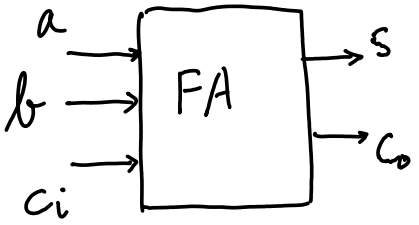
06/09/2013

Friday, September 06, 2013 3:00 PM

1.4. Giới thiệu về ngôn ngữ mô tả phần cứng HDL (Khái niệm; Ưu nhược điểm; Ví dụ đơn giản; Khái niệm về module; port; Chú thích; Khái niệm về mô hình cấu trúc, mô hình hành vi và mô hình dòng dữ liệu)

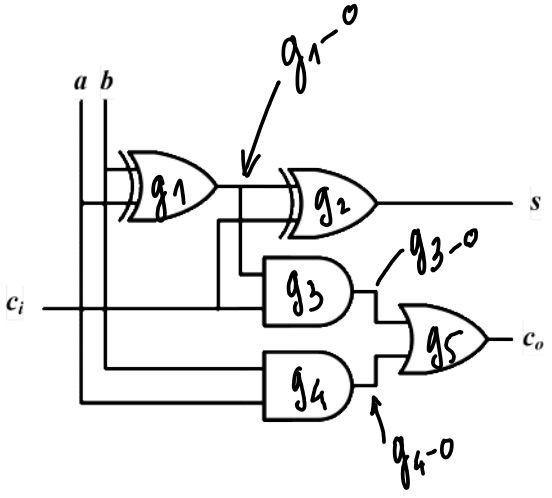
1.4.1. Khái niệm
1.4.2.Ví dụ



- Bộ cộng đủ 1 bit (Spec)
  - Đầu vào: a, b, ci
  - Đầu ra: s, co
  - Hàm truyền đạt
    - s = a xor b xor ci
    - co = a and b or a and ci + b or ci
- Verilog - HDL Description

```
module full_adder_dataflow
        (
                input a, b, ci;
                output s, co;
        )
        // Mô tả hoạt động của mạch bằng mô hinh dòng dữ liệu

        assign s = a ^ b ^ ci;
        assign co = a&b |a&ci | b&ci;
endmodule
```




```
module full_adder_dataflow
        (
                input a, b, ci;
                output s, co;
        )
        // Mô tả hoạt động của mạch bằng
        // mô hình cấu trúc

        // khai báo các dây dẫn dùng để kết nối
        // các thành phần trong mạch
        wire g1_0, g3_o, g4_o;

        // tạo ra các thành phần của mạch từ
        // kiểu cổng logic nguyên tố (primitive logic gate)

        // tạo ra 1 cổng có tên g1 kiểu xor, đầu ra g1_o
        // đầu vào a và b
        XOR g1(g1_o, a, b);
        XOR g2(s, g1_o, ci);
        AND g3(g3_o, g1_o, ci);
        AND g4(g4_o, a, b);
        OR g5(co, g3_o, g4_o);
endmodule
```

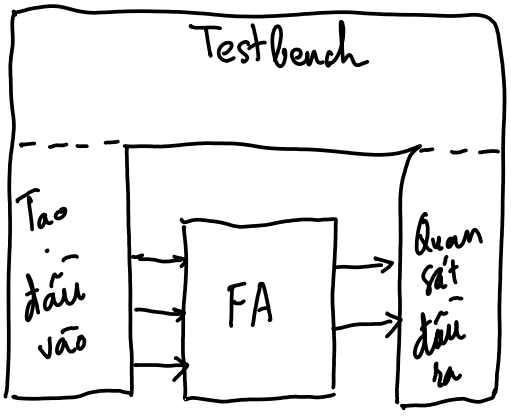
```
module full_adder_behavior
    (
        input a, b, ci;
        output s, co;
    )
    // Mô tả hoạt động của mạch bằng
    // mô hình hoạt động


    reg s, co; // khái báo biến s, co để dùng trong khối always

    always @(a or b or ci)
    begin
        if (
            ((a == 1)  && (b==1) && (ci==1)) |
            ((a == 1)  && (b==0) && (ci==0)) |
            ((a == 0)  && (b==1) && (ci==0)) |
            ((a == 0)  && (b==0) && (ci==1))
        )
            s = 1;
        else
            s = 0;
        if (
            ((a==1)  && (b==1)) |
            ((a==1)  && (ci==1)) |
            ((b==1) && (ci==1))
        )
            co = 1;
        else
            co = 0;
    end
endmodule
```

- Kiểm tra chức năng (functional verification)

  Tạo ra mô hình môi trường hoạt động của vi mạch (testbench) nhằm tạo ra đầu vào, đưa vào mạch và quan sát/kiểm tra đầu ra.




```
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

    //full_adder_behavior fa_duv(.a(a_sim), .b(b_sim), .ci(ci_sim), .s(s_sim), .co(co_sim));

    // 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));
```

```
//full_adder_dataflow 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; b_sim = 0; ci_sim = 0;
                #5 a_sim = 0; b_sim = 1; ci_sim = 0;
                #5 a_sim = 1; b_sim = 1; ci_sim = 0;
                #5 a_sim = 0; b_sim = 0; ci_sim = 1;
                #5 a_sim = 1; b_sim = 0; ci_sim = 1;
                #5 a_sim = 0; b_sim = 1; ci_sim = 1;
                #5 a_sim = 1; b_sim = 1; ci_sim = 1;
        end // initial begin
endmodule // full_adder_testbench
```

16/9/2016: Bài tập về nhà: Cài đặt modelsim, nghiên cứu cách sử dụng và mô phỏng hoạt động của mạch full-adder (thiết kế theo 3 loại mô hình khác nhau) sử dụng testbench đã cho.

- Tổng hợp thiết kế
  Q: Phần mềm tổng hợp cần các thông tin đầu vào nào
  A: Nguyễn Phi Hùng +5

1.5. Một số ví dụ về thiết kế IC số, hệ thống số (Vi xử lý 32, mạch thông tin đơn giản UART, QAM16)

Thiết kế bộ vi xử lý MIPS 32 bit

Đánh giá: Thi với 3 mức đánh giá A, B, C dựa trên diên tích mạch và tốc độ hoạt động của mạch

Thiếu 01 chức năng (thiếu lệnh): -0.5 điểm
Thêm 01 chức năng (thêm lệnh, thêm interupt handler): +0.5

Bài thuyết trình 1 (Hạn 13/9/2013): Tìm hiểu nhưng vi mạch MIPS32 có trên thị trường. Trình bày chỉ tiêu kỹ thuật của các vi mạch đó. Đề xuất chỉ tiêu kỹ thuật của vi mạch sẽ thiết kế.

1.6. Bài tập ví dụ:

Sử dụng thiết kế bộ cộng đủ để tạo ra các bộ cộng 4 bit, 8 bit, 16 bit.

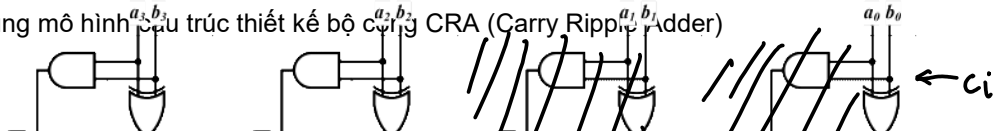
Spec:

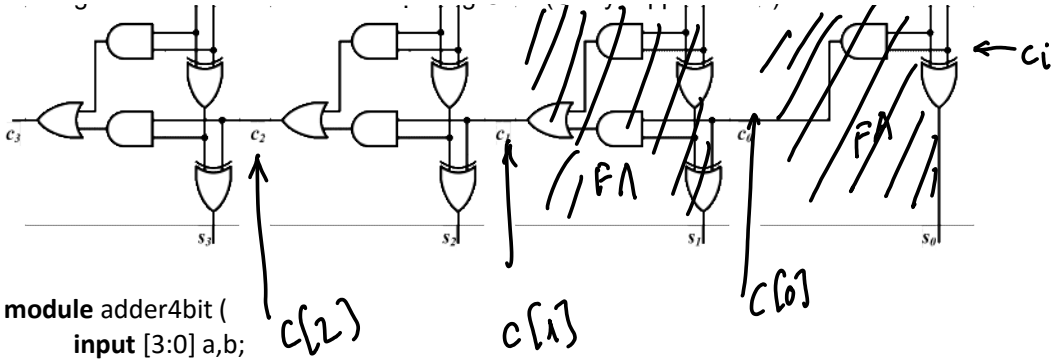
Bộ cộng 4bit:

- Input:
  - a, b kích thước 4 bit. Khai báo trong Verilog: **input** [3:0] a, b;
  - ci kích thước 1 bit. Khai báo trong Verilog: **input** ci
- Output
  - s: kích thước 4 bit. Khai báo trong Verilog: **output** [3:0] s;
  - co kích thước 1 bit. Khai báo trong Verilog: **output** co;
- Behavior
  - {co, s} = a+b+ci // dùng toán tử concat để ghép 2 bit vector

Design:

Dùng mô hình cấu trúc thiết kế bộ cộng CRA (Carry Ripple Adder)






```
module adder4bit (
        input [3:0] a,b;
        input ci;
        output [3:0] s;
        output co
        )

        wire [2:0] c;

        full_adder_dataflow FA1(.a(a[0]),.b(b[0]), .ci(ci), .s(s[0]),.co(c[0]));
        ...
endmodule
```

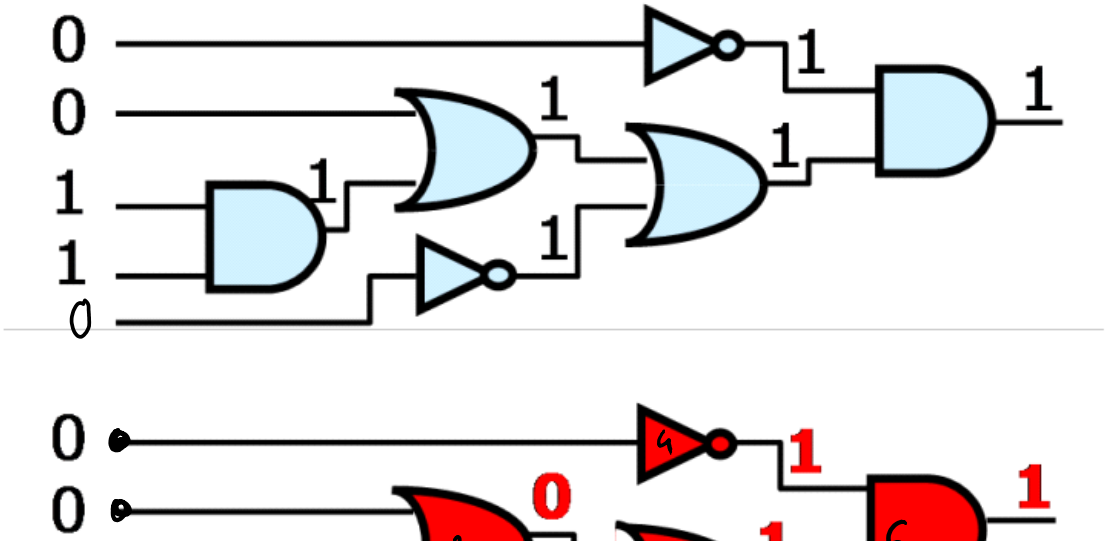
Bài tập về nhà: Hoàn thành nốt bộ cộng 4bit, 8bit, 16bit
Bài tập lựa chọn:

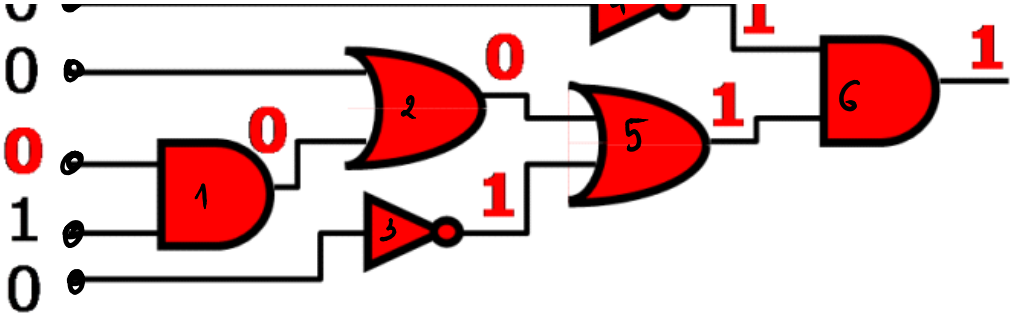
- Xây dựng bộ cộng Carry Look Ahead bằng mô hình kiến trúc Verilog
- Xây dựng bộ nhân 8bit song song (có dấu, không có dấu)

Tham khảo: ca.edabk.org (slide chương 2)

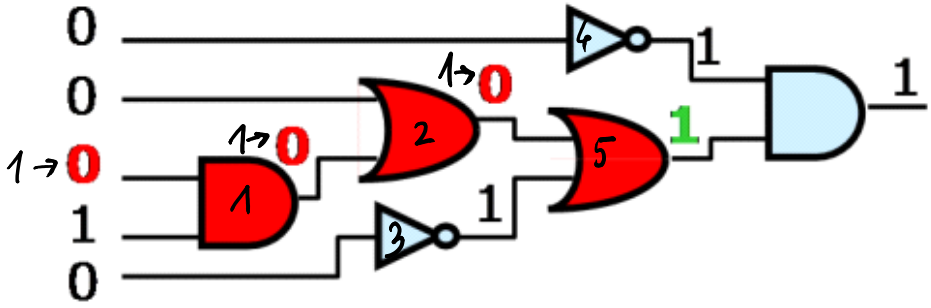
CHƯƠNG 2: Kiểm tra chức năng IC số, hệ thống số (4 LT+1BT)
2.1. Khái niệm trong mô phỏng mạch số (Mô phỏng dựa trên sự kiện; logic 4 giá trị; thời gian trong mô phỏng) – 1 LT
2.1.1. Định nghĩa mô phỏng mạch số: Là quá trình phân tích mô hình mô tả mạch để tính toán giá trị đầu ra từ đầu vào
2.1.2. Nguyên tắc mô phỏng mạch số

Khi đầu vào thay đổi giá trị, duyệt lại từng phần tử của mạch và tính toán lại đầu ra cho nó, đến khi đạt tới cổng đầu ra. Thuật toán được sử dụng là thuật toán duyệt mạch Breadth First Search.





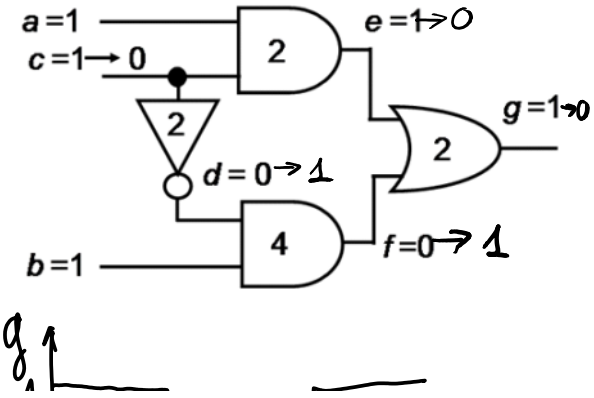
2.1.3. Mô phỏng dựa trên sự kiện



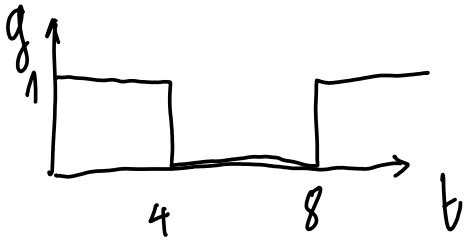
Nguyên tắc cơ bản của mô phỏng dựa trên sự kiện là chỉ tính toán lại các phần tử trong mạch khi đầu vào của phần tử thay đổi giá trị

- B1: Đưa các tín hiệu đầu vào thay đổi vào hàng đợi sự kiện tại thời gian mô phỏng 0
- B2: Lặp lại đến khi nào hàng đợi rỗng
  - Lấy một tín hiệu ra khỏi hàng đợi sự kiện
  - Với mỗi phần tử mạch (cổng logic, câu lệnh Verilog, module, lệnh gán...) là fanout(sink) của tín hiệu,
    - tính lại giá trị đầu ra tương ứng của phần tử
    - Nếu đầu ra thay đổi giá trị, đưa tín hiệu đầu ra vào hàng đợi tại thời điểm mô phỏng = thời điểm hiện tại + trễ của phần tử
- B3: Khi hàng đợi rỗng, giữ nguyên kết quả mô phỏng hoặc đi đến thời gian mô phỏng tiếp theo

thời gian	sự kiện	hoạt động tính toán
t=0	c=1->0	d=1;e=0;
t=1		
t=2	d=0->1 e=1->0	f=1 g=0
t=3		
t=4	g=1->0	do nothing
t=6	f=0->1	g=1



t=6	f=0->1	g=1
t=8	g=0->1	do nothing



Chú ý:

- Các phần tử trong mạch có thể là cổng logic, có thể là câu lệnh Verilog hoặc phép gán (đầu vào là toán hạng bên tay phải phép gán, đầu ra là biến ở tay trái phép gán), module.
- Khi độ trễ bằng 0, thì sự kiện sẽ được gán vào hàng đợi sự kiện hiện tại
- Các hoạt động tính toán ở cùng một thời điểm sẽ được thực hiện theo thứ tự không xác định. **Ta cần coi các hoạt động tính toán này xảy ra đồng thời (cách nhau thời gian \delta)**

Ưu điểm của mô phỏng sự kiện

- Tốc độ nhanh
- Mô phỏng được thời gian
- Mô phỏng được mạch mô tả bằng Verilog kiểu cấu trúc/RTL/dòng dữ liệu/hoạt động

2.1.4. Mô phỏng logic 4 trạng thái