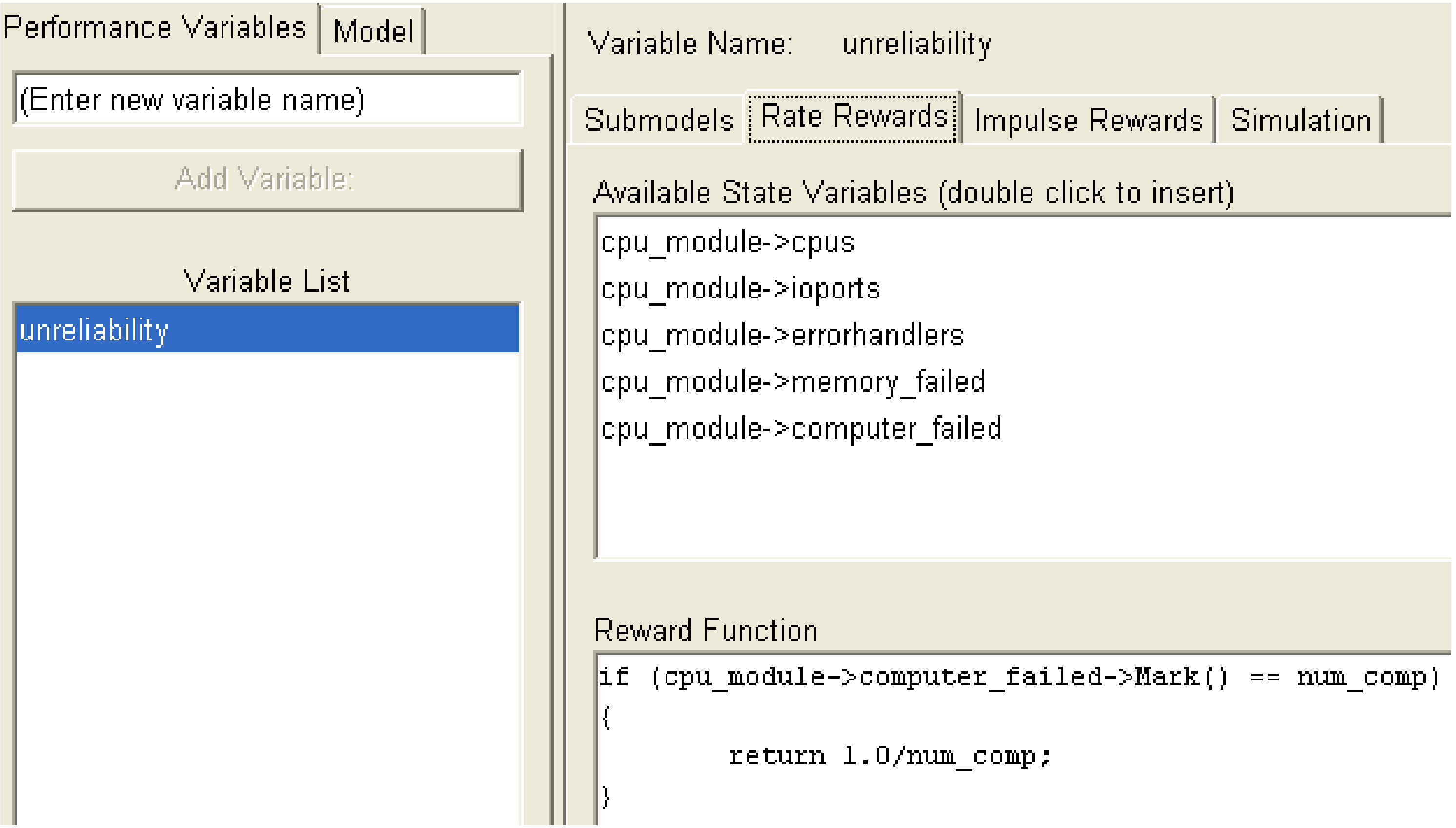
Output Gate Attributes

Name: Og_OG1

Output Function

```
cpus->Mark() = 0;
ioports->Mark() = 0;
memory_failed->Mark() = 2;
computer_failed->Mark()++;
```

Reward Model



unreliability

Variable Name: unreliability

Available State Variables (double click to insert)

cpu_module->cpus
cpu_module->ioports
cpu_module->errorhandlers
cpu_module->memory_failed
cpu_module->computer_failed

Reward Function

```
if (cpu_module->computer_failed->Mark() == num_comp)
{
        return 1.0/num_comp;
}
```

1/num_comp because we are replicating the module num_comp times

Time:
Instant of time
20 years

Solver

The State Space Generator is the Flat State Space Generator.

- De-activate experiment 3 because it will take too much time

And the Solver used is the Transient Solver

- The result for the experiment 2 produced a mean value of 0.01746523

Exercises

1) Create the atomic model of the IO port

2) Create the atomic model of the cpu unit

3) Create the atomic model of the memory module

The whole model is available as an archived model. To open it:

Project -> Unarchive, choose Multi-Proc and hit Unarchive;right click on Multi-Proc and Resave.

Then delete one atomic model at the time and try to obtain the same result with your version.

Remember to fix the error on the errorhandler atomic model

References

https://www.mobius.illinois.edu/wiki/index.php/Fault-Tolerant_Multiprocessor_Model

https://www.mobius.illinois.edu/wiki/index.php/Möbius_Documentation

Thanks to prof. Andrea Domenici for previous version of the slides.