SYSTEMVERILOG ASSERTION

Nguồn: https://www.chipverify.com/systemverilog/systemverilog-assertions

Hành vi của một hệ thống có thể viết bằng một assertion mà nó nên đúng ở mọi trường hợp. Do đó assertions được sử dụng để xác thực hành vi của một hệ thống, có thể cũng được dùng trong functional coverage

What are properties (đặc tính) of a design?

Nếu một đặc tính của thiết kế được kiểm tra bởi một assertion không hoạt động như được kì vọng, **assertion thất bại.**

Ví dụ: giả sử một thiết kế yêu cầu và mong muốn nhận được một ack trong 4 chu kỳ kế tiếp. Nhưng nếu thiết kết nhận được một ack ở chu kỳ thứ 5, đặc tính của thiết kế mà một ack sẽ được trả về trong 4 chu kỳ bị vi phạm và assertion được xem như thất bại

Nếu một đặc tính của thiết kế đang được kiểm tra bởi một assertion, mà đặc tính không thể xảy ra do bị chặn, **assertion thất bại.**

Ví dụ: Giả sử một processor đơn giản có chức năng decode các instruction được đọc từ memory, gặp phải một unknow instruction và kết quả trả về là một fatal error. Nếu viễn cảnh này không bao giờ được kỳ vọng bởi thiế kế, đặc tính của thiết kế mà chỉ valid instruction có thể được đọc từ memory bị vi phạm và assertion được xem như thất bại.

Từ 2 ví dụ trên, các đặc tính của một thiết kế được kiểm tra bằng cách viết các SV assertions.

Why do we need assertions?

Một assertion đơn giản chỉ là một thể hiện ngắn gọn của một bộ kiểm tra function (functional checker). Functionality biểu diễn bởi một assertion cũng có thể được viết bằng một SV task hoặc một checker mà user phải tự code. Một vài bất lợi của các phương pháp này là:

- SV dài dòng và khó để duy trì và scale code với số lượng đặc tính
- là một procedural language, user sẽ gặp khó khăn trong việc viết checkers liên quan đến nhiều sự kiện song song chung một khoảng thời gian

```
// A property written in Verilog/SystemVerilog
always @ (posedge clk) begin
    if (!(a && b))
        $display ("Assertion failed");
end
```

SVA là một declarative language dùng để specify các điều kiện tạm thời, SVA ngắn gọn và dễ dàng hơn trong việc duy trì.

```
// The property above written in SystemVerilog Assertions syntax
assert property(@(posedge clk) a && b);
```

Type of Assertion Statements

Type	Description
assert	To specify that the given property of the design is true in simulation

assume	To specify that the given property is an assumption and used by formal tools to generate input stimulus
cover	To evaluate the property for functional coverage
restrict	To specify the property as a constraint on formal verification computations and is ignored by simulators

Building Blocks of Assertions

Sequence

Chức năng của bất kỳ thiết kê nào thường đều được hình thành bởi một sequence (chuỗi) gồm nhiều logical events. Các events này có thể trải dài nhiều xung clocks hoặc chỉ tồn tại trong một xung clock nhất định. Để giữ mọi thứ đơn giản, các events nhỏ hơn có thể được miêu tả bằng cách sử dụng các assertions đơn giản, sau đó các assertions này có thể được dùng để xây dựng các behavior patterns phức tạp hơn.

```
// Sequence syntax
sequence <name_of_sequence>
  <test expression>
endsequence

// Assert the sequence
assert property (<name_of_sequence>);
```

Property

Các evnets có thể được biểu diễn như một chuỗi (sequence) và một vài sequences có thể được gộp lại để tạo ra các chuỗi đặc tính phức tạp hơn.

User cần phải thêm một clocking event vào trong chuỗi (sequence) hoặc đặc tính (property) để assert.

```
// Property syntax
property <name_of_property>
  <test expression> or
  <sequence expressions>
endproperty

// Assert the property
assert property (<name_of_property>);
```

Có 2 loại assertions - tức thì và đồng thời (immediate and concurrent).

Immediate Assertion

Immediate assertions được thực thi như một statement trong một proceduarl block và follow simulation event semantics.

Immediate assertion được dùng để verify một đặc tính tức thì trong quá trình mô phỏng.

```
// Define a property to specify that an ack should be
// returned for every grant within 1:4 clocks
property p_ack;
    @(posedge clk) gnt ##[1:4] ack;
endproperty

assert property(p_ack);    // Assert the given property is true always
```

Concurrent Asseritons

Concurrent assertions are base on clock semantics (phụ thuộc vào một clock được định nghĩa) và dùng các giá trị được sampled trong expressions của chúng. hành vi của mạch được mô tả sử dụng các đặc tính SV (SV properties), các đặc tính này được đánh giá mỗi khi clock đã định nghĩa và Mỗi khi có một lỗi xảy ra trong quá trình mô phỏng thể hiện rằng functional behavior được mô tả đã bị vi phạm.

```
// Define a property to specify that an ack should be
// returned for every grant within 1:4 clocks
property p_ack;
    @(posedge clk) gnt ##[1:4] ack;
endproperty

assert property(p_ack);    // Assert the given property is true always
```

Các bước để tạo assertions

Bên dưới là các bước để tạo assertions:

B1: Tạo các boolean expressions

B2: Tạo các sequence expressions

B3: Tạo property

B4: Assert property

Ví dụ:

sequence đầu tiên **s_ab** kiểm tra **b** có tích cực cao ở clock kế tiếp khi **a** ở mức cao, và sequence thứ 2 **s_cd** kiểm tra **d** có ở mức cao sau 2 xung clock sau khi **c** được xác nhận đã tích cực mức cao. **property đó** asserts sequence thứ 2 ở chu kỳ kế tiếp sau sequence thứ nhất.

```
module tb;
  bit a, b, c, d;
  bit clk;

  always #10 clk = ~clk;

  initial begin
    for (int i = 0; i < 20; i++) begin
      {a, b, c, d} = $random;
      $display("%0t a=%0d b=%0d c=%0d d=%0d", $time, a, b, c, d);
      @(posedge clk);
    end
    #10 $finish;
  end

  sequence s_ab;
    a ##1 b;
  endsequence

  sequence s_cd;
    c ##2 d;
  endsequence

  property p_expr;
    @(posedge clk) s_ab ##1 s_cd;
  endproperty

  assert property (p_expr);
endmodule
```

SV Immediate Assertions Detail

Nguồn: https://www.chipverify.com/systemverilog/systemverilog-immediate-assertions

Immediate assertion được thực thi dựa trên **ngữ nghĩa event mô phỏng**, Immediate assertion cần phải được specify **trong một proceduarl block**. Nó được trình mô phỏng xem như một expression trong một mệnh đề if trong quá trình mô phỏng.

Immediate assertion sẽ pass nếu expression giữ giá trị true tại thời điểm mệnh đề được thực thi, và nó sẽ fail nếu mệnh đề được đánh giá là thất bại (X, Z hoặc 0).Những assertions này được **dùng trong simulation và không phù hợp cho formal verification.** Nó có thể được dùng trong **cả RTL code và testbench** để bắt lỗi trong quá trình mô phỏng.

Syntax:

```
// Simple assert statement
assert(<expression>);

// Assert statement with statements to be executed for pass/fail conditions
assert(<expression>) begin
    // If condition is true, execute these statements
end else begin
    // If condition is false, execute these statements
end

// Optionally give name for the assertion
[assert_name] : assert(<expression>);
```

1. IMMEDIATE ASSERTION IN DESIGN

Ví dụ về assertion bên dưới là về 1 fifo, assertion fail nếu push khi full = 1 hoặc pop khi empty = 1.

```
module my_des (my_if _if);

  always @ (posedge _if.clk) begin
    if (_if.push) begin
        // Immediate assertion and ensures that
        // fifo is not full when push is 1
        a_push: assert (!_if.full) begin
            $display("[PASS] push when fifo not full");
        end else begin
            $display("[FAIL] push when fifo full !");
        end
    end

    if (_if.pop) begin
        // Immediate assertion to ensure that fifo is not
        // empty when pop is 1
        a_pop: assert (!_if.empty) begin
            $display ("[PASS] pop when fifo not empty");
        end else begin
            $display ("[FAIL] pop when fifo empty !");
        end
    end
  end
endmodule
```

testbench:

```
interface my_if(input bit clk);
  logic pop;
  logic push;
  logic empty;
  logic full;
endinterface

module tb;
  bit clk;
  always #10 clk <= ~clk;

  my_if _if (clk);
  my_des u0 (.*);

  initial begin
    for (int i = 0; i < 5; i++) begin
      _if.push  <= $random;
      _if.pop   <= $random;
      _if.empty <= $random;
      _if.full  <= $random;
      $strobe("[%0t] push=%0b full=%0b pop=%0b empty=%0b",
              $time, _if.push, _if.full, _if.pop, _if.empty);
      @(posedge clk);
    end
    #10 $finish;
  end
endmodule
```

Kết quả:

```
ncsim> run
[0] push=0 full=1 pop=1 empty=1
ncsim: *E,ASRTST (./design.sv,13): (time 10 NS) Assertion tb.u0.a_pop has failed
[FAIL] pop when fifo empty !
[10] push=1 full=0 pop=1 empty=1
[PASS] push when fifo not full
ncsim: *E,ASRTST (./design.sv,13): (time 30 NS) Assertion tb.u0.a_pop has failed
[FAIL] pop when fifo empty !
[30] push=1 full=1 pop=1 empty=0
ncsim: *E,ASRTST (./design.sv,5): (time 50 NS) Assertion tb.u0.a_push has failed
[FAIL] push when fifo full !
[PASS] pop when fifo not empty
[50] push=1 full=0 pop=0 empty=1
[PASS] push when fifo not full
[70] push=1 full=1 pop=0 empty=1
ncsim: *E,ASRTST (./design.sv,5): (time 90 NS) Assertion tb.u0.a_push has failed
[FAIL] push when fifo full !
Simulation complete via $finish(1) at time 100 NS + 0
./testbench.sv:25      #10 $finish;
ncsim> exit
```

2. IMMEDIATE ASSERTION IN TESTBENCH

Giả sử có một lớp Packet được tạo và randomize. Tuy nhiên ví dụ này có một lỗi constraint và randomization sẽ luôn thất bại. Tuy nhiên, failure sẽ được hiển thị như một warning message và nếu user không cẩn thân, test có thể display hành vi sai và thậm chí có thể hiện là assertion đã pass

```
class Packet;
  rand bit [7:0] addr;

  constraint c_addr { addr > 5; addr < 3; }
endclass

module tb;
  initial begin
    Packet m_pkt = new();

    m_pkt.randomize();
  end
endmodule
```

=> như constraint phía trên, ta ko thể randomize một addr mà giá trị của nó vừa lớp hơn 5 vừa nhỏ hơn 3.

Thay vào đó ta có thể sử dụng một immediate assertion đựat vào randomization method để chắc chắn rằng giá trị trả về luôn là 1, chỉ định rằng randomization thành công. Nếu assertion fails, prompt sẽ được hiển thị để user có thể thấy được, giúp giảm debug time.

```
class Packet;
  rand bit [7:0] addr;

  constraint c_addr { addr > 5; addr < 3; }
endclass

module tb;
  initial begin
    Packet m_pkt = new();

    assert(m_pkt.randomize());
  end
endmodule
```

Tương tự, assert có thể được dùng với bất kỳ expression nào để đánh giá true hoặc false bên trong một procedural block

SV Concurrent Assertions Detail

Nguồn: https://www.chipverify.com/systemverilog/systemverilog-concurrent-assertions

Concurrent assertions mô tả hành vi được dàn trải xuyên suốt thời gian mô phỏng và các concurrent assertion này chỉ được đánh giá ở clock tick.

Mệnh đề SV concurrent assertion có thể được specified trong **module, interface hoặc program block** chạy song song với các mệnh đề khác. Bên dưới là các đặc tính của một concurrent assertion:

- Test expression được đánh giá tại cạnh của xung clock dựa trên các giá trị trong biến được sampled
- sampling của các biến được hoàn thành trong **preponed region** và việc đánh giá của mệnh đề được hoàn thành trong **observed region** của simulation scheduler
- chúng có thể được đặt bên trong **procedural, module, interface hoặc program block**
- chúng có thể được dùng trong cả dynamic và formal verification technique

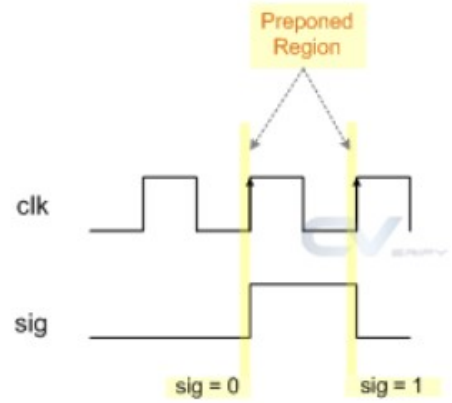
 # Example 1

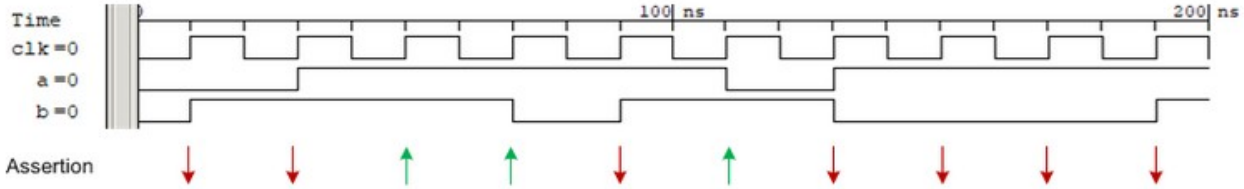
2 tín hiệu a và b được declared và được lái tại cạnh lên của một clock với một vài giá trị ngẫu nhiên để mô tả cách một concurrent assertion hoạt động. Assertion này được viết bằng mệnh đề assert trên một immediate property, property này xác định mối quan hệ giữa các tín hiệu và một clocking event

Trong ví dụ này, cả tín hiệu a và b được mong đợi là 1 tại cạnh lên xung clock trong toàn bộ mô phỏng. assertion được mong đợi sẽ thất bại cho tất cả instances khi a hoặc b là 0

```
module tb;
    bit a, b;
    bit clk;

    always #10 clk = ~clk;

    initial begin
        for (int i = 0; i < 10; i++) begin
            @(posedge clk);
            a <= $random;
            b <= $random;
            $display("[%0t] a=%0b b=%0b", $time, a, b);
        end
        #10 $finish;
    end

  // This assertion runs for entire duration of simulation
  // Ensure that both signals are high at posedge clk
  assert property (@(posedge clk) a & b);

endmodule
```

assertion được thực thi ở mỗi cạnh lên xung clock (clk) và đánh giá exoressuib sử dụng giá trị của các biến trong **preponed region** (which is delta cycle before given edge of clock). **Do đó, nếu a thay đổi từ 0 sang 1 ngay vị trí xung clock chuyển từ 0 sang 1, giá trị a được lấy để đánh giá bởi assertion sẽ là 0 bởi vì nó là 0 ngay trước cạnh lên xung clock**





Từ hình trên có thể thấy assertion fail ở tất cả trường hợp khi a hoặc b bằng 0 bởi vì mệnh đề assert được mong đợi là đúng xuyên suốt quá trình mô phỏng

Time (ns)	a	b	Result
10	0	0	FAIL
30	0	1	FAIL
50	1	1	PASS
70	1	1	PASS
90	1	0	FAIL
110	1	1	PASS
130	0	1	FAIL
150	1	0	FAIL
170	1	0	FAIL
190	1	0	FAIL

Kết quả khi chạy simulation trong vivado (kết quả khác bảng trên do random seed khác):

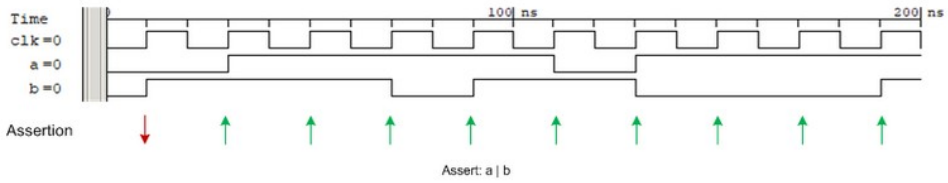
```
[10000] a=0 b=0
ERROR: Assertion failed.
Time: 10 ns Started: 10 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:40
[30000] a=0 b=1
ERROR: Assertion failed.
Time: 30 ns Started: 30 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:40
[50000] a=1 b=1
[70000] a=1 b=1
[90000] a=1 b=0
ERROR: Assertion failed.
Time: 90 ns Started: 90 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:40
[110000] a=1 b=1
[130000] a=0 b=1
ERROR: Assertion failed.
Time: 130 ns Started: 130 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:40
[150000] a=1 b=0
ERROR: Assertion failed.
Time: 150 ns Started: 150 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:40
[170000] a=1 b=0
ERROR: Assertion failed.
Time: 170 ns Started: 170 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:40
[190000] a=1 b=0
ERROR: Assertion failed.
Time: 190 ns Started: 190 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:40
$finish called at time : 200 ns : File "/home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv" Line 36
```

 # Example 2

```
module concurrent_assertion_ex1;
    bit a, b;
    bit clk;

    always #10 clk = ~clk;

    initial begin
        for (int i = 0; i < 10; i++) begin
            @(posedge clk);
            a <= $random;
            b <= $random;
            $display("[%0t] a=%0b b=%0b", $time, a, b);
        end
        #10 $finish;
    end

    check_1: assert property (@(posedge clk) a | b);
endmodule
```

Expression được defined như một property cho mệnh đề assert được modified từ ví dụ trên thành điều kiên OR (a & b => a | b)

Kết quả mệnh đề assert này chỉ fail khi cả 2 giá trị của a và b đều la 0:



Time (ns)	a	b	Result
10	0	0	FAIL
30	0	1	PASS
50	1	1	PASS
70	1	1	PASS
90	1	0	PASS
110	1	1	PASS
130	0	1	PASS
150	1	0	PASS
170	1	0	PASS
190	1	0	PASS

Kết quả mô phỏng khi chạy trên vivado (kết quả khác do seed khác):

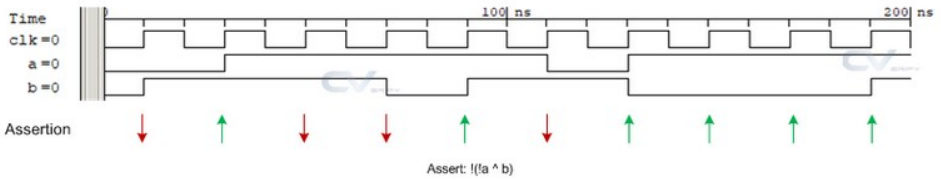
```
[30000] a=0 b=1
[50000] a=1 b=1
[70000] a=1 b=1
[90000] a=1 b=0
[110000] a=1 b=1
[130000] a=0 b=1
[150000] a=1 b=0
[170000] a=1 b=0
[190000] a=1 b=0
[210000] a=1 b=1
[230000] a=0 b=1
[250000] a=0 b=0
ERROR: Assertion failed.
Time: 250 ns Started: 250 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[270000] a=0 b=1
[290000] a=0 b=1
```

 # Example 3

Expression này được defined như một property của mệnh đề assert được modified từ ví dụ trên thành XNOR sau khi đảo tín hiệu a

```
module concurrent_assertion_ex1;
    bit a, b;
    bit clk;

    always #10 clk = ~clk;

    initial begin
        for (int i = 0; i < 20; i++) begin
            @(posedge clk);
            a <= $random;
            b <= $random;
            $display("[%0t] a=%0b b=%0b", $time, a, b);
        end
        #10 $finish;
    end

    check_1: assert property (@(posedge clk) !(!a ^ b));
endmodule
```

Ví dụ kết quả:



Time (ns)	a	b	Expression !(!a ^ b)	Result
10	0	0	0	FAIL
30	0	1	1	PASS
50	1	1	0	FAIL
70	1	1	0	FAIL
90	1	0	1	PASS
110	1	1	0	FAIL
130	0	1	1	PASS
150	1	0	1	PASS
170	1	0	1	PASS
190	1	0	1	PASS

Kết quả sau khi chạy mô phỏng vivado:

```
# run 1000ns
[10000] a=0 b=0
ERROR: Assertion failed.
Time: 10 ns Started: 10 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[30000] a=0 b=1
[50000] a=1 b=1
ERROR: Assertion failed.
Time: 50 ns Started: 50 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[70000] a=1 b=1
ERROR: Assertion failed.
Time: 70 ns Started: 70 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[90000] a=1 b=0
[110000] a=1 b=1
ERROR: Assertion failed.
Time: 110 ns Started: 110 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[130000] a=0 b=1
[150000] a=1 b=0
[170000] a=1 b=0
[190000] a=1 b=0
[210000] a=1 b=1
ERROR: Assertion failed.
Time: 210 ns Started: 210 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[230000] a=0 b=1
[250000] a=0 b=0
ERROR: Assertion failed.
Time: 250 ns Started: 250 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[270000] a=0 b=1
[290000] a=0 b=1
[310000] a=1 b=1
ERROR: Assertion failed.
Time: 310 ns Started: 310 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[330000] a=1 b=0
[350000] a=0 b=0
ERROR: Assertion failed.
Time: 350 ns Started: 350 ns Scope: /concurrent_assertion_ex1 File: /home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv Line:39
[370000] a=0 b=1
[390000] a=0 b=1
$finish called at time : 400 ns : File "/home/hao/Documents/1.KY_THUAT_THIET_KE_KIEM_TRA/SVA/SVA.srcs/sim_1/new/concurrent_assertion_ex1.sv" Line 36
```