28/10/2013

Monday, October 28, 2013 1:39 PM

3.2.2. Mô hình hành vi

a) Khối lệnh always
b) Câu lệnh gán tuần tự (blocking)
c) Câu lệnh điều kiện if/else
d) Câu lệnh lựa chọn case
e) Câu lệnh lặp
f) Chương trình con

- Chương trình con được sử dụng để đóng gói một đoạn mã và sử dụng lại đoạn mã đó nhiều lần.
- 2 loại chương trình con trong Verilog là task và function
  - task không trả về giá trị và được sử dụng như một câu lệnh thủ tục trong khối always hoặc initial
  - function trả về 1 giá trị vô hướng và được sử dụng trong biểu thức tính toán
- Cú pháp
  - task:

```
task tên_task;
    khai_báo_output;
    khai_báo_input;
    khai_báo_biến;
    begin
        lệnh_thủ_tục;
    end
endtask
```

  - function:

```
function kiểu_giá_trị_trả_về tên_function;
    khai_báo_input;
    khai_báo_biến;
    begin
        lệnh_thủ_tục;
        tên_function = giá_trị_trả_về;
    end
endfunction
```

- Tổng hợp: Task và Function sẽ được tổng hợp thành khối logic tổ

hợp với các đầu vào được chỉ ra trong khai_báo_input và đầu ra chỉ ra trong khai_báo_output (trường hợp task) và 1 đầu ra có kiểu_giá_trị_trả_về (trường hợp function). Để có thể tổng hợp được, task và function không được chứa các câu lệnh điều khiển thời gian và sự kiện (@, wait)

- Ví dụ:
  - alu

3.3. Mô tả các phần tử nhớ Latch, Flip-flop theo sườn và theo mức (Mô tả phần tử chốt theo mức dùng phép gán liên tục; Mô tả phần tử Flip-flop theo sườn dùng cấu trúc always và phép gán non-blocking; Khái niệm về sườn đồng hồ; Mô tả tín hiệu khởi tạo Flip-flop không đồng bộ;So sánh phép gán non-blocking và phép gán blocking)-2 LT

3.3.1. Mô tả latch

- Khi biến ở bên tay trái phép gán được sử dụng ở phép toán bên tay phải phép gán liên tục thì phần tử chốt (latch) sẽ được tạo ra.
- Ví dụ:

```
assign d_latch = (en==1)?d_in:d_latch;
```

3.3.2. Mô tả flip-flop

- Sử dụng cấu trúc always với sự kiện là sườn dương hoặc sườn âm của đồng hồ.
- Cú pháp

```
always @(posedge clk)
begin
    d_ff <= d_in;
end

always @(negedge clk)
begin
    d_ff <= d_in;
end
```

- Chú ý: Trong khối always có đồng hồ/xung nhịp thường sử dụng phép gán song song (non-blocking <=). Phép gán song song trong cùng khối always được thực hiện song song:
  - Khi thực hiện mô phỏng, phép toán bên phải phép gán sẽ được tính toán trước; sau đó giá trị tính toán được gán cùng lúc cho các biến bên trái phép gán
  - Khi tổng hợp, các biến ở bên phải phép gán nhận giá trị hiện tại (giá trị tại đầu ra Q của Flip-flop)
- Ví dụ so sánh phép gán non-blocking và phép gán blocking
- Tham khảo: Clifford E. Cummings, "Nonblocking Assignments in Verilog Synthesis, Coding Styles That Kill!"
- Chú ý: Các phép gán non-blocking và blocking trong khối mô tả mạch tổ hợp

có thể hoạt động khác nhau khi mô phỏng, khi tổng hợp ở mức cổng và tổng hợp tới mức layout

Để đảm bảo mạch hoạt động đúng, phải mô phỏng sau khi layout-mô phỏng mạch ở lớp cổng với các thông tin đầy đủ về độ trễ của các cổng logic và độ trễ dây dẫn.

Ngoài ra có thể hạn chế sự sai khác (dẫn tới lỗi) khi mô phỏng, thiết kế mạch bằng cách sử dụng mạch đồng bộ

3.3.3 Mô tả tín hiệu reset của flip flop

a) Reset đồng bộ (FPGA)

```
always @(posedge clk)
begin
	if (reset)
		q <= 1'b0;
	else
		q <= d;
end
```

b) Reset không đồng bộ (ASIC)

```
always @(posedge clk or posedge reset)
begin
	if (reset)
		q <= 1'b0;
	else
		q <= d;
end
```

Bài tập về nhà: Cài đặt synopsys design compiler. Mô tả các loại flip-flop d điều khiển bằng sườn dương, âm của đồng hồ, có reset đồng bộ, không đồng bộ. So sánh kích thước sau khi tổng hợp.

Kiểm tra: làm theo nhóm, gọi random theo người, tính điểm theo nhóm

	posedge clk	negedge clk
reset đồng bộ		
reset không đồng bộ, sườn âm		
reset không đồng bộ, sường dương		