گروه سخت افزار
دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف

پروژه درس آزمون پذیری

پیاده سازی یک سیستم تولید تست

استاد: دکتر شاهین حسابی

دانشجو: وحید مواجی

نیمسال اول ۸۴ - ۱۳۸۳

نیازمندی های پروژه

نیازمندی های این پروژه به شرح زیر می باشد:

۱. توصیف ساختاری یک مدار به فرمت VHDL یا Verilog که منطق سه مقداره 0, 1, X و مدار ترکیبی را پشتیبانی نماید.

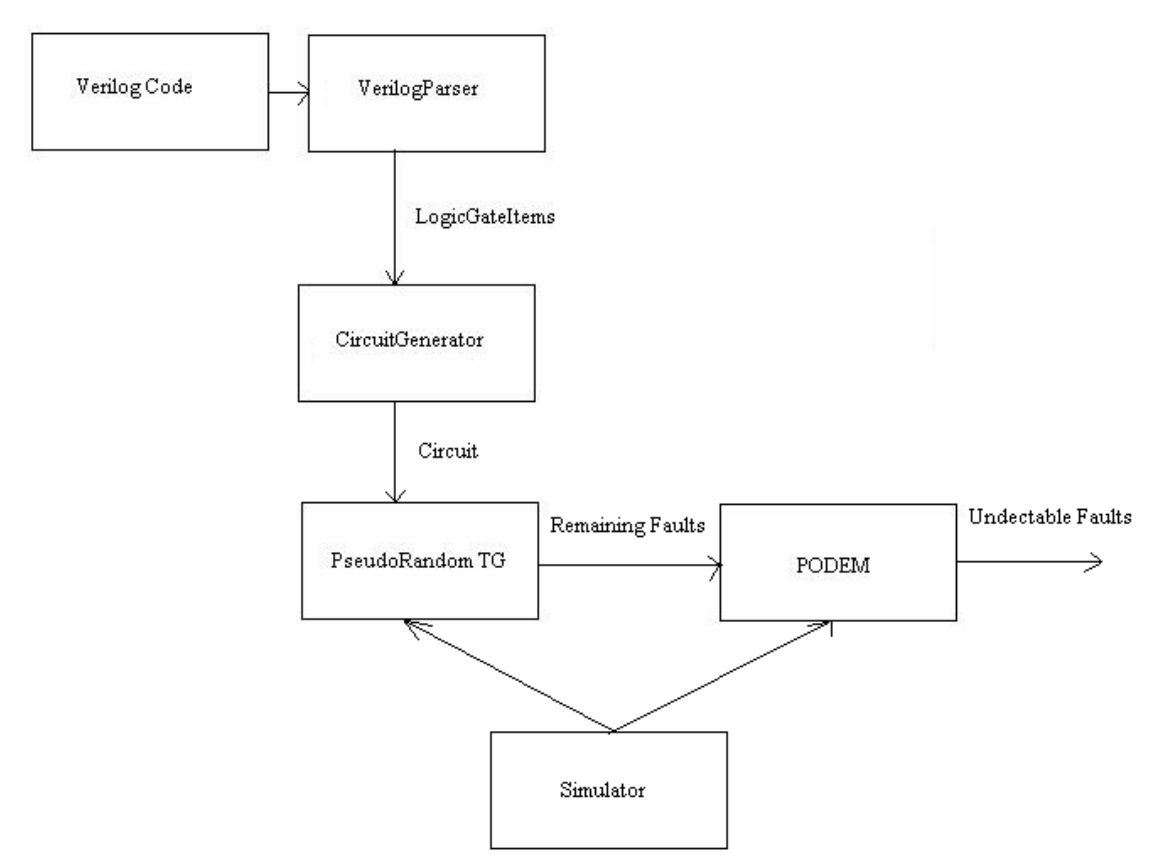
۲. یک شبیه ساز منطقی (Logic Simulator) که شبیه سازی تأخیر را پشتیبانی نماید ← Unit delay, rise, fall.

۳. دارای یک سیستم ATPG که جزئیات آن بصورت زیر باشد :

- ابتدا fault های مدار را collapse نماید و سپس روی این fault های collapse شده، کار کند.
- با روش شبه تصادفی (Pseudo Random) یک بردار تست ایجاد نماید.
- سپس توسط یکی از روشهای fault simulation مثلا parallel, deductive, concurrent مدار را شبیه سازی کرده و fault های تشخیص داده شده را از لیست حذف کند و درجایی ثبت شود که این fault با کدام بردار تست، حذف شده است.
- تا زمانیکه fault coverage <= 50% - 60% می باشد، مراحل بالا باید تکرار شوند.
- اگر همه fault ها تشخیص داده شده باشند، الگوریتم پایان می یابد.
- با یکی از روشهای مبتنی بر fault مثل D, PODEM یک بردار تست برای یکی از fault های کشف نشده تولید نماید.
- شبیه سازی fault را انجام داده و fault های دیگری که ممکن است با این بردار کشف شوند را بدست آورده و از لیست حذف نماید.
- اگر fault coverage >= 99% شود یا timeout اتفاق بیفتد، باید اعلام شود که fault، قابل تشخیص نیست.
- در انتها باید نتایج گزارش داده شوند← کدام fault با کدام بردار تشخیص داده شده است و کدام fault قابل تشخیص نیست.

ما در این گزارش سعی می کنیم بصورت گام به گام با توضیح قسمت های مختلف برنامه نحوه پیاده سازی خود را بیان نماییم. هر جا که لازم بوده است لیستی از اعضا یا توابع کلاس مورد نظرآورده شده است تا عملکرد آن کلاس را واضحتر بیان کرده باشیم.

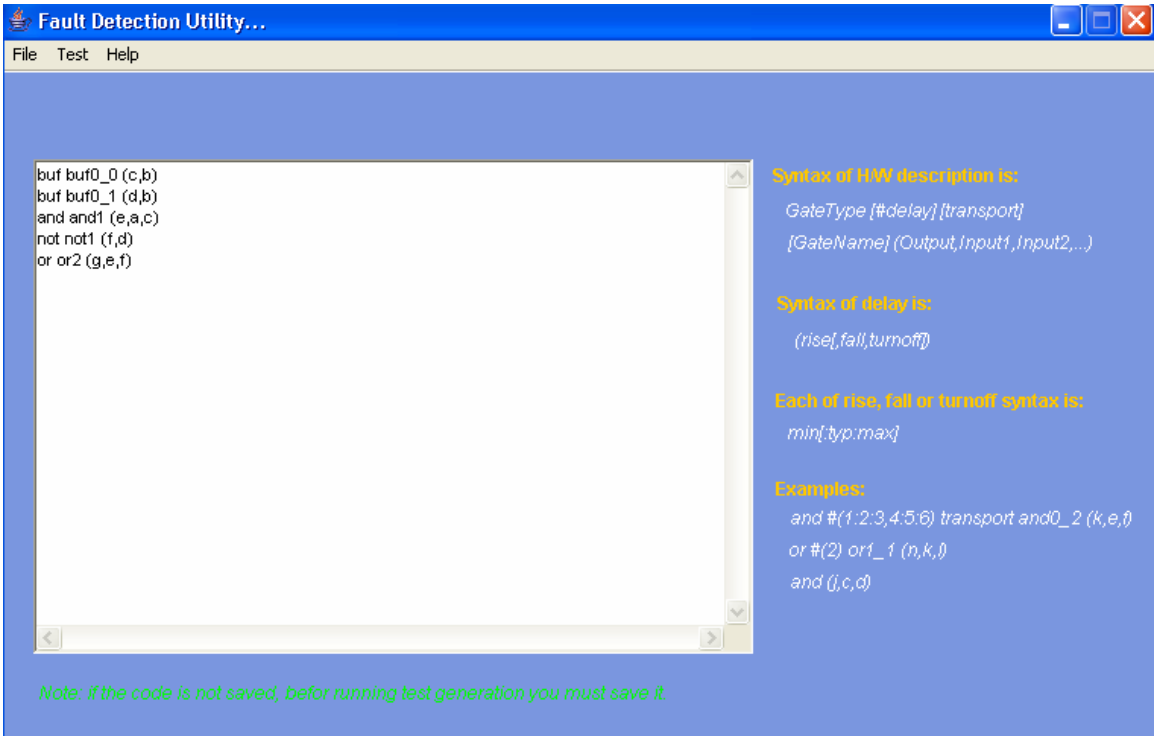
بطور خلاصه عملکرد اجزا مختلف برنامه بدین صورت است:



در نهایت، گزارش کاملی از تمام مراحل مختلف اجرای برنامه به فرمت xml در یک فایل ریخته می شود.

نمونه ای از اجرای برنامه را در این جا می آوریم:

ابتدا از منوی File → Load کد مورد نظر خود را به برنامه می آوریم.



مدار داده شده به برنامه

سپس با استفاده از گزینهDo Test Generation → Test ، خروجی زیر حاصل می گردد:

```
<test_generation>
        <circuit>
                <level_0>
                        <logic_gate type="buf" name="buf0_0" output="c">
                                <inputs>
                                        (b)
                                </inputs>
                                <delay type="inertial" average="0">
                                        <rise average="0"/>
                                        <fall average="0"/>
                                        <turn_off average="0"/>
                                </delay>
                        </logic_gate>
                        <logic_gate type="buf" name="buf0_1" output="d">
                                <inputs>
                                        (b)
                                </inputs>
                                <delay type="inertial" average="0">
                                        <rise average="0"/>
                                        <fall average="0"/>
                                        <turn_off average="0"/>
                                </delay>
                        </logic_gate>
```

```
                </level_0>
                <level_1>
                        <logic_gate type="and" name="and1" output="e">
                                <inputs>
                                        (a, c)
                                </inputs>
                                <delay type="inertial" average="0">
                                        <rise average="0"/>
                                        <fall average="0"/>
                                        <turn_off average="0"/>
                                </delay>
                        </logic_gate>
                        <logic_gate type="not" name="not1" output="f">
                                <inputs>
                                        (d)
                                </inputs>
                                <delay type="inertial" average="0">
                                        <rise average="0"/>
                                        <fall average="0"/>
                                        <turn_off average="0"/>
                                </delay>
                        </logic_gate>
                </level_1>
                <level_2>
                        <logic_gate type="or" name="or2" output="g">
                                <inputs>
                                        (e, f)
                                </inputs>
                                <delay type="inertial" average="0">
                                        <rise average="0"/>
                                        <fall average="0"/>
                                        <turn_off average="0"/>
                                </delay>
                        </logic_gate>
                </level_2>
                <all_faults size="14">
                        [c@0, c@1, b@0, b@1, d@0, d@1, e@0, e@1, a@0, a@1, f@0,
f@1, g@0, g@1]
                </all_faults>
                <collapsed_faults size="7">
                        [c@1, b@0, b@1, d@0, d@1, a@1, a@0]
                </collapsed_faults>
        </circuit>
        <pseudo_random_test_generation fault_coverage_required="0.53">
                <random_test_vector>
                        [b=0 a=1]
```

<events>
<event wire="b" time="0" value="0">
<deduction>
[b@1]
</deduction>
</event>
<event wire="a" time="0" value="1">
<deduction>
[a@0]
</deduction>
</event>
<event wire="c" time="0" value="0">
<deduction>
[b@1, c@1]
</deduction>
</event>
<event wire="d" time="0" value="0">
<deduction>
[b@1, d@1]
</deduction>
</event>
<event wire="e" time="0" value="0">
<deduction>
[b@1, c@1]
</deduction>
</event>
<event wire="f" time="0" value="1">
<deduction>
[b@1, d@1]
</deduction>
</event>
<event wire="g" time="0" value="1">
<deduction>
[d@1]
</deduction>
</event>
</events>
<detected_faults>
[d@1]
</detected_faults>
<fault_coverage value="0.14285714285714285" />
</random_test_vector>
<random_test_vector>
[b=1 a=1]
<events>
<event wire="b" time="0" value="1">

```
                        <deduction>
                                [b@0]
                        </deduction>
                </event>
                <event wire="a" time="0" value="1">
                        <deduction>
                                [a@0]
                        </deduction>
                </event>
                <event wire="c" time="0" value="1">
                        <deduction>
                                [b@0]
                        </deduction>
                </event>
                <event wire="d" time="0" value="1">
                        <deduction>
                                [b@0, d@0]
                        </deduction>
                </event>
                <event wire="e" time="0" value="1">
                        <deduction>
                                [a@0, b@0]
                        </deduction>
                </event>
                <event wire="f" time="0" value="0">
                        <deduction>
                                [b@0, d@0]
                        </deduction>
                </event>
                <event wire="g" time="0" value="1">
                        <deduction>
                                [a@0]
                        </deduction>
                </event>
        </events>
        <detected_faults>
                [a@0]
        </detected_faults>
        <fault_coverage value="0.2857142857142857" />
</random_test_vector>
<random_test_vector>
        [b=0 a=0]
        <events>
                <event wire="b" time="0" value="0">
                        <deduction>
                                [b@1]
```

```
                </deduction>
            </event>
            <event wire="a" time="0" value="0">
                <deduction>
                    [a@1]
                </deduction>
            </event>
            <event wire="c" time="0" value="0">
                <deduction>
                    [b@1, c@1]
                </deduction>
            </event>
            <event wire="d" time="0" value="0">
                <deduction>
                    [b@1, d@1]
                </deduction>
            </event>
            <event wire="e" time="0" value="0">
            </event>
            <event wire="f" time="0" value="1">
                <deduction>
                    [b@1, d@1]
                </deduction>
            </event>
            <event wire="g" time="0" value="1">
                <deduction>
                    [b@1, d@1]
                </deduction>
            </event>
        </events>
        <detected_faults>
            [b@1, d@1]
        </detected_faults>
        <fault_coverage value="0.42857142857142855" />
</random_test_vector>
<random_test_vector>
        [b=1 a=0]
        <events>
            <event wire="b" time="0" value="1">
                <deduction>
                    [b@0]
                </deduction>
            </event>
            <event wire="a" time="0" value="0">
                <deduction>
                    [a@1]
```

```
                        </deduction>
                    </event>
                    <event wire="c" time="0" value="1">
                        <deduction>
                            [b@0]
                        </deduction>
                    </event>
                    <event wire="d" time="0" value="1">
                        <deduction>
                            [b@0, d@0]
                        </deduction>
                    </event>
                    <event wire="e" time="0" value="0">
                        <deduction>
                            [a@1]
                        </deduction>
                    </event>
                    <event wire="f" time="0" value="0">
                        <deduction>
                            [b@0, d@0]
                        </deduction>
                    </event>
                    <event wire="g" time="0" value="0">
                        <deduction>
                            [a@1, b@0, d@0]
                        </deduction>
                    </event>
                </events>
                <detected_faults>
                    [a@1, b@0, d@0]
                </detected_faults>
                <fault_coverage value="0.8571428571428571" />
            </random_test_vector>
            <remaining_faults size="1">
                [c@1]
            </remaining_faults>
        </pseudo_random_test_generation>
        <PODEM>
            <fault name="c@1">
                No test can be generated
                <fault_coverage value="0.8571428571428571" />
            </fault>
        </PODEM>
</test_generation>
```

خلاصه ای از کلاس های مهم برنامه و تشریح عملکرد آنها

util.parser
Interface Keywords

تمام کلمات کلیدی زبان Verilog را در خود نگهداری می کند.

util.parser
Class VerilogParser

```
java.lang.Object
  └util.parser.VerilogParser
```

از این کلاس برای تجزیه (parse) یک کد شبه Verilog استفاده می شود. Syntax این زبان به صورت زیر است:

GateType [#delay] [<transport>] [GateName] (Output,Input1,Input2,…)

delay ::= (rise[,fall,turnoff])

rise ::= min[:typ:max]

مثلا خطوط توصیف زیر همگی معتبر هستند:

and (c,a,b)
and #(2) (c,a,b)
and #(2:3:6) a1(c,a,b)
and #(2:4,4:5:6,1) (c,a,b)

خلاصه اعضا	
private java.util.Vector	**events** لیست همه رویداد های اولیه که بعنوان نتیجه تجزیه کد برگردانده می شود.
private java.util.Vector	**logicGateItems** لیست تمام گیت های منطقی که بعنوان نتیجه تجزیه کد برگردانده می شود.
private java.io.BufferedReader	**verilogBufferedReader** هر بار یک خط کامل از کد را می خواند.
private java.lang.String	**verilogFileName** مسیر فایلی که کد Verilog را در خود دارد.

util

Interface Symbols

تمام سمبل هایی (کاراکترهایی) که ممکن است درطی تجزیه کد یا دیگر قسمت های برنامه مورد نیاز باشند را در خود نگهداری می کند. سمبل ها بر اساس حروف الفبا مرتب شده اند.

util

Class CircuitGenerator

```
java.lang.Object
  └util.CircuitGenerator
```

این کلاس از روی یک netlist که همان کد مدار باشد، کل مدار را می سازد. کد اولیه می تواند به هر ترتیبی وارد شود یعنی ترتیب نوشتن گیت ها تأثیری در مدار نهایی نخواهد داشت، چون این کلاس، مدار را سطح بندی کرده(levelize) و بصورت یک لیست دو بعدی بر می گرداند. در بعد اول، عنصر i ام لیست، خود یک لیست (که در حقیقت همان بعد دوم می باشد) شامل عناصر سطح i ام است.

خلاصه اعضا	
private java.util.Vector	**levels** بعد اول لیست دوبعدی که عنصر i ام آن، عناصر سطح i ام مدار را درخود دارد.
private AbstractLogicGate	**logicGate** گیت منطقی جاری که ساخته شده است و سطح آن قرار است محاسبه شود.
private java.util.Vector	**logicGateItems** داده های ورودی که سطوح مدار بر اساس آن تولید می شوند.
private java.util.Vector	**logicGates** لیست گیت های منطقی که بر اساس سطح گیت های منطقی مرتب شده است و با خواند هر ورودی، مقدار آن update می شود.

util

Class LogicGateItem

```
java.lang.Object
  └util.LogicGateItem
```

این کلاس، اطلاعات خام یک گیت شامل نوع، تأخیر، اسم، ورودی و خروجی های آنرا در خود نگهداری می کند. این اطلاعت باید بعدا به گیتهای واقعی متناظر، نگاشته شوند.

خلاصه اعضا	
private GateDelay	**delay** مقادیر تأخیری که به این گیت منطقی نسبت داده شده است.
private java.util.Vector	**inputs** ورودی های این گیت منطقی.
private java.lang.String	**name** اسم گیت.
private java.lang.String	**output** خروجی گیت منطقی.
private java.lang.String	**type** نوع این گیت منطقی.

simulation

Class Simulator

```
java.lang.Object
  └simulation.Simulator
```

این کلاس، یک شبیه ساز منطقی است که می تواند شبیه سازی معمولی و شبیه سازی fault را انجام دهد و همچنین می تواند بطور مستقل برای شبیه سازی وقتی که رویدادهای مدار در کد اولیه مشخص شده اند، استفاده شود، هرچند در این پروژه، این کلاس ورودی خود را از سیستم تولید تست می گیرد. در این پروژه برای شبیه سازی از منطق ۹ مقداره استفاده کرده ایم؛ هرچند این ۹ مقدار برای حالت های گذرای سیستم استفاده شده و در نهایت از همان منطق ۵ مقداره استفاده می شود.

circuit.logicelements.logicgates

Class AbstractLogicGate

```
java.lang.Object
  └circuit.logicelements.logicgates.AbstractLogicGate
```

این کلاس، یک کلاس مجرد است که کلاس پایه همه گیت های منطقی می باشد یعنی این کلاس ویژگی های مشترک همه گیت های منطقی را دارا است و همه گیت ها از آن مشتق می شوند.

خلاصه اعضا	
protected GateDelay	**delay** مقادیر تأخیر گیت.
protected java.util.Vector	**faults** لیست faultهای موجود در خروجی و ورودیهای گیت.
protected java.util.Vector	**inputs** ورودیهای گیت.

int	**level** سطح این گیت منطقی.
protected java.lang.String	**name** اسم این گیت منطقی.
protected Wire	**output** خروجی گیت.
protected java.lang.String	**type** نوع گیت.

خلاصه توابع	
abstract java.util.Vector	**collapse** (java.util.Vector collapsedFaults) fault های این گیت را collapse می کند.
abstract java.util.Vector	**getOutputFaultList**(java.util.HashMap recent) برای deduction از این تابع استفاده می شود.
abstract java.lang.String	**logicFunction**() تابع منطقی خروجی این گیت می باشد.
void	**resetValues**() مقدار تمام سیم های گیت را به حالت اولیه یعنی نامعلوم می برد.

circuit.logicelements.logicgates

Class AND

java.lang.Object
  └ circuit.logicelements.logicgates.AbstractLogicGate
    └ **circuit.logicelements.logicgates.AND**

کلاس متناظر با گیت and .

circuit.logicelements.logicgates

Class NAND

java.lang.Object
  └ circuit.logicelements.logicgates.AbstractLogicGate
    └ **circuit.logicelements.logicgates.NAND**

کلاس متناظر با گیت nand .

circuit.logicelements.logicgates

Class OR

```
java.lang.Object
  └circuit.logicelements.logicgates.AbstractLogicGate
      └circuit.logicelements.logicgates.OR
```

کلاس متناظر با گیت or .

circuit.logicelements.logicgates

Class NOR

```
java.lang.Object
  └circuit.logicelements.logicgates.AbstractLogicGate
      └circuit.logicelements.logicgates.NOR
```

کلاس متناظر با گیت nor .

circuit.logicelements.logicgates

Class XOR

```
java.lang.Object
  └circuit.logicelements.logicgates.AbstractLogicGate
      └circuit.logicelements.logicgates.XOR
```

کلاس متناظر با گیت xor .

circuit.logicelements.logicgates

Class XNOR

```
java.lang.Object
  └circuit.logicelements.logicgates.AbstractLogicGate
      └circuit.logicelements.logicgates.XNOR
```

کلاس متناظر با گیت xnor .

circuit.logicelements.logicgates

Class NOT

```
java.lang.Object
  └circuit.logicelements.logicgates.AbstractLogicGate
      └circuit.logicelements.logicgates.NOT
```

کلاس متناظر با گیت not .

circuit.logicelements.logicgates

Class BUF

```
java.lang.Object
  └circuit.logicelements.logicgates.AbstractLogicGate
      └circuit.logicelements.logicgates.BUF
```

کلاس متناظر با گیت buf؛ این کلاس یک تفاوت با بقیه کلاس ها دارد و آن این است که چون می خواهیم انشعاب را هم پشتیبانی کنیم، از این گیت استفاده می کنیم. بنابرین در موقع collape کردن fault ها یا شبیه سازی، این گیت، نقشی بازی نمی کند.

circuit.logicelements

Class Wire

```
java.lang.Object
  └java.util.Observable
      └circuit.logicelements.Wire
```

All Implemented Interfaces:

این کلاس نمایانگر یک سیم است. وقتی مقدار منطقی اش تغییر می کند، مقدار منطقی سیمهایی را که به آن متصل هستند تغییر می دهد یا برعکس وقتی سیم هایی که آنرا drive می کنند، مقدارشان عوض شود، مقدار این سیم نیزعوض می شود.

circuit.delay

Class Delay

```
java.lang.Object
  └circuit.delay.Delay
```

این کلاس، نمایانگر یک مولفه تأخیر می باشد که خود از سه مقدار minimum, typical, maximum تشکیل شده است.

circuit.delay

Class GateDelay

```
java.lang.Object
  └circuit.delay.GateDelay
```

این کلاس نشانگر تأخیر یک گیت منطقی است که از سه عنصر rise, fall, turnoff تشکیل شده است. نوع پیش فرض این تأخیر، inertial است، برای داشتن تأخیر transport باید آنرا صریحا در کد مشخص کنیم.

circuit

Class Circuit

```
java.lang.Object
  └circuit.Circuit
```

این کلاس، اطلاعات کل یک مدار همراه با fault های اولیه و fault های collapse شده را در خود نگهداری می کند.

خلاصه اعضا	
private java.util.Vector	**allFaults** لیست تمام fault هایی که این مدار دارد.
private java.util.Vector	**collapsedFaults** لیست fault های collapse شده مدار.
private java.util.Vector	**levels** لیستی که عنصر iام آن، حاوی گیت های سطح iام مدار می باشد.
private int	**numberOfPrimaryInputs** تعداد ورودیهای اولیه مدار.
private int	**numberOfPrimaryOutputs** تعداد خروجی های اولیه مدار.
private java.util.Vector	**primaryInputs** لیست ورودی های اولیه.
private java.util.Vector	**primaryOutputs** لیست خروجی های اولیه.

circuit

Class Event

```
java.lang.Object
  └circuit.Event
```

این کلاس نمایانگر یک رویداد می باشد که دارای سه قسمت اسم سیم، مقدار منطقی سیم و زمان رویداد است.

circuit

Class Fault

```
java.lang.Object
  └circuit.Fault
```

این کلاس نمایانگر یک stuck-at fault می باشد.

atpg

Class PseudoRandomTG

```
java.lang.Object
  └ atpg.PseudoRandomTG
```

این کلاس تمام کارهای مربوط به تولید تست شبه تصادفی را برعهده دارد.

atpg

Class PODEM

```
java.lang.Object
  └ atpg.PODEM
```

این کلاس تمام کارهای مربوط به تولید تست PODEM را برعهده دارد.