

## 开发环境

- Yosys: v0.33。使用`sudo apt install yosys`安装。
- cvc5: v1.1.2。使用`pysmt-install --cvc5`安装。
- PySMT: v0.9.6。
- Python 3.12.3。
- Ubuntu 20.04。

## 使用

### 运行命令

```
1 | python main.py xxx.v top_module_name testbench_file [-k step]
```

xxx.v为待测Verilog代码。

top\_module\_name为Verilog代码中顶层模块名称。

testbench\_file为自定义testbench文件。

-k为BMC最大展开步数，默认为20。

### 示例

1. 在dut文件夹下创建待测模块

```
1 // counter.v
2 module counter(
3     input clk,
4     input rst_n, // 低电平有效复位
5     input [3:0] initval, // (在这个测试中未使用)
6     output reg [3:0] out
7 );
8
9     initial out = 'd0;
10
11    always @(posedge clk) begin
12        if (rst_n == 1'b0) begin
13            out <= 4'b0000;
14        end else begin
15            out <= out + 1;
16        end
17    end
18
19 endmodule
```

2. 在tb文件夹下创建testbench

```
1 ; test01.txt
2 [CLOCK]
3 clk = 1
```

```

4
5 [PROPERTY]
6 out == 2
7
8 [PROCESS]
9 ; 1. 启动时, 保持复位状态
10 initval = 0
11 rst_n = 0
12
13 ; 2. 保持复位 5 个时钟周期 (#5 意味着持续 5 步, k=0 到 k=4)
14 #5
15
16 ; 3. 在第 5 步 (k=5) 时, 释放复位
17 rst_n = 1

```

testbench使用自定义格式, 主要包含了3部分。

- **[CLOCK]**: 定义时钟周期, 例如clk = 2, 表示占空比为50%的时钟, 每2个step电平翻转。
- **[PROPERTY]**: 不变量/属性, 例如out == 2表示, 我们要检查它是否能在k步内达到 2。目前支持 ==, !=, <, <=, >, >= 操作符。
- **[PROCESS]**: 输入信号控制流程, 内部只有两种语法:
  - 阻塞赋值:symbol = val。
  - 延时:#n,n的单位为step。

3. 在工程根目录下运行:

```
1 | python main.py dut/counter.v counter tb/test01.txt -k 10
```

运行结果如下:

```

1 | --- 正在执行 Yosys 转换 ---
2 | 命令: yosys -p "read_verilog -nomem2reg -sv
   | /home/lvt/Verilog_MBC/proj/dut/counter.v; prep -top counter;
   | hierarchy -check; memory -nomap; flatten;; clk2fflogic;; setundef -
   | undriven -anyseq; write_btor
   | /home/lvt/Verilog_MBC/proj/dut/counter.btor2"
3 | --- 转换完成, BTOR2文件生成: dut/counter.btor2 ---
4 | --- 解析 BTOR2 文件: dut/counter.btor2 ---
5 |
6 | --- 解析结果摘要 ---
7 | 状态变量: ['state_9', 'state_12', 'state_7']
8 | 输入变量: ['clk', 'initval', 'rst_n']
9 | 初始状态: ((state_7 = 0_4) & (state_9 = 0_4) & (state_12 = 1_1))
10 | 不变量: (out = (((state_12::clk) bvcomp 1_2) = 1_1) ? state_9 :
   | state_7))
11 | 转换关系:
12 |   state_7 -> (((state_12::clk) bvcomp 1_2) = 1_1) ? state_9 :
   | state_7)
13 |   state_9 -> ((rst_n = 1_1) ? (((state_12::clk) bvcomp 1_2) = 1_1)
   | ? state_9 : state_7) + (1_1 ZEXT 3)) : 0_4)
14 |   state_12 -> clk
15 | -----
16 |
17 | 加载属性: out == 2
18 | 加载了 2 个 process 段。

```

```
19 加载了时钟: {'clk': 1}
20 --- 开始 BMC (K_max = 10) ---
21 正在检查步骤 K = 0 (段 0, 段内步数 0)
22 正在检查步骤 K = 1 (段 0, 段内步数 1)
23 正在检查步骤 K = 2 (段 0, 段内步数 2)
24 正在检查步骤 K = 3 (段 0, 段内步数 3)
25 正在检查步骤 K = 4 (段 0, 段内步数 4)
26 正在检查步骤 K = 5 (段 1, 段内步数 0)
27 正在检查步骤 K = 6 (段 1, 段内步数 1)
28 正在检查步骤 K = 7 (段 1, 段内步数 2)
29 正在检查步骤 K = 8 (段 1, 段内步数 3)
30 正在检查步骤 K = 9 (段 1, 段内步数 4)
31
32 !!! 属性 'out == 2' 在步骤 9 变为 TRUE !!!
33
34 --- 反例信息 ---
35 --- 步骤 0 ---
36     clk: 0_1
37     initval: 0_4
38     rst_n: 0_1
39     state_7: 0_4
40     state_9: 0_4
41     state_12: 1_1
42     out: 0_4
43 --- 步骤 1 ---
44     clk: 1_1
45     initval: 0_4
46     rst_n: 0_1
47     state_7: 0_4
48     state_9: 0_4
49     state_12: 0_1
50     out: 0_4
51 --- 步骤 2 ---
52     clk: 0_1
53     initval: 0_4
54     rst_n: 0_1
55     state_7: 0_4
56     state_9: 0_4
57     state_12: 1_1
58     out: 0_4
59 --- 步骤 3 ---
60     clk: 1_1
61     initval: 0_4
62     rst_n: 0_1
63     state_7: 0_4
64     state_9: 0_4
65     state_12: 0_1
66     out: 0_4
67 --- 步骤 4 ---
68     clk: 0_1
69     initval: 0_4
70     rst_n: 0_1
71     state_7: 0_4
72     state_9: 0_4
```

```

73     state_12: 1_1
74     out: 0_4
75     --- 步骤 5 ---
76     clk: 1_1
77     initval: 0_4
78     rst_n: 1_1
79     state_7: 0_4
80     state_9: 0_4
81     state_12: 0_1
82     out: 0_4
83     --- 步骤 6 ---
84     clk: 0_1
85     initval: 0_4
86     rst_n: 1_1
87     state_7: 0_4
88     state_9: 1_4
89     state_12: 1_1
90     out: 0_4
91     --- 步骤 7 ---
92     clk: 1_1
93     initval: 0_4
94     rst_n: 1_1
95     state_7: 0_4
96     state_9: 1_4
97     state_12: 0_1
98     out: 1_4
99     --- 步骤 8 ---
100    clk: 0_1
101    initval: 0_4
102    rst_n: 1_1
103    state_7: 1_4
104    state_9: 2_4
105    state_12: 1_1
106    out: 1_4
107    --- 步骤 9 ---
108    clk: 1_1
109    initval: 0_4
110    rst_n: 1_1
111    state_7: 1_4
112    state_9: 2_4
113    state_12: 0_1
114    out: 2_4
115    -----

```

运行流程示意图:

