2023（第五届）集成电路EDA设计精英挑战赛
赛题指南

一、 赛题名称

基于 VCD 的 FSM 覆盖率统计

二、 命题企业

芯华章科技股份有限公司

三、 赛题Chair

李康（西安电子科技大学）

四、 背景知识

Coverage是衡量数字验证质量的重要指标，FSM Coverage作为其中重要一环，衡量了验证过程中的状态和状态转移覆盖状况。通过检查各状态和状态转移是否被覆盖到，我们可以检查预期的功能是否被覆盖；相反的，通过检查是否有不在设计范围内的状态或状态转移被意外覆盖到，进而检测到设计中的漏洞。

FSM Coverage通常包括两部分内容，第一部分是FSM（有限

状态机）识别和模型提取，该模型描述了有限个状态以及这些状态之间的转移行为；第二部分是FSM覆盖情况，即提取的状态和状态转移模型是否在仿真过程中被覆盖到。

值得注意的是，第一部分有限状态机识别发生在静态分析的过程中，识别的是可能的状态和可能的状态转移，并不代表被识别的状态或状态转移一定能在仿真过程中被覆盖，这一点需要跟第二步区分开来。

在本赛题中，我们会关注其中更为重要的第二步，FSM Coverage的计算，对于第一步状态机的识别，我们已使用芯华章GalaxSim的自动识别提取FSM的功能，通过YAML文件提供了FSM的相关信息，包括FSM的信号名字，状态名字，状态值，状态转移。

五、 赛题背景

为了更好的理解FSM识别，图1和图2分别是verilog源文件示例及其对应的FSM YAML文件：

```
1  // RTL
2  `timescale 1ns/1ns
3  module test_fsm(clk, reset, detect)
4  input clk, reset, detect;
5  reg [1:0] current, next;
6  parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
7
8  always @(clk)
9  case(current)
10   S0: begin
11      if (detect) next = S1;
12      else next = S3;
13   end
14   S1: next = S2;
15   default: next = S0;
16 endcase
17
18 always @(posedge clk)
19   if (reset) current = S0;
20   else current = next;
21
22 endmodule
```

```
1  //Testbench:
2  `timescale 1ns/1ns
3  module test;
4  reg clk, reset, detect;
5
6  always #50 clk = ~clk;
7
8  initial begin
9    clk = 0;
10   reset = 0;
11   detect = 1;
12   #400 $finish;
13 end
14
15 test_fsm test_fsm1(clk, reset, detect);
16
17 endmodule
```

图1 RTL以及Testbench样例

在图1 RTL示例中，由第9行到16行的case语句块可以判断，当前current的值决定了信号next的值，由18行到20行的always语句块可知，每次有上升沿的信号clk，如果reset=true，信号current会在19行被重置成S0；否则，在20行，next的值又会被赋给current。我们由此不难发现这个FSM的几个要素：

1. 信号current被用来追踪当前信号的值；
2. 信号current可能达到的几个状态值，分别是S0，S1，S2，S3；

3. 信号next作为用来给信号current传递值的媒介，可将下个状态值传递给current；
4. 信号current在接收到下个状态值时，产生状态转移。

```
1  FSMCONFIG:
2   -FSM: current
3    LINKS:
4      - next
5    MODULE: test_fsm
6    STATES:
7      - S3: 3
8      - S2: 2
9      - S0: 0
10     - S1: 1
11   TRANSITIONS:
12     - S1->S2
13     - S0->S1
14     - S3->S0
15     - S2->S0
16     - S1->S0
17     - S0->S3
```

图2 自动识别生成的FSM YAML文件（芯华章科技股份有限公司GalaxSim提供）

对照图1的样例参看图2的YAML文件，第2行FSM为有限状态机的信号名字，在这里即current。第3行LINKS是直接或间接给FSM (current)信号赋值的信号（可能有多个）。第5行MODULE是当前FSM信号所在的模块名字。第6行STATES包含了所有状态的名字和他们各自的值。第11行TRANSITIONS包含了所有状态转移，每个状态转移的形式为：<from_state_name> + '->' + <to_state_name>。

由第10，11行可知，当current = S0且detect = 0时，信

号current可以发生状态转移S0->S1 （next信号会被赋值，并在某个时机，即20行，信号next又会赋值给current）；由第12行可知，当current = S0且detect != 0时，信号current可以发生状态转移S0->S3；由第14行可知，当current = S1时，信号current可发生状态转移S1->S2；由第15行default语句可知，只有当current不等于S0，S1时才会进入这行，而我们考虑的信号current的状态总共有S0，S1，S2，S3 四种，所以只考虑current = S2或S3两种状态时进入15行的default语句，所以信号current在15行会有两个可能的状态转移：S2->S0，S3->S0。剩下几个状态转移均由第16行的reset分支产生，任何当前状态都有可能被重置成S0，本赛题不考虑自身转移，即S0->S0，所以此处产生S1->S0，S2->S0，S3->S0 三种状态转移。所以共有六个状态转移，分别为S0->S1，S0->S3，S1->S2，S1->S0，S2->S0，S3->S0。

VCD（Value Change Dump）是IEEE 1364（Verilog语言标准）中定义的基于ASCII的文件。它完整的记录了仿真过程中信号变化的信息，我们可以通过VCD文件确定在仿真过程中，信号的取值变化，进而确定FSM的状态和状态转移是否被覆盖到。对于图1给出的RTL和Testbench示例，通过GalaxSim可以生成如下VCD文件：

```
$timescale
1ns
$end

$scope module test $end
$var reg 1 ! clk $end
$var reg 1 “ reset $end
$var reg 1 # detect $end

$scope module test_fsm1 $end
$var wire 1 $ clk $end
$var wire 1 % reset $end
$var wire 1 & detect $end
$var reg 2 ‘ current [1:0] $end
$var reg 2 ( next [1:0] $end
$upscope $end

$upscope $end

$enddefinitions $end
```

```
#0
$dumpvars
0!
b00 (
bxx ‘
1&
0%
0$
1#
0“
$end
#50
1!
1$
b00 ‘
b01 (
#100
0!
0$
```

```
#150
b01 ‘
1$
1!
b10 (
#200
0$
0!
#250
b10 ‘
1!
b00 (
#300
0!
0$
#350
b00 ‘
1$
1!
b01 (
```

图3 VCD文件

为了更加形象具体的展示 VCD 文件的细节，图4给出了芯华章科技股份有限公司的商业调试软件Fusion Debug的波形展

示图，这里可以看出信号clk每50ns翻转一次，current信号值随着时间的变化如下图所示。

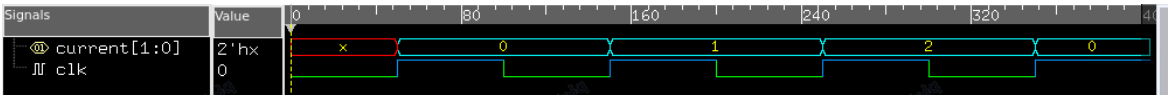

图4 信号变化波形图

图1中RTL例子总共有6个状态变化，由上图可知，总共有0->1（S0->S1），1->2（S1->S2），2->0（S2->S0）的状态转移被覆盖，所以覆盖率为3/6 = 50%（注意此赛题中我们计算FSM覆盖率只关注状态转移的覆盖率，无需考虑状态的覆盖率）。

FSM覆盖率 =（状态转移覆盖数量）/（状态转移总数）

六、 赛题介绍

题一、 VCD 建模并计算 FSM 覆盖率 （50 分）

根据测试用例所提供的：Verilog设计源文件；filelist.f(如果有多个Verilog源文件，提供此文件列表)；由GalaxSim抽取出FSM YAML文件；仿真时产生的VCD文件；Gold文件(答案需要跟Gold文件一致才能得分)，参赛者需要正确解析YAML文件，可使用第三方库（需要在报告中提交第三方库的来源），以读取FSM相关数据，创建FSM模型，针对整个仿真过程，即从时刻0到仿真结束，计算FSM覆盖率，将结果写入summary.csv文件。

（1）支持 VCD 文件解析（可使用第三方库）与建模 （5 分）

（2）计算整个仿真过程的FSM覆盖率 （15分）

（3）回归测试 （5分）

（4）单核及多核性能 （25分）

建议使用C++/C等高性能编程语言。

分析过程详细说明见“**七、赛题与评分标准解析**”章节。

题二、 支持连续时间窗口的FSM覆盖率统计 （50分）

假设输入为初始时间T0，时间窗口为t，从初始时刻T0到结束时刻T1，总共有N个时间窗口(最后一个时间窗口长度可能小于t)，其中：

$$T0 + t * N \geq T1 > T0 + t * (N - 1)$$

当$0 < k < N$，统计每一个$[T0,\ T0 + k * t]$时间范围内的FSM覆盖率，注意包括$T0$和$T0 + k * t$两个时间点。

当$k = N$，统计$[T0,\ T1]$的覆盖率，注意包括$T0$和$T1$两个时间点。

T0，T1，t的时间单位均为每个对应样例中的timescale，所以无需关注时间单位转换。对于每个测试用例，我们会提供文件input_windows.csv。运行程序“-windows input_windows.csv”，将结果写入summary_windows.csv文件。

input_windows.csv内容如下，每一行表示一组T0，T1和t的输入值，我们会测试一个或多个组合，所以该文件有一或多

行。但 T1 保持不变。例如下面表格表示对于当前测试样例，我们会测试（T0=50，T1=400，t=50）以及（T0=100，T1=400，t=80）两个组合（图 5 为示例，实际为不含表头且以英文逗号分隔的 csv 文件）

初始时间 T0	结束时间 T1	时间窗口 t
50	400	50
100	400	80

图5 input_windows.csv文件示例

(1) 计算每个测试用例给定的时间窗口覆盖率统计 （20 分）

(2) 回归测试 （5 分）

(3) 单核及多核性能 （25 分）

分析过程详细说明见“**七、赛题与评分标准解析**”章节。

题三、 附加题 （20 分）

支持 FSM 覆盖率的合并功能。具体而言，我们会提供附加的测试用例，放在名为 merge 的文件夹下，基于每一个测试用例，我们会提供多个 VCD 文件，是由测试用例通过随机性测试多次仿真得出来的结果。但都基于同一个 RTL 设计，也就是说，在静态分析 FSM 阶段，FSM 的识别是相同的，所以一个测试用例还是只有一个 FSM YAML 文件。

（1）基于多个仿真结果(多个 VCD 文件)，合并覆盖结果，计算

覆盖率 （10 分）

即某个状态转移在任意一个仿真结果中被覆盖到，就认为它是被覆盖到了，计算覆盖率。

（2）单核及多核性能 （10 分）

分析过程详细说明见“**七、赛题与评分标准解析**”章节。

七、 赛题与评分标准解析

题一、 VCD 建模并计算 FSM 覆盖率（50 分）

1）支持VCD文件解析与建模 （5分）

正确解析 VCD 文件，可以正确读取 FSM 信号值，可使用第三方库。解析错误不得分。如果使用第三方库，需要在报告中提交第三方库的来源。

2）计算整个仿真过程的FSM覆盖率 （15分）

以图 1，2，三所描述的测试用例为例，Gold 文件如下：

test_fsm.current,3,6,50.00%

总共四列，以英文 ‘,’ 隔开，注意没有空格，每一列分别表示：

1. 模块名字 + ‘.’ + FSM信号名字
2. 在仿真过程中该FSM所覆盖的状态转移个数
3. 该FSM状态转移总数

4. FSM覆盖率

其中FSM覆盖率 = 覆盖的状态转移个数 / 状态转移总数

以百分数形式输出，百分号前保留两位小数，不足两位小数则补零。

本赛题（包含题一，题二，题三）会提供公共测试集，包含十个测试用例，以供参赛者验证功能正确性以及性能指标，任何测试样例输出结果与Gold不同则不得分。（注意在**单核及多核性能**模块，会使用另外隐藏的性能测试集来测试代码性能，包含十个性能测试用例，该测试集不提供给参赛选手。）

3）回归测试 （5分）

注意该程序能够一次跑完所有的测试用例，并且将每个结果和Gold进行对比，体现出与Gold匹配，否则得零分。

4）单核及多核性能（25分）

（i）单核性能 （15 分）

对每个用例，以程序运行开始到结束时间为衡量依据（之后简称时间性能），时间性能超越80%队伍得1.5分，时间性能超越50%但不超越80%的队伍得1分；以程序运行过程的峰值内存为衡量依据（之后简称内存消耗），内存消耗少于95%队伍得1分。

（ii）多核性能（四核） （10 分）

对每个用例，时间性能超越80%队伍得1分，时间性能超越50%但不超越80%队伍得0.5分；内存消耗少于95%队伍得0.5分。

题二、 支持连续时间窗口的 FSM 覆盖率统计（50 分）

1）计算每个测试用例给定的时间窗口覆盖率统计 （20 分）

假设input_windows.csv内容如下，每一行表示一组T0，T1和t的输入值，我们会测试一个或多个组合，所以该文件有一或多行。但T1保持不变。例如下面表格表示对于当前测试样例，我们会测试（T0=50，T1=400，t=50）以及（T0=100，T1=400，t=80）两个组合（图5为示例，实际为不含表头且以英文逗号分隔的csv文件）

初始时间 T0	结束时间 T1	时间窗口t
50	400	50
100	400	80

图5 input_windows.csv文件示例

同样以图1，2样例为例，根据图5的输入时间组合，其Gold文件如图6所示（图6为示例，实际为不含表头且以英文逗号分隔的csv文件）

FSM名字	窗口左边界	窗口右边界	FSM覆盖率
test_fsm.current	50	100	0.00%
test_fsm.current	50	150	16.67%
test_fsm.current	50	200	16.67%
test_fsm.current	50	250	33.33%
test_fsm.current	50	300	33.33%
test_fsm.current	50	350	50.00%
test_fsm.current	50	400	50.00%
test_fsm.current	100	180	16.67%
test_fsm.current	100	260	33.33%
test_fsm.current	100	340	33.33%
test_fsm.current	100	400	50.00%

图6 单个FSM根据input_windows.csv的输出示例

每一列分别表示：

1. 模块名字 + ‘.’ + FSM信号名字
2. 时间窗口左边界值，应该等于其对应的T0。
3. 时间窗口右边界值，按照顺序以$T0 + k * t$的计算方法逐步递增，对应一组输入时间组合的最后一行为$T1$。
4. 在此行时间窗口左右边界内的FSM覆盖率，注意左右时间

点都包括在内。覆盖率以百分数形式输出，百分号前保留两位小数，不足两位小数则补零。

当T0=50，T1=400，t=50时，由于信号current在150ns时刻发生了0->1 （S0->S1）状态转移，所以在此之前的[50:100]窗口，覆盖率为0.00%；而在[50:150]窗口覆盖率为1/6=16.67%；在250ns时刻，current发生1->2（S1->S2）状态转移，所以在[50:200]窗口覆盖率为1/6=16.67%，而在[50:250]窗口覆盖率为2/6=33.33%；在350ns时刻，current发生2->0（S2->S0）状态转移，所以在[50:300]窗口覆盖率为33.33%，而在[50:350]窗口覆盖率为50%；在此之后current信号值并没有发生改变，所以[50:400]窗口覆盖率依然保持50%。

当T0=100，T1=400，t=80时，由于信号current在150ns时刻发生了0->1（S0->S1）的状态转移，之后在180ns前没有再变化，所以在[100:180]窗口状态转移覆盖率为1/6=16.67%；以此类推，在[100:260]窗口范围内，信号current存在0->1，1->2的两个状态转移，所以覆盖率为2/6=33.33%；在[100:340]窗口范围内，信号current存在0->1，1->2两个状态转移，所以覆盖率为2/6=33.33%；在[100:400]窗口范围内，信号current存在0->1，1->2，2->0三个状态转移，所以覆盖率为50%。

2）回归测试 （5 分）

注意该程序能够一次跑完所有的测试用例，并且将每个结果和Gold进行对比，体现出与Gold匹配，否则得零分。

3）单核及多核性能（25 分）

（i）单核性能 （15 分）

对每个用例，以程序运行开始到结束时间为衡量依据（之后简称时间性能），时间性能超越80%队伍得1.5分，时间性能超越50%但不超越80%的队伍得1分；以程序运行过程的峰值内存为衡量依据（之后简称内存消耗），内存消耗少于95%队伍得1分。

（ii）多核性能（四核） （10 分）

对每个用例，时间性能超越80%队伍得1分，时间性能超越50%但不超越80%队伍得0.5分；内存消耗少于95%队伍得0.5分。

题三、 附加题（20 分）

1） 基于多个仿真结果(多个 VCD 文件)，合并覆盖结果，计算覆盖率 （10 分）

以图一所示的RTL设计为例。假如我们给定了两个VCD文件，分别表示两种不同的随机激励产生的仿真结果（注意他们对应的YAML文件是同一个），在这里用图8（a）和图8（b）表示：

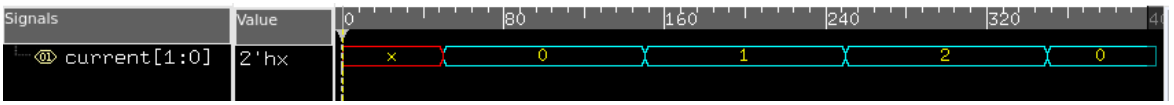

图8(a) VCD1对应的波形图(附加题)

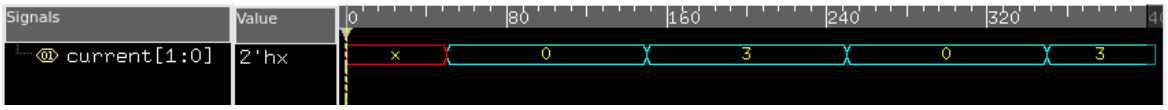

图8(b) VCD2对应的波形图(附加题)

由图8（a）可知，S0->S1，S1->S2，S2->S0 三个状态转移被覆盖到。根据图8（b）可知，S0->S3，S3->S0被覆盖到，所以总共有S0->S1，S1->S2，S2->S0，S0->S3，S3->S0 五个状态转移被覆盖到，覆盖率为5/6=83.33%。

结果需要写入summary_merge.csv文件，上述样例对应的Gold文件如下：

top.current,5,6,83.33%

每一列分别表示：

1. 模块名字 + ‘.’ + FSM信号名字
2. 合并后该FSM所覆盖的状态转移个数
3. 状态转移总数
4. 合并后的FSM覆盖率

其中FSM覆盖率 = 合并后覆盖的状态转移个数 / 状态转移总数

以百分数形式输出，百分号前保留两位小数，不足两位小

数则补零。

注意Gold文件可以有多行，如果一个用例中有多个FSM，需要计算所有FSM覆盖率的合并结果，顺序按照YAML文件给定的FSM顺序，给定的VCD文件数量大于等于2，需要全部合并起来。

2） 单核及多核性能（10 分）

（i）单核性能 （5 分）

共十个性能测试用例，对每个用例，以程序运行开始到结束时间为衡量依据（之后简称时间性能），时间性能超越80%队伍得0.5分，时间性能超越50%但不超越80%的队伍得0.25分；以程序运行过程的峰值内存为衡量依据（之后简称内存消耗），内存消耗少于95%队伍得0.25分。

（ii）多核性能（四核） （5 分）

共十个性能测试用例，对每个用例，时间性能超越80%队伍得0.5分，时间性能超越50%但不超越80%的队伍得0.25分；内存消耗少于95%队伍得0.25分。

八、 参考资料

1）1800-2017 - IEEE Standard for SystemVerilog— Unified Hardware Design, Specification, and Verification Language

2）1364-2001 - IEEE Standard Verilog Hardware Description Language

3）Chris Terman, MIT Spring 2017, Computation Structures, https://ocw.mit.edu/courses/6-004-computation-structures-spring-2017/pages/c6/c6s1/

4）Gim P. Hom, Joe Steinmyer, MIT Fall 2017, Introductory Digital Systems Laboratory, https://web.mit.edu/6.111/www/f2017/handouts/L06.pdf

5）John, Wawrzynek, UC Berkeley, College of Engineering Department of Electrical Engineering and Computer Science, Digital Design, https://inst.eecs.berkeley.edu/~cs150/sp12/resources/FSM.pdf

九、 赛题解释权归命题企业所有