ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL - CE213.Q12

GIẢNG VIÊN HƯỚNG DẪN: Ths. NGÔ HIẾU TRƯỜNG

BÀI TẬP CORNER AND RANDOM TEST.

Sinh viên thực hiện :

Nguyễn Đình Anh. MSSV: 23520057

Lớp CE224.Q12

MỤC LỤC

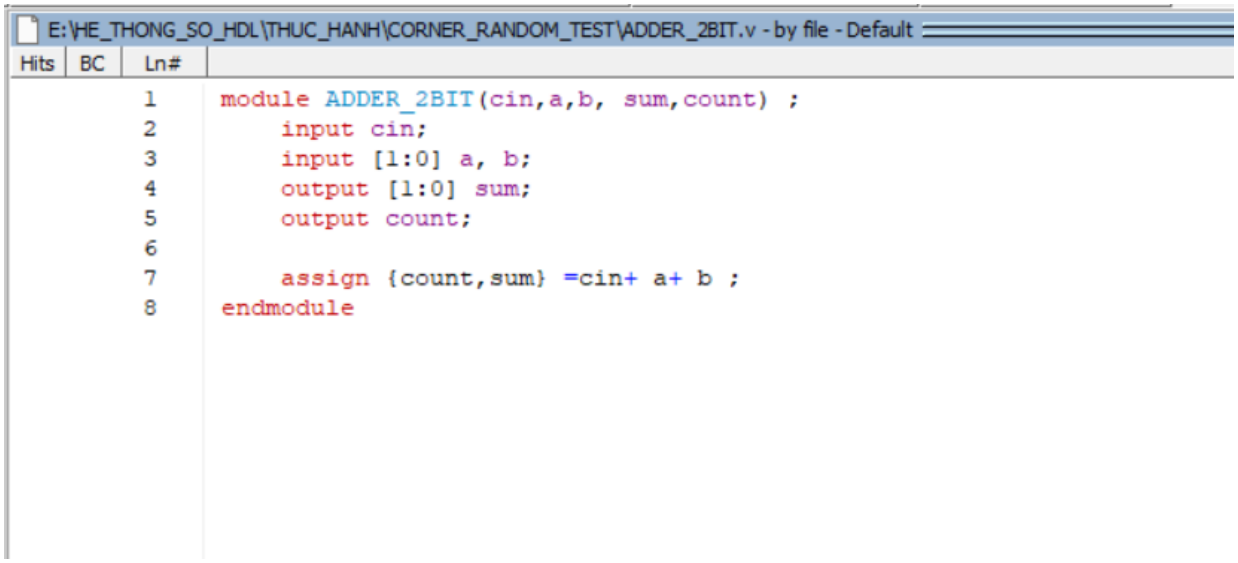
ĐỀ BÀI: Thiết kế mạch cộng 2 bít:

1. Test coverage code với 4 trường hợp

A = 00 b = 00 , A = 11 b = 11, a = 2'b01 ; b = 2'b01, a = 2'b01; b = 2'b11;

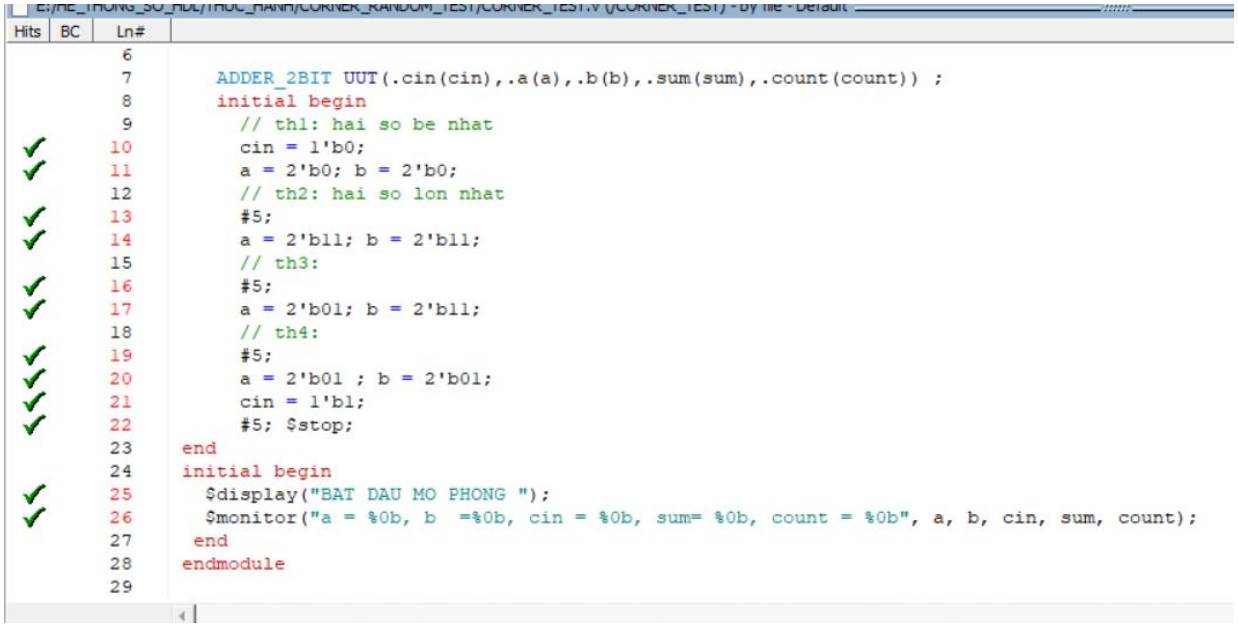
2. Chia ra 4 nhóm random test có ràng buộc.

1.1. Code chương trình



Hình 1. Code chương trình

```
module ADDER_2BIT(cin,a,b, sum,count) ;
    input cin;
    input [1:0] a, b;
    output [1:0] sum;
    output count;

    assign {count,sum} =cin+ a+ b ;
endmodule
```



Hình 2. File test bench corner test.

```
module CORNER_TEST;
  reg [1:0] a, b ;
  reg cin;
  wire [1:0] sum;
  wire count;

  ADDER_2BIT UUT(.cin(cin),.a(a),.b(b),.sum(sum),.count(count)) ;
  initial begin
    // th1: hai so be nhat
    cin = 1'b0;
    a = 2'b0; b = 2'b0;
    // th2: hai so lon nhat
    #5;
    a = 2'b11; b = 2'b11;
    // th3:
    #5;
    a = 2'b01; b = 2'b11;
```

```
    // th4:
    #5;
    a = 2'b01 ; b = 2'b01;
    cin = 1'b1;
    #5; $stop;
end
initial begin
  $display("BAT DAU MO PHONG ");
  $monitor("a = %0b, b  =%0b, cin = %0b, sum= %0b, count = %0b", a, b, cin, sum, count);
 end
endmodule
```

```
Coverage Report Summary Data by file

File: ADDER_2BIT.v
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                            1         1         0     100.0
    Branches                         0         0         0     100.0
    FEC Condition Terms              0         0         0     100.0
    FEC Expression Terms             0         0         0     100.0
    FSMs                                                     100.0
        States                       0         0         0     100.0
        Transitions                  0         0         0     100.0
    Toggle Bins                     10         7         3      70.0

File: CORNER_TEST.v
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           17        17         0     100.0
    Branches                         0         0         0     100.0
    FEC Condition Terms              0         0         0     100.0
    FEC Expression Terms             0         0         0     100.0
    FSMs                                                     100.0
        States                       0         0         0     100.0
        Transitions                  0         0         0     100.0
    Toggle Bins                     16        12         4      75.0


Total Coverage By File (code coverage only, filtered view): 86.5%
```

Hình 3. Báo cáo về coverage code.

1.2. Giải thích về độ coverage code trong báo cáo

- File adder_2bit.v

- Hàng đầu tiên là stmts : statements:

Ý nghĩa: có 1 dòng lệnh (active =1) là dòng assign {count, sum} = a + b + cin; và test bench đã chạy qua nên hits =1

- Branches /fsm : thiết kế không có lệnh rẻ nhánh, case hay fsm nên active 0;
- Toggle pin:

Ý nghĩa: là đo lường xem mõi tín hiệu trong thiết kế đã chuyển trạng thái 0->1 hay 1-> 0 hay chưa.

Active = 10: tức là đang xem 10 mục chuyển đổi 5 tín hiệu đầu vào (5bits) đồng nghĩa với 10 mục cần theo dõi (mỗi tín hiệu có 2 mục 0->1 và 1->0).

Hit =7, miss =3 : có nghĩa là testbench chỉ kích hoạt được 7 trong 10 mục, còn 3 mục chuyển đổi bị bỏ lỡ.

- File CORNER_TEST.V

- Toggle pin =75 %

Active =16: testbench có 8 tín hiệu trong đó 5 tín hiệu reg và 3 tín hiệu wire.

Miss = 4 : bị bỏ lỡ 4 mục;

Giải thích như sau:

Tín hiệu cin chạy 0 -> 0 -> 0 -> 1 rồi dừng. Nó đã bỏ lỡ (miss) trạng thái 1 -> 0.

Tín hiệu a[0] chạy 0 -> 1 -> 1 -> 1 rồi dừng. Nó đã bỏ lỡ (miss) trạng thái 1 -> 0.

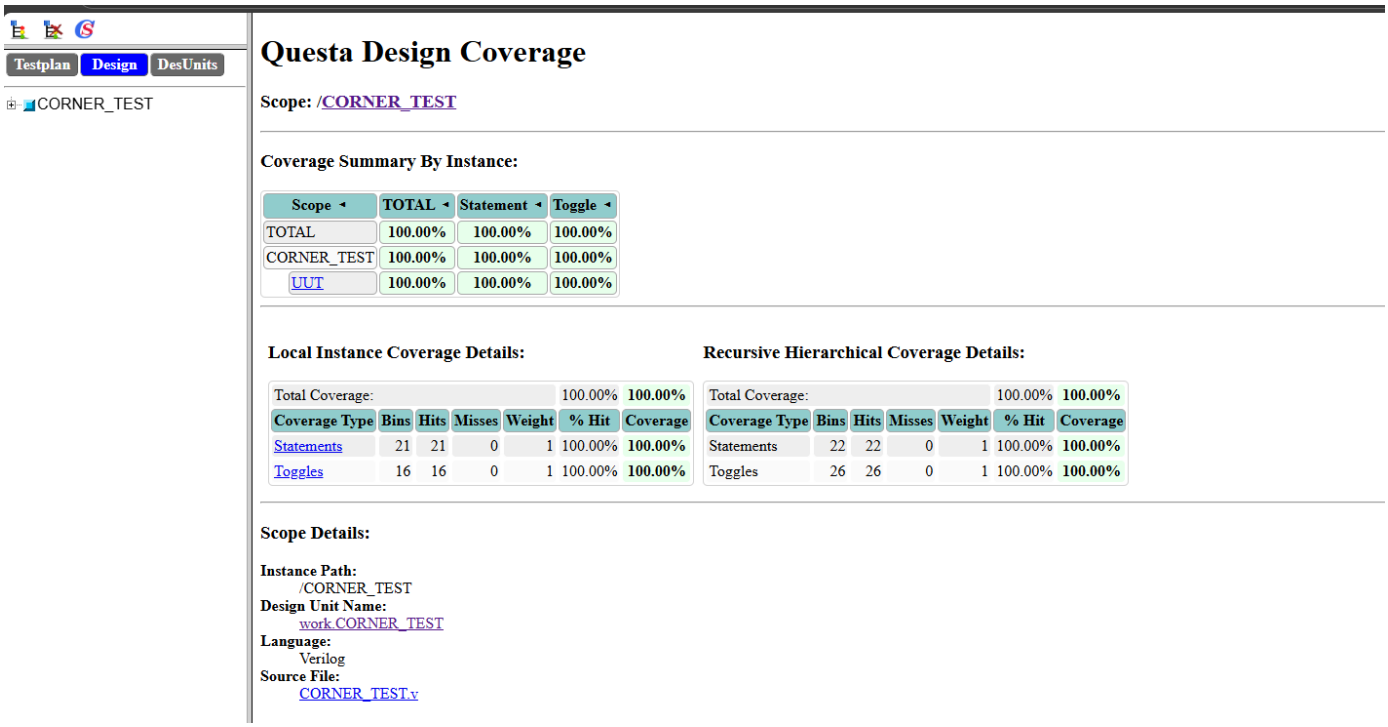
Tín hiệu b[0] chạy 0 -> 1 -> 1 -> 1 rồi dừng. Nó đã bỏ lỡ (miss) trạng thái 1 -> 0.

Vì vậy để đạt được độ bao phủ 100% thì chúng ta phải thêm trường hợp:

a = 2'b00; // a[0] chuyen tu 1 -> 0

b = 2'b00; // b[0] chuyen tu 1 -> 0

cin = 1'b0; // cin chuyen tu 1 -> 0



Testplan | Design | DesUnits

CORNER_TEST

Questa Design Coverage

Scope: /CORNER_TEST

Coverage Summary By Instance:

Scope	TOTAL	Statement	Toggle
TOTAL	100.00%	100.00%	100.00%
CORNER_TEST	100.00%	100.00%	100.00%
UUT	100.00%	100.00%	100.00%

Local Instance Coverage Details:

Total Coverage:					100.00%	100.00%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	21	21	0	1	100.00%	100.00%
Toggles	16	16	0	1	100.00%	100.00%

Recursive Hierarchical Coverage Details:

Total Coverage:					100.00%	100.00%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	22	22	0	1	100.00%	100.00%
Toggles	26	26	0	1	100.00%	100.00%

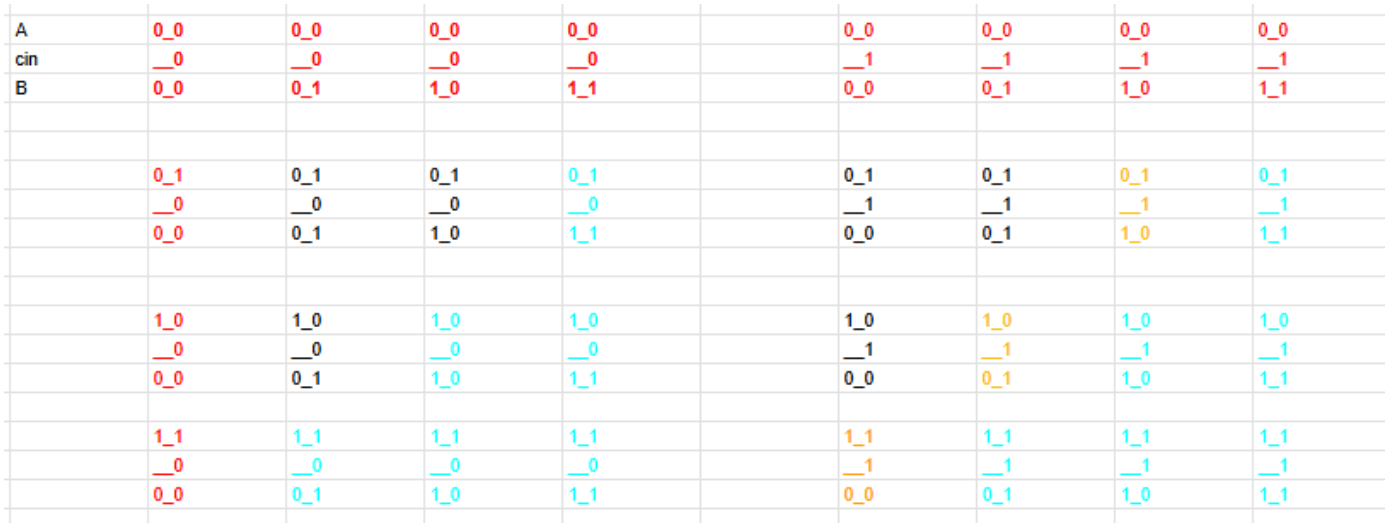
Scope Details:

Instance Path:
/CORNER_TEST
Design Unit Name:
work.CORNER_TEST
Language:
Verilog
Source File:
CORNER_TEST.v

Hình 4: Độ bao phủ khi chỉnh sửa test.

2. Yêu cầu 2: Chia ra 4 nhóm randomtest có ràng buộc và xác định độ bao phủ của nó (chỉ test 50 % mỗi nhóm).

2.1. Chiến lược chia nhóm



A	0_0	0_0	0_0	0_0		0_0	0_0	0_0	0_0
cin	__0	__0	__0	__0		__1	__1	__1	__1
B	0_0	0_1	1_0	1_1		0_0	0_1	1_0	1_1
	0_1	0_1	0_1	0_1		0_1	0_1	0_1	0_1
	__0	__0	__0	__0		__1	__1	__1	__1
	0_0	0_1	1_0	1_1		0_0	0_1	1_0	1_1
	1_0	1_0	1_0	1_0		1_0	1_0	1_0	1_0
	__0	__0	__0	__0		__1	__1	__1	__1
	0_0	0_1	1_0	1_1		0_0	0_1	1_0	1_1
	1_1	1_1	1_1	1_1		1_1	1_1	1_1	1_1
	__0	__0	__0	__0		__1	__1	__1	__1
	0_0	0_1	1_0	1_1		0_0	0_1	1_0	1_1

Hình 5: Chiến lược chia nhóm với mỗi nhóm là 1 màu riêng

- Nhóm 1: Kiểm tra khi một hoặc cả hai toán hạng bằng 0, cin ngẫu nhiên.

nhóm 1: kiểm tra một hoặc cả 2 toán hạng bằng 0			
a	0_0	0_0	0_0
cin	__0	__0	__0
b	0_0	0_1	1_0
	0_1	0_0	0_0
	__0	__1	__1
	0_0	0_0	0_1
	1_0	0_0	
	__0	__0	
	0_0	1_1	
	1_1	0_0	0_0
	__0	__1	__1
	0_0	1_0	1_1

Hình 6.

- Nhóm 2: Kiểm tra bít Count, tức là kiểm tra tràn, Cin ngẫu nhiên.

Mục đích chính là khi tổng hai số lớn hơn 3 thì xem bít count có bật hay không.

Nhóm 2: kiểm tra tràn (count 1 khi tổng lớn hơn 3)		
0_1	1_1	1_0
__0	__0	__1
1_1	1_0	1_1
1_0	1_1	1_1
__0	__0	__1
1_0	1_1	0_1
1_0	0_1	1_1
__0	__1	__1
1_1	1_1	1_0
1_1	1_0	1_1
__0	__1	__1
0_1	1_0	1_1

Hình 7.

- Nhóm 3: Kiểm tra truyền bít nhớ.

Mục đích: khi hai số có tổng bằng 3 với cin =1 thì bít count có bật lên hay không.

)	NNhóm3: Kiểm thử truyền bit nhớ.	
a	0_1	
cin	__1	
b	1_0	
	1_0	
	__1	
	0_1	
	1_1	
	__1	
	0_0	

Hình 8.

- Nhóm 4: Các trường hợp còn lại.

).	Nhóm 4: Các trường hợp còn lại	
a	0_1	0_1
cin	__0	__1
b	0_1	0_0
	1_0	1_0
	__0	__1
	0_1	0_0
	0_1	0_1
	__0	__1
	1_0	0_1

Hình 9.

2.2. Kết quả độ coverage code trên phần mềm.

Ràng buộc: Mỗi nhóm lấy ra 50 % số lượng test case và kiểm tra độ coverage code của nó. Ở đây, mình lấy 50 % test case đầu tiên của mỗi nhóm.

2.2.1. File testbench

```
module CORNER_TEST;
  reg [1:0] a, b ;
  reg cin;
  wire [1:0] sum;
```

```
  wire count;

  ADDER_2BIT UUT(.cin(cin),.a(a),.b(b),.sum(sum),.count(count)) ;
  initial begin
   // ====================== NHOM 1: KIEM TRA MOT HOAC CA HAI TOAN HANG
BANG 0============================
   // CIN NGAU NHIEN
   a = 2'b00; b = 2'b00; cin  = 1'b0;
   #5;

   a = 2'b00; b = 2'b01; cin  = 1'b0;
   #5;

   a = 2'b00; b = 2'b10; cin  = 1'b0;
   #5;

   a = 2'b01; b = 2'b10; cin  = 1'b0;
   #5;
   a = 2'b00; b = 2'b01; cin  = 1'b1;
   #5;

   // ======================NHOM 2: KIEM TRA TRAN, CIN NGAU NHIEN
============================

   a = 2'b01; b = 2'b11; cin  = 1'b0;
   #5;

   a = 2'b11; b = 2'b10; cin  = 1'b0;
   #5;
```

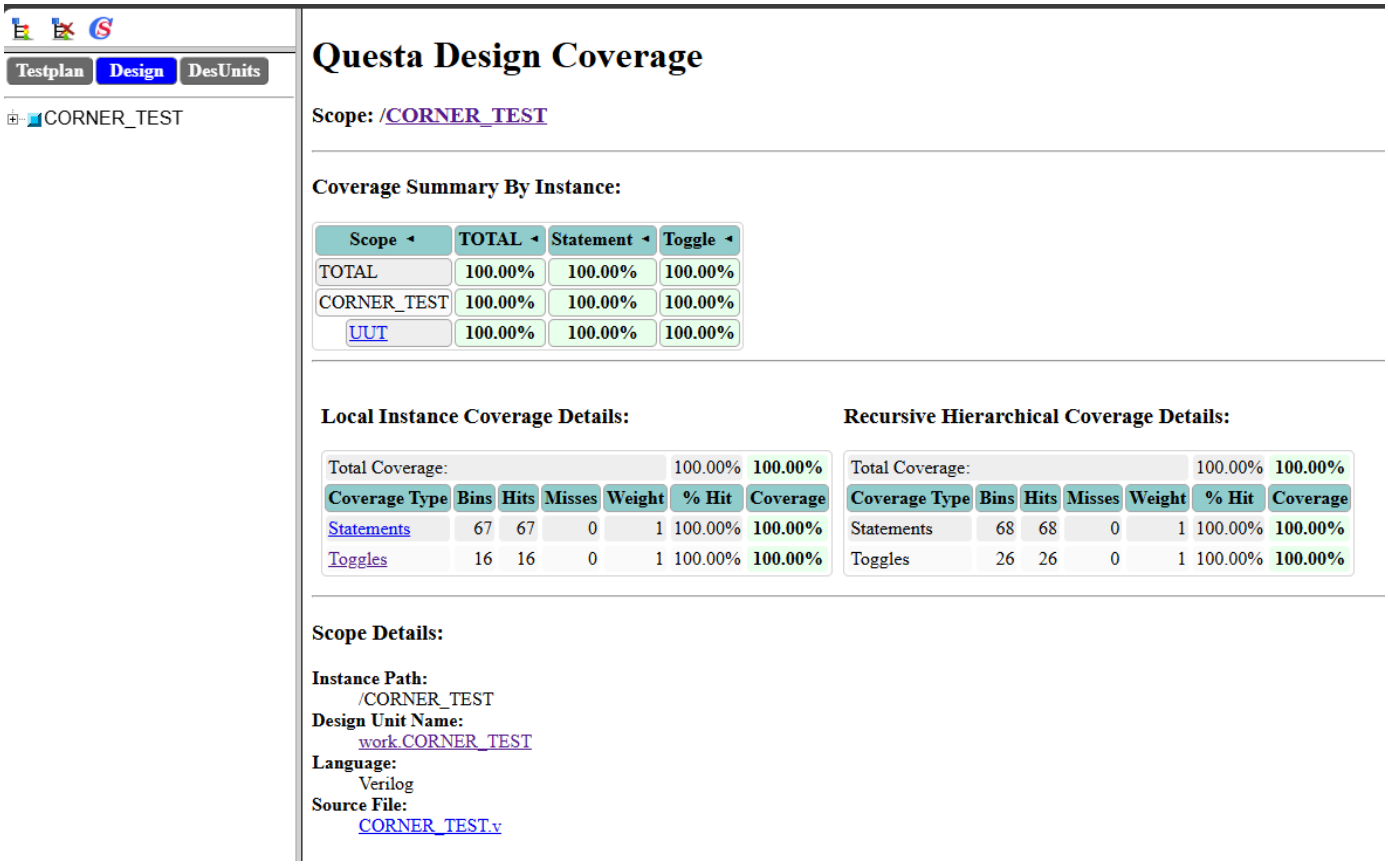
```
    a = 2'b10; b = 2'b1; cin  = 1'b1;
    #5;

    a = 2'b10; b = 2'b10; cin  = 1'b0;
    #5;

    a = 2'b11; b = 2'b11; cin  = 1'b0;
    #5;

    a = 2'b11; b = 2'b01; cin  = 1'b1;
    #5;
    // ======================NHOM 3: KIEM TRA BIT NHO
==========================================

    a = 2'b01; b = 2'b10; cin  = 1'b1;
    #5;
    a = 2'b10; b = 2'b01; cin  = 1'b1;
    #5;


     // ======================NHOM 4: CAC TRUONG HOP CON LAI
==================================

    a = 2'b01; b = 2'b01; cin  = 1'b0;
    #5;

    a = 2'b01; b = 2'b0; cin  = 1'b1;
    #5;
```

```
    a = 2'b10; b = 2'b01; cin  = 1'b0;
    #5;
     $stop;
end
initial begin
  $display("BAT DAU MO PHONG ");
  $monitor("a=%0b,\tb=%0b,\tcin=%0b,\tsum=%0b,\tcount=%0b", a, b, cin, sum, count);
 end
endmodule
```

2.2.2. Đánh giá coverage.

```
# Coverage Report Summary Data by File
#
# File: ADDER_2BIT.v
#     Enabled Coverage            Active      Hits    Misses % Covered
#     ----------------            ------      ----    ------ ---------
#     Stmts                            1         1         0     100.0
#     Branches                         0         0         0     100.0
#     FEC Condition Terms              0         0         0     100.0
#     FEC Expression Terms             0         0         0     100.0
#     FSMs                                                     100.0
#         States                       0         0         0     100.0
#         Transitions                  0         0         0     100.0
#     Toggle Bins                     10        10         0     100.0
#
# File: CORNER_TEST.v
#     Enabled Coverage            Active      Hits    Misses % Covered
#     ----------------            ------      ----    ------ ---------
#     Stmts                           67        67         0     100.0
#     Branches                         0         0         0     100.0
#     FEC Condition Terms              0         0         0     100.0
#     FEC Expression Terms             0         0         0     100.0
#     FSMs                                                     100.0
#         States                       0         0         0     100.0
#         Transitions                  0         0         0     100.0
#     Toggle Bins                     16        16         0     100.0
#
#
# Total Coverage By File (code coverage only, filtered view): 100.0%
```

Hình 10. Report trên console



Hình 11. Report trên file index.html

➔ Total coverage = 100 %, tức là test bench đã đi qua và thực thi hết trong code RTL của thiết kế.

```
# coverage save -onexit -directive -codeAll report.ucdb
#  run -all
# BAT DAU MO PHONG
# a=0,  b=0,    cin=0,  sum=0,  count=0
# a=0,  b=1,    cin=0,  sum=1,  count=0
# a=0,  b=10,   cin=0,  sum=10, count=0
# a=1,  b=10,   cin=0,  sum=11, count=0
# a=0,  b=1,    cin=1,  sum=10, count=0
# a=1,  b=11,   cin=0,  sum=0,  count=1
# a=11, b=10,   cin=0,  sum=1,  count=1
# a=10, b=1,    cin=1,  sum=0,  count=1
# a=10, b=10,   cin=0,  sum=0,  count=1
# a=11, b=11,   cin=0,  sum=10, count=1
# a=11, b=1,    cin=1,  sum=1,  count=1
# a=1,  b=10,   cin=1,  sum=0,  count=1
# a=10, b=1,    cin=1,  sum=0,  count=1
# a=1,  b=1,    cin=0,  sum=10, count=0
# a=1,  b=0,    cin=1,  sum=10, count=0
# a=10, b=1,    cin=0,  sum=11, count=0
# Break in Module CORNER_TEST at CORNER_TEST.v line 59
VSIM 23> vcover report report.ucdb
```

Hình 12. Kết quả của các test case.