ECE 18-764 Class Project 1
Defect-circuit-layout Relationship

Wing Chiu Tam Fall 2008

1) Background Information

This section gives an overview of IC testing for students without IC testing knowledge so that they can have the necessary background knowledge to understand and do (and hopefully appreciate) this project. Please skip this section if you are already familiar with IC testing.

In IC manufacturing, to ensure high quality of manufactured IC product, testing are applied to all the manufactured IC's to screen out the weak or failing parts. Testing for all possible defects (or deformations) in an IC is economically infeasible. Instead, defects are often abstracted in the logic level and this is called *fault model*. One most common fault model used in industry today is called the *single stuck-line (SSL) model*. In this model, only a single line is assumed to be defective in the circuit and is either permanently stuck at logic one or permanently stuck at logic zero. Testing for stuck-at faults means generating and applying *test vectors* (i.e. inputs to the circuit or sub-circuit) targeting at SSL fault model at carefully selected locations of the circuit and checking the output response to ensure correctness. Particularly, stuck-at test vector generation enforces two conditions: a) the fault location must be driven to opposite polarity. This means that testing stuck-at-one at location **z** requires **z** to be set to logic zero. This is to ensure that an error is generated at the fault location. The second condition is that the erroneous value must propagate to an observable point (typically an output pin or a register in the circuit). A series of test vectors is also called *test patterns* or a *test set*. Test patterns are applied to the manufactured product in a *tester*. The tester collects the responses of the circuit under the applied test patterns. Response that does not agree with the *golden signature* (i.e. the expected response) is called *faulty response* or *fault signature*. Typically, a circuit with faulty response fails for a subset of the test patterns. Patterns for which the IC produces expected responses are called *passing patterns*. Otherwise, they are called *failing patterns*. *Diagnosis* of IC typically analyzes fault signatures with the associated failing patterns in the logic circuit to narrow as much as the possible locations of failure. The output of diagnosis is a set of *suspect site*s, which is a list of nets/nodes in the circuit which might contain the defect. Passing patterns are also used to further prune the suspect sites.

2) Project Objective

Diagnosis of integrated circuits (IC) is an indispensable part of IC manufacturing process for yield improvement. However, this is the case if and only if the cause of failure (i.e., the "defect") is precisely located, exposed, and understood so that the appropriate corrective measures can be taken. In this project, we will be dealing with the first and perhaps the most important step of IC diagnosis: Defect Localization.

Test data are used as the starting point for many diagnosis algorithms. By analyzing the responses of the failed parts to its input test patterns, much can be learned about the potential locations of the defects. Typical defect localization consists of a set of suspect nets that might contain the defect. In this project, we will go a step further and ask ourselves this question: Given a set of suspect failure sites of a circuit and the partially incorrect response of the circuit under given input patterns, what defective circuit(s) could have produced the same faulty response? In other words, it is expected that you "design" (or find) a faulty circuit(s) to satisfy the given response using the defect-free circuit and the list of suspect sites as a starting point.

The ability to extract equivalent circuit level model(s) of the defect behavior is a significant step in understanding of the defect. By corresponding circuit level model of the defect with its layout, it can yield insight on how and where the defect occurs within a given site.

3) Project Description

Figure 3 shows the overview of this project. In this project, a simple ISCAS 85 benchmark[1] circuit c432 is used. This circuit implements a priority decoder and is placed and routed using a 0.18µm, 5-metal-layer TSMC process. To imitate the IC testing environment, SPICE simulation is utilized. The test vectors, together with the modified SPICE netlist (which represents a defective IC caused by a certain defect), are inputs to the SPICE simulator. In each run, the modified SPICE netlist is simulated over all the vectors in the test set. Since the circuit is modified, it is expected to produce a faulty response for some patterns. **Your job is to find suitable modifications of the circuit needed to produce a particular faulty response (which is given to you).** The modified circuit should also agree with expected response for the passing patterns.

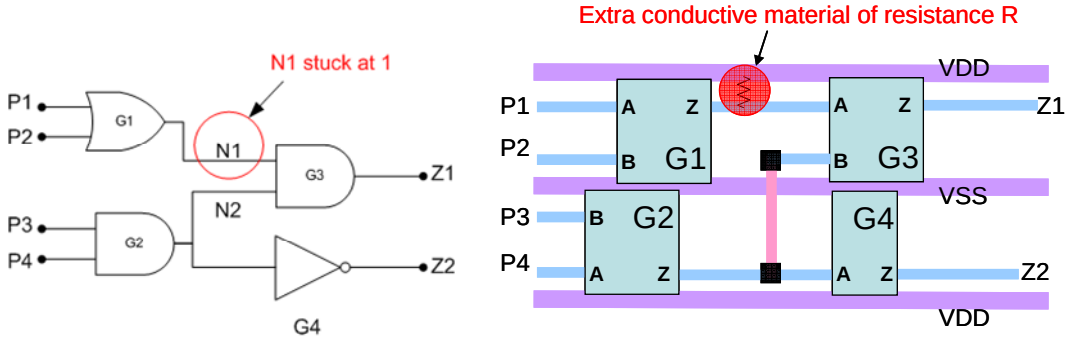

Figure 1. (a) Defect manifested as a stuck-at-1 fault at N1 in logic level. (b) Defect may be caused by additional conductive material from N1 to VDD in the layout.

Figure 1 shows an example. If stuck-at-1 fault at location **`N1`** produces a faulty response that tallies with the given faulty response for all failing patterns and tallies with the expected response for all passing patterns, then additional conductive material of low resistance may have occurred between line **`VDD`** and **`N1`**. This can be modeled by the defect-free circuit with an extra resistor of small resistance R (*e.g.*, 12 Ω) bridging **`N1`** with **`VDD`**. This means that you should modify the defect-free SPICE netlist by adding the following statement:

[1] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits," *Proc. IEEE Int'l Symp. Circuits and Systems*, 1985, pp. 695-698.

```
Rshort N1 VDD 12
```

Clearly, for every valid defective circuit that you might find, a wide range of parameter value is possible (*e.g,* R might be between 0 Ω to 50 Ω). You are expected to find this range of values for which the modified circuit still tallies with the given response for all patterns. Furthermore, it is plausible that more than one circuit topology can be found that yields the same given response. For example, as shown in Figure 2 (b), additional conductive material might have occurred between **`N1`** and **`N2`**. Coincidently, **`N2`** happens to have a logic value of one every time when **`N1`** has a valid propagating path to the output. (Let's assume that **`N2`** is strongly driven such that it is not affected by **`N1`**.) This means that the addition of the following statement to the defect-free SPICE netlist would have produced the same given response.

```
Rshort N1 N2 12
```

This is considered as a **different circuit topology** (*i.e.,* the modified circuit based on Figure 2 (b) is structurally different from the modified circuit based on Figure 1 (b)). You are encouraged to find as many *different* topologies as possible.

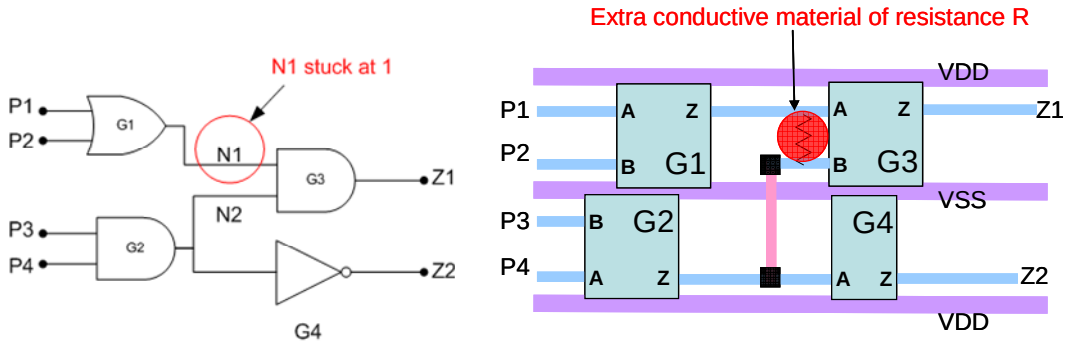

Figure 2. (a) Defect manifested as a stuck-at-1 fault at N1 in logic level. (b) Defect may be caused by additional conductive material from N1 to N2 in the layout.

After finding the defective circuit topology with the range of parameters, you should then look at the layout to conjecture what could have happened in the process to cause this equivalent circuit behavior. (Hint: you might find it helpful to look at the layout first to guide your search for the defective circuit.) In other words, you should explain how the defect/deformation can possibly arise from manufacturing process such that the defective circuit is resulted. In addition, if more than one circuit topology is found, you should comment on the relative likelihood of different topologies.

In this project, these files will be given to you:

- The spice netlist of the c432 circuit (annotated with net names).
- The layout of the c432 in Cadence Encounter Database format.
- The layout of the c432 in Caltech Intermediate Format (CIF)[2].
- The input test patterns of the c432.

[2] More details can be found in this webpage: http://www.rulabinsky.com/cavd/text/chapb.html

- The gate level netlist of the c432 in ISCAS bench format and Verilog format.
- The faulty response of a defective c432 circuit.
- A set of suspect nets that might contain the defect.

The SPICE netlist is annotated such that the nodes in the SPICE bears correspondence to the given logic level netlist. However, the correspondence is not one-to-one. This is because a net in the logic level would be mapped into a network of nodes consisting of the parasitic resistors and capacitors. In other words, the mapping from logic to SPICE is one-to-many. For example, one such mapping is: `N307→{N307_ADDED_NODE_001, N307_ADDED_NODE_002, N307_ADDED_NODE_005, N307_ADDED_NODE_006}`. You can see from this example that the convention for the mapping is:

```
<NetName>→{ <NetName>_ADDED_NODE_<3 digit number> }
```

In addition, a spice simulation framework will be given to you. This framework allows easy addition and removal of components through the use of simple XML syntax. Figure 3 shows an overview of this project.

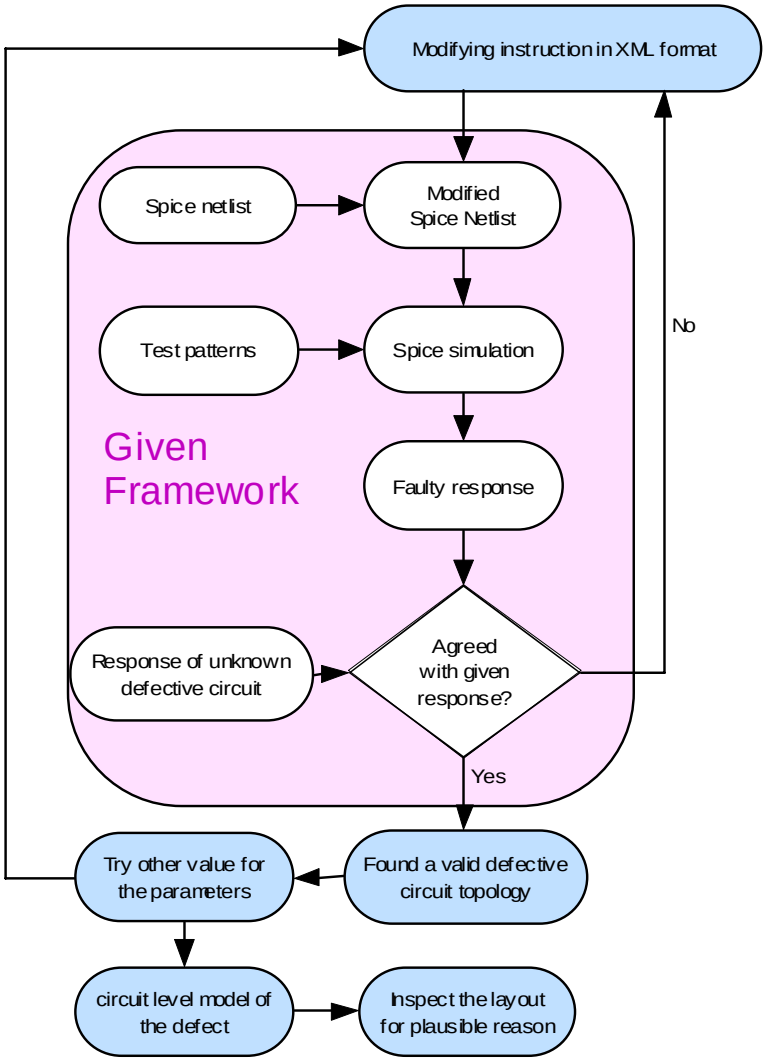

Figure 3. Overview of the project.

3) Project Component

a) Find a defective circuit topology to satisfy given responses for all passing patterns and all failing patterns. The defective circuit topology should be found by modifying the defect-free circuit SPICE netlist (i.e. adding components or removing components or both) using the given framework. You should find as many different circuit topologies as possible. (Two circuit topologies are considered different if they are structurally different, as illustrated in Figure 1 and Figure 2). Submit all the circuit topologies you found in terms of the XML instructions needed to modify the defect-free circuit. If you use a script/program to create the XML file, submit the source code of this script/program as well. Otherwise, indicate that you create this XML file manually in your report.

b) Find the range of parameter for every circuit topologies found in part (a) such that the generated response still agrees with the given response for all patterns.

c) The defective circuit topology with its parameter range is called a circuit level model of the defect. For each of the circuit-level model of the defect that you have found, analyze the given c432 layout to conjecture what could have gone wrong in the processing step. Give reasons to support your conjecture based on what you have learnt in the lectures.

d) If more than one circuit topology is found in part (a), comment on the relative likelihood of the different topologies. If you think one topology is more likely than the other, give reasons.

e) A written report with no more than 6 pages of text and unlimited number of drawings, with

   1. Description of the approaches you have adopted to find the circuit level model of the defect. (e.g., do you find it by trial-and-error manually? Do you do an exhaustive search? Do you intelligently analyze the search space and prune it?)
   2. Pictures of all the defective circuit topologies that you can find, showing the parameters and its range of possible values. (Do not show the entire c432 circuit, just the circuit of the localized defective site.)
   3. The conjecture you make in (c) and its reasons. Show a portion of the layout side by side with the defective circuit topology. (Again do not show the entire circuit or entire layout, just the relevant portion.)
   4. The comment and its reasons in (d), if you are able to find more one circuit topology.
   5. Any finding that you find interesting.

What you should NOT do: The defective circuit you "designed" (or found) must bear sufficient resemblance to the original defect-free circuit. Sufficient resemblance means that you should be able to establish a one-to-one mapping of components from the defect-free circuit to the defective circuit you designed for most of the components in the design. Specifically, there should not be more than 20 differences between your defective circuit and the defect-free circuit. This means that you should NOT derive logic equations based on the given faulty response and the circuit inputs and use the derived equations to completely re-synthesize the gate level netlist! However, it is okay for you to derive logic

equations as an aid to help yourself figure out what modifications might be needed to craft the desired defective circuit.

4) Project Instruction

a) Project directory is: `/afs/ece/class/ece764/764_2008/project`. This is denoted as `<proj_dir>` below.

b) Getting and checking the simulation framework. We provide a simulation platform that automates multiple SPICE modification and simulation as well as the extraction of the response. Copy the following directories to your working directory (henceforth referred to as `<your_dir>`.) There are symbolic links between these directories, so please do NOT alter their relative positions.

- `<proj_dir>/bin` `# all scripts reside here`
- `<proj_dir>/results` `# run your simulation here`
- `<proj_dir>/lib` `# library directory for scripts`
- `<proj_dir>/data` `# data needed for this project`
- `<proj_dir>/response` `# response of unknown defective circuit`

Go to the directory `<proj_dir>/results/my_defect_1/run_common`. You will see the files required for SPICE simulation:

- `c432.bench` `# logic description of c432 in bench`
- `c432.v` `# logic description of c432 in verilog`
- `c432_flat.spice.orig` `# defect-free netlist`
- `cmos_0p18.spice` `# MOSIS models for TSMC's 0.18um process`
- `c432.test` `# test set`
- `fault_list.xml` `# example of file you need to generate`
- `c432.inc` `# a wrapper around flat spice netlist`

`fault_list.xml` is a sample file providing information about how the parser is going to modify the SPICE netlist. `fault_list.xml` **is the file that you should generate (manually or automatically, it is your choice)**. Appendix shows two examples.

The underlined parts in the examples identify the description of the defect. They won't be used to modify SPICE but they should provide enough information about the defect. The bold-faced parts are important to the parser. “deformation-list length” specifies how many deformations are there in this file (include all defective circuit topology in this XML). “rank” is the unique ID (starting from 1) for each defect. Within the <modify> and </modify> labels, the "<add>" labels tell the parser what you are going to add to the original SPICE netlist, and the "<remove element>" labels tell the parser what elements are going to be removed.

To check whether the modified SPICE netlist is what you expected, run the following commands:

```
% cp c432.spi.orig c432.spi
% <your_dir>/bin/find_deform.py fault_list.xml rank=1 current_dir
current_dir
```

(The repeated current_dir options have different purposes in the whole simulation environment. Here, for generating the modified SPICE, just do it as shown above.)

```
% <your_dir>/bin/modify_spice cut.xml
```

The script `find_deform.py` extract the defect based on the conditions you specified (rank=l) and write to file `cut.xml`. The script `modify_spice` then modifies the `c432_flat.spice` based on `cut.xml`. Compare the generated `c432_flat.spice` and the original `c432_flat.spice.orig`.

c) Running simulation using the given framework. After finishing your fault_list.xml file, do the following steps:

```
% cd <your_dir>/result
% mkdir <your_defect>          # choose your own name
% cp -r my_defect_l/run_common  <your_defect>/run-common
```

Replace `<your_defect>/run_common/fault_list.xml` with the one you created Go back to `<your_dir>/results/`.

```
% ../bin/run_all <your_defect>
```

This will start the SPICE modification and simulation process. For each defective circuit topology you specify, the script will create a temporary working directory under `/scratch/your_andrew_id/run_exec`. When the simulation is complete, the temporary directory will be cleaned up automatically and you will see directories corresponding to each defect in `./results/<your_defect>`, e.g. `run_0001`, `run_0002`, …, etc. where the number matches the rank specified in the `fault_list.xml`. The `c432_response.xml.gz` in each defect directory contains the extracted simulation results for all the test vectors. You can unzip this file with the command:

```
% gunzip c432_response.xml.gz
```

Check `<result *>` line for each vector. This line contains the simulated tester response alongside with the expected response, where "expected" ≡ expected golden signature and "actual" ≡ simulated output response. The expected and actual logic values of outputs should be the same for a defect-free circuit. The given response of the unknown defective circuit is specified in exactly the same format (with the details of the defect omitted) and is given to you in the path `<proj_dir>/response/unknown.xml`. To make your life easier, the list of suspect sites which **might** contain the defect is given and its path is `<proj_dir>/response/suspect`. You should find defect circuit level model that agrees with the file `unknown.xml` for all patterns using the `suspect` file as a starting

point. For your convenience, a simple script has already been written for you and you can use it like this:

```
%../bin/compareResponse.pl -g <unknown.xml> -r <c432_response.xml.gz>
```

Do a "`../bin/compareResponse.pl -h`" will show you how to run the script.
For this project, you should **ignore the fields "iddq" and "delay" in the response**.

Hint: Add "nohup" to your command and put it to background if you want to run you job overnight. For example:

```
% nohup ../bin/run_all <your_defect> &
```

This allows you to log out of the machines without stopping any simulation.

d) Analyzing the suspect sites. The suspect sites in the following format.

```
Description:    extra material defect (6 AM)
Size:           less than 0.72 um
Nodes involved: N13, N14, N15
```

Figure 4 shows the corresponding picture at the layout level. The size of the defect specifies the radius of the minimum enclosing circle of the defect as shown in the Figure 4. The field "`nodes involved`" tell you the names of the net affected by the defect in the logic level netlist. Note that this list of nodes may or may not physically touch the defect. You should bear this mind when finding the defective circuit.

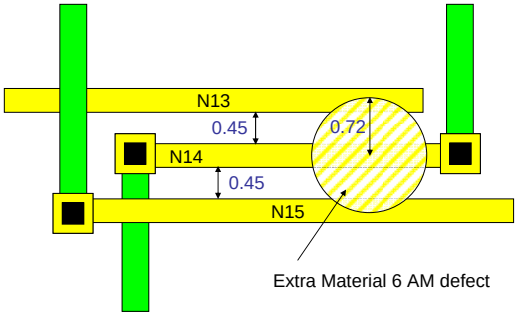

Figure 4. Illustration of the suspect site.

e) Viewing the project layout. You should visually analyze the given layout to understand the defect circuit level model and to perform the required project task.

The layout is in this path `<your_dir>/data/layout/encounter/c432`. You can bring up Cadence Encounter by using the following command:

```
% cd <your_dir>/data/layout/encounter/c432
% source encounter_4.2.bash
% encounter
```

After you have Encounter window opened, select "Design→Restore Design…", a window will pop up, enter the name `c432.filled.enc` and click "open". The default view is "floorplan" view. Click on the "physical view" button located near

the top right corner of the layout canvas. The design is now ready for layout viewing.

Unfortunately, the encounter program does not show the polysilicon layer, c432.cif file is thus also provided for you. And a simple CIF viewer is also provided to view this c432.cif graphically. The CIF layout is in this path `<your_dir>/data/layout/c432.cif`. You can view the layout by using the following command:

```
% <your_dir>/bin/layout_view c432_flat.cif
```

5) Project Deadline and Deliverables

Project 1 Deadline: 11/20/2008

Deliverables for project 1:

a) A written report. (Refer to section 3 (e) for more details)
b) All the circuit topologies that can produce the given response in one XML file, name this file `fault_list.xml`. For given topology, a wide range of parameters is possible, you can set the parameters in the XML to any number in the valid range in the `fault_list.xml`.
c) Any scripts that you wrote to generate the file `fault_list.xml`. Include a README file on how to run your script. If you generate this XML file manually, indicate it in the report.
d) Submit these files to this directory:
`/afs/ece/class/ece764/764_2007/project/submission/group-no/proj1`

NOTE: It is important that you finish project 1 on time because project 2 will be based on the work you have done in project 1. Without project 1 completed, you will not be able to proceed to project 2. More details on project 2 will be given out later.

6) Project Grading

- Project 1: 70 % The TA will simulate the file `fault_list.xml` to verify your work. Submitting `fault_list.xml` that incorrectly/partially produces the given response will result in loss of credit. More weight is given to failing patterns than to passing patterns, i.e. if your circuit level model does not agree with a faulty response, then you will be penalized more than if it does not agree with an expected response. Your `fault_list.xml` should contain at least 1 topology. You can get more credit if you can come up with substantially different topologies that produce the given response.
- Late submission policy: For project 1, you will be penalized 10% per day and up to 6 days after the deadline. If you submit after 6 days you would receive 0% credit.
- Project 2: 30 %. More details will be given later.

7) Appendix

Example of fault_list.xml

```
<?xml version="1.0"?>
<deformation_list length="2">
  <deformation class="metal short" id="1" rank="1">
    <description>
      Short in metal connects signals N165, N203
    </description>
    <modeling>
      <modify>
      <add> .param Rshort="10" </add>
      <add> Rinjected0 N165_ADDED_NODE_008 N203_ADDED_NODE_067 Rshort </add>
      </modify>
    </modeling>
  </deformation>
  <deformation class="class="transistor" id="2" rank="2">
    <description>
      Complex defect involving transistor
    </description>
    <modeling>
      <modify>
        <add> .param Rshort="10" </add>
        <remove element="M3293"/>
        <add> M3293 92 N351_ADDED_NODE_001 507 92 tsmc18dP L=180e-9 W=269e-7
AD=117e-15 AS=89e-15 PD=1.03e-6 PS=900e-9 M=1 </add>
        <add> Rinjected0 N351_ADDED_NODE_001 507 Rshort </add>
      </modify>
    </modeling>
  </deformation>
</deformation_list>
```

Example of c432_response.xml

```
<CUT run="run_0001" dir="my_defect_1">
  <deformation class="metal short" id="vdd,gnd" rank="1">
    <description>
      Short in metal connects signals N165, N203.
    </description>
    <modeling>
      <modify>
      <add> .param Rshort="10" </add>
      <add> Rinjected0 N165_ADDED_NODE_008 N203_ADDED_NODE_067 Rshort </add>
      </modify>
    </modeling>
  </deformation>
  <response corner="nominal" vdd="1.8V" period="10.0" temperature="25">
  <pins output="n223,n329,n370,n421,n430,n431,n432"/>
  <result vector="1" expected="HLHHHHL" actual="HLHHHHL" delay="2.4943ns"
iddq="failed"/>
  <result vector="2" expected="HLHHHHH" actual="HLHHHHH" delay="0.57ns"
iddq="failed"/>
  <result vector="3" expected="HHLLLLL" actual="HHLLLLL" delay="1.875ns"
iddq="failed"/>
   …
```

```
    …
    …
</response>
</CUT>
```