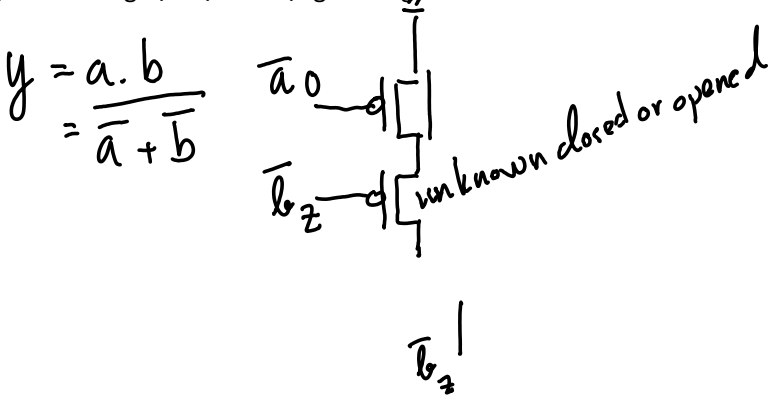
09/09/2013

Monday, September 09, 2013 1:28 PM

2.1. Khái niệm trong mô phỏng mạch số (Mô phỏng dựa trên sự kiện; logic 4 giá trị; thời gian trong mô phỏng) – 1 LT
2.1.1. Định nghĩa mô phỏng mạch số: Là quá trình phân tích mô hình mô tả mạch để tính toán giá trị đầu ra từ đầu vào
2.1.2. Nguyên tắc mô phỏng mạch số
2.1.4. Logic 4 trạng thái trong mô phỏng

Các tín hiệu trong mạch được mô phỏng có thể nhận 4 giá trị: 0, 1, x(unknown), z (high-impedance) và các phần tử trong mạch sẽ được tính toán dựa trên bảng sự thật mở rộng cho logic 4 trạng thái



a	y=NOT(a)
0	1
1	0
x	x
z	x

y = a & b	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

Bài tập ví dụ: Viết module test các phép toán logic 4 giá trị trong Verilog

Chữa:

```
module test_logical_operators;

reg a,b;
// Cách 1: Dùng hàm $monitor và gán giá trị các biến
initial $monitor("%b & %b = %b",a, b, a&b);
initial $monitor("%b | %b = %b",a, b, a|b);
initial
begin
        a=0;b=0;#5;
        a=0;b=1'b1;#5;
        a=0;b=1'bx;#5;
        a=0;b=1'bz;#5;
        ....
end
// Cách 2: Dùng hàm $display và các phép toán trực tiếp các hằng số
$display("%b & %b = %b",1'b0,1'b0,1'b0 & 1'b0);
$display("%b & %b = %b",1'b0,1'b1,1'b0 & 1'b1);
$display("%b & %b = %b",1'b0,1'bx,1'b0 & 1'bx);

endmodule
```

2.2. Testbench (Khái niệm; Chức năng; Các thành phần cơ bản của testbench; Phương pháp tạo kích thích đầu vào; Phương pháp kiểm tra đầu ra) – 1LT

2.2.1. Khái niệm và chức năng của testbench

- Testbench là mô hình môi trường hoạt động của vi mạch được sử dụng để kiểm tra thiết kế mạch bằng phương pháp mô phỏng
- Testbench nó có 3 chức năng chủ yếu
    - **Tạo ra các giá trị đầu vào** (input stimulus, input patterns) và đưa các tín hiệu đầu vào tới cổng vào của thiết kế (theo đúng kịch bản hoạt động, theo đúng giao thức vào yêu cầu). Ví dụ: testbench sẽ tạo ra các chuỗi lệnh được mã hóa đúng theo MIPS ISA và đưa vào đầu vào Instr của thiết kế khi được yêu cầu.
    - **Quan sát giá trị đầu ra, tự động tạo ra giá trị đầu ra đúng và kiểm tra giá trị đầu ra của thiết kế** (so sánh với giá trị đầu ra đúng)
    - **Giám sát quá trình mô phỏng và đo lường mức độ hoàn thành** (chất lượng của quá trình mô phỏng)

2.2.2. Các thành phần cơ bản của testbench

Testbench tối thiểu sẽ gồm các thành phần cơ bản sau:

1) Tạo ra thiết kế cần được kiểm tra (Design Under Test, Design Under Verification) và kết nối với các biến được khai báo bên trong testbench
2) Quan sát và kiểm tra các giá trị đầu ra
3) Tạo các giá trị đầu vào
4) [Tùy chọn]-Đo lường mức độ hoàn thành của quá trình mô phỏng

```
module full_adder_testbench;  // khai bao module testbench khong co dau vao/dau ra

    reg a_sim, b_sim, ci_sim; // khai báo các biến sử dụng trong testbench
    wire s_sim, co_sim; // khai báo các biến sử dụng trong testbench

    // tạo ra một thành phần có kiểu là thiết kế full_adder_structure tên là fa_duv trong
    testbench

    // module instantiation
    // nối tín hiệu tên a_sim, b_sim, ci_sim, s_sim, co_sim vào các đầu vào a, b, ci, s, co tương
    ứng của thiết kế
    full_adder_structure fa_duv(.a(a_sim), .b(b_sim), .ci(ci_sim), .s(s_sim), .co(co_sim));

    // đưa ra giá trị đầu ra để kiểm tra
    initial $monitor("%t: a=%b,b=%b,ci=%b,s=%b,co=%b",$time,
    a_sim,b_sim,ci_sim,s_sim,co_sim);

    // tạo ra mẫu giá trị đầu vào
     initial
          begin
                    // sau 5 đơn vị thời gian mô phỏng thì thay đổi 1 mẫu giá trị đầu vào
                    #5 a_sim = 0; b_sim = 0; ci_sim = 0;
                    #5 a_sim = 1;
          end
endmodule // full_adder_testbench
```

2.2.3. Các phương pháp tạo kích thích đầu vào (mẫu đầu vào)

a) Tạo ra giá trị đầu vào trực tiếp (Direct test): Tạo ra một vài giá trị tổ hợp đầu vào và đưa vào mạch.

- Dễ hiểu,

- Dễ tạo ra giá trị đầu vào, nhanh tạo được testbench
- Có khả năng không phát hiện được lỗi ở góc (corner cases)
- Nhược điểm:
    - Không bao quát hết các trường hợp (miss corner cases)
    - Không mô phỏng được toàn bộ các tổ hợp đầu vào cho mạch lớn
    - Khó duy trì testbench

b) Tạo ra tất cả các tổ hợp giá trị đầu vào (Full test-Complete test/simulation)

- Sử dụng vòng **for** trong Verilog.
- Chậm
- Không khả thi
- Sử dụng cho các mạch nhỏ
- Đảm bảo chắc chắn phát hiện được các lỗi của mạch

c) Tạo ra các tổ hợp giá trị đầu vào ngẫu nhiên (Random test)
Sử dụng hàm hệ thống **$random** trong Verilog

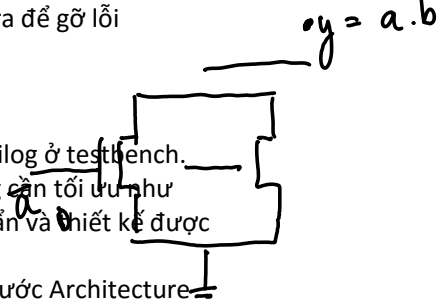
- Dễ tạo ra testbench
- Có thể tạo ra các tín hiệu đầu vào không hợp lệ
- Có khả năng không phát hiện được lỗi ở góc

d) Kết hợp phương pháp đầu vào ngẫu nhiên và đầu vào trực tiếp cùng với quá trình đo lường chất lượng mô phỏng

B1: Mô phỏng ngẫu nhiên mạch
B2: Đo chất lượng mô phỏng
B3: Phân tích các kịch bản hoạt động của mạch để tạo ra các trường hợp đầu vào trực tiếp tương ứng với các kịch bản hoạt động của mạch để đạt được chất lượng mô phỏng tốt hơn

2.2.4. Các phương pháp quan sát và kiểm tra giá trị đầu ra

a) Phương pháp thủ công
    - Giá trị tín hiệu đầu vào và đầu ra được hiển thị dưới dạng đồ thị (waveform) hoặc dạng text và được so sánh đối chiếu thủ công bởi kỹ sư kiểm tra
    - Không mất thời gian công sức để xây dựng testbench
    - Mất nhiều công sức để so sánh đối chiếu
    - Chỉ có thể áp dụng cho số lượng rất ít tổ hợp giá trị đầu vào
    - Chỉ áp dụng để gỡ lỗi
    - Cần được thay thế bằng phương pháp tự động và teschbench tự kiểm tra (self-test)
b) Phương pháp tự động
    - Giá trị tín hiệu đầu ra chuẩn được tính toán bởi testbench và sẽ được so sánh với đầu ra của thiết kế. Khi có sai khác, testbench sẽ cảnh báo kỹ sư kiểm tra để gỡ lỗi
    - Mất nhiều công sức xây dựng testbench
    - Quá trình kiểm tra sẽ được thực hiện tự động
    - Giá trị đầu ra chuẩn có thể được tính toán bằng
        - C1: Tạo ra một mô hình khác của thiết kế bằng ngôn ngữ Verilog ở testbench. Mô hình này ở mức độ trừu tượng cao hơn thiết kế và không cần tối ưu như thiết kế để làm mô hình chuẩn (golden model). Mô hình chuẩn và thiết kế được cùng mô phỏng và đầu ra sẽ được so sánh với nhau
        - C2: Mô hình phần mềm của thiết kế (thường được tạo ra ở bước Architecture Design) sẽ được sử dụng làm mô hình chuẩn
            - Testbench sẽ giao tiếp với mô hình phần mềm chuẩn thông qua giao tiếp tập tin (đọc kết quả chuẩn được ghi bởi mô hình phần mềm) hoặc thông qua giao diện lập trình PLI của Verilog (gọi các hàm tính toán C/C++) hoặc thông qua giao tiếp Hardware In Loop (đồng mô phỏng các hàm Matlab/Simulink)



2.2.5. Kiểm soát chất lượng mô phỏng

Chất lượng mô phỏng được đo lường/đánh giá thông qua tham số mức độ bao phủ (coverage; tính bằng %). Mục tiêu cơ bản của quá trình mô phỏng là đạt được mức độ bao phủ 100%. Mức độ bao phủ không đạt 100% có nghĩa là có những khu vực của thiết kế chưa được mô phỏng, và cần được mô phỏng bằng các tổ hợp đầu vào trực tiếp hoặc phải được giải thích bằng các giá trị đầu vào cấm. **Chú ý, mức độ bao phủ 100% không đảm bảo là thiết kế không có lỗi.**

Độ bao phủ chỉ có thể chỉ ra rằng mô phỏng chưa hoàn thành, chứ không chỉ ra mô phỏng đã hoàn thành.

Các loại độ bao phủ

1) Độ bao phủ mã: Được tính toán dựa trên các thành phần trong mã thiết kế và được tính toán tự động bởi chương trình mô phỏng