3.1.3. Các mô hình mô tả mạch

a. Mô hình cấu trúc

- Mô hình cấu trúc sẽ miêu tả mạng các thành phần cấu trúc của mạch. Tức là mô tả các thành phần của mạch và sự kết nối giữa chúng.
- Các thành phần của mạch có thể là
    - Cổng logic nguyên tố (primitive gates): Là các cổng logic **and, or, xor, inv, buff** được khai báo sẵn trong ngôn ngữ Verilog

n-Input	n-Output, 3-state
and	buf
nand	not
or	bufif0
nor	bufif1
xor	notif0
xnor	notif1
Cổng logic có 1 đầu ra (cổng thứ nhất) và n đầu vào	Cổng logic có n đầu ra và 1 đầu vào

    - Các cổng logic nguyên tố sẽ được sử dụng trong mạch bằng cú pháp
      kiểu_cổng #giá_trị_trễ tên_cổng (danh_sách_tín_hiệu);

      Trong đó:
        - kiểu_cổng là các từ khóa **and, nand, or,** ...
        - #giá_trị_trễ: là tùy chọn chỉ ra độ trễ lan truyền khi mô phỏng của cổng. Có nghĩa là đầu ra sẽ thay đổi giá trị tại thời điểm = thời điểm hiện tại + giá_trị_trễ
        - tên_cổng: là tùy chọn tuân theo quy tắc đặt tên biến
        - danh_sách_tín_hiệu là tên các biến bắt đầu bằng các biến được nối với đầu ra và theo sau bởi các biến nối với đầu vào

      Ví dụ:
        and #5 g1(a, b, c, d); // Tạo ra một cổng AND 3 đầu vào nối với b, c, d; đầu ra nối với a. Cổng này có độ trễ 5 đơn vị mô phỏng.
- Các thành phần của mạch có thể là các module thiết kế trước. Các module con được sử dụng trong mạch theo cú pháp

  tên_module tên_instance (danh_sách_kết_nối_vào_ra);

  Trong đó:
    - tên_module là tên khi khai báo module con

- tên_instance là tùy chọn tên của đối tượng module con được sử dụng trong thiết kế
- danh_sách_kết_nối_vào_ra được chỉ ra theo
  - cách ẩn: gồm danh sách tên các biến được kết nối theo **thứ tự** tới các cổng khai báo trong module: (tên_biến_1, tên_biến_2, ...): tín hiệu tên_biến_1 sẽ được kết nối với cổng vào/ra thứ nhất trong module.
  - cách rõ ràng: chỉ ra tên biến và tên cổng được kết nối với nhau

    .tên_cổng_1 (tên_biến_1),
    .tên_cổng_2 (tên_biến_2),
    ...
  - Lời khuyên: nên sử dụng theo cách rõ ràng để tránh lỗi.

- Mảng các thành phần của mạch được tạo ra như sau
  - Cú pháp

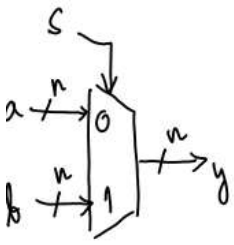
    tên_module/kiểu_cổng tên_đối_tượng [khoảng chỉ_số] (danh_sách_kết_nối_vào_ra);

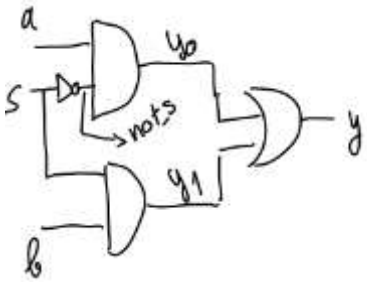
  - Với mảng thành phần thì danh_sách_kết_nối_vào_ra cần chỉ ra sự kết nối của các bit vector đại diện cho từng cổng vào ra của module được sử dụng.

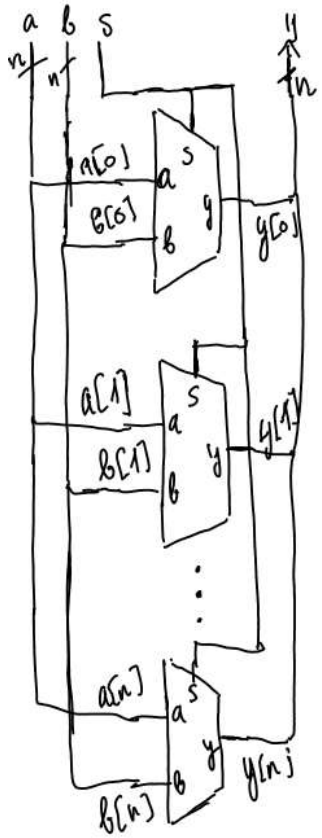
  - Cú pháp

    .tên_cổng_1 (tên_bit_vector_1),
    .tên_cổng_2 (tên_bit_vector_2),

    trong đó tên_bit_vector_1, tên_bit_vector_2 cần có kích thước bằng số thành phần được tạo ra nhân với kích thước cổng. Khi đó thành phần thứ nhất của bit_vector_1[0] sẽ được kết nối với cổng_1 của đối tượng[0] được tạo ra.
- Ví dụ: MUX 21 n-bit
-








```
module mux_21_1bit (
input a, b, s,
output y);

wire y0, y1, not_s;

not #1 g_not_s(not_s, s);
and #1 g_y0 (y0, a, not_s);
and #1 g_y1 (y1, b, s);
or #1 g_y (y, y0, y1);

endmodule
```

```
module mux_21_nbit
	#(parameter n=8)
	(
		input [n-1:0] a,b,
		input s,
		output [n-1:0] y
	);

	mux_21_1bit
		mux_21_1bit_inst[n-1:0]
			(
/* đầu vào a[0] của mux_21_nbit được kết nối với đầu vào a của đối tượng mux_21_1bit_inst[0]
*/
			.a(a),
			.b(b),
/* tạo ra n-bit vector gồm n bits từ đầu vào s của mux_21_nbit để kết nối với đầu vào s của n đối
tượng mux_21_1bit_inst[n-1:0] */
			.s({n{s}}),
			.y(y)
			);
endmodule

module test_mux;
```

```
    wire [15:0] y;
    reg      [15:0] a,b;
    reg             s;

    mux_21_nbits #(.n(16))
        duv(.a(a), .b(b),.s(s), .y(y));

    initial
        begin
            $monitor ("%t: a=%d,b=%d,s=%b,y=%d", $time, a, b, s, y);
            repeat (10)
            begin
                a=$random();
                b=$random();
                s=$random();

                #5;
            end

        end // initial begin
endmodule // test_mux
```

b. Mô hình dòng dữ liệu

- Mô hình dòng dữ liệu trong Verilog mô tả các biểu thức Bool của hàm truyền đạt của mạch.
- Mô hình dòng dữ liệu được dùng để mô tả logic tổ hợp
- Mô hình dòng dữ liệu sử dụng câu lệnh gán liên tục **assign** để gán biểu thức Bool cho 1 biến wire
- Hay được dùng để mô tả đường dữ liệu
- Mô hình dòng dữ liệu sẽ được phần mềm tổng hợp chuyển đổi thành mạch tự động
- Mô hình dòng dữ liệu sẽ đơn giản, dễ hiểu hơn mô hình cấu trúc. Tuy nhiên trong các trường hợp cần mạch tốc độ rất cao, mô hình cấu trúc vẫn được sử dụng

Ví dụ: Bộ mux n-bit

```
module mux_21_nbit
    #(parameter n=8)
    (
        input [n-1:0] a,b,
        input s,
        output [n-1:0] y
    );

    assign y = (s==0)?a:b;
```

endmodule

c. Mô hình hành vi

Mô hình hành vi mô tả hành động tính toán đầu ra của mạch khi đầu vào thay đổi giá trị.

- Sử dụng khối lệnh always và các câu lệnh thủ tục để xây dựng mô hình hành vi. Câu lệnh thủ tục: if else, case, for, repeat, phép gán
- Mô hình hành vi có thể dùng để mô tả logic tổ hợp và logic tuần tự
- Mô hình hành vi đơn giản gần với ngôn ngữ bậc cao, ngôn ngữ lập trình.
- Chú ý: Mô hình hành vi sẽ được chuyển đổi (tổng hợp) thành phần cứng nhờ phần mềm tự động; nhưng không phải bất cứ mô hình hành vi nào cũng có thể tổng hợp thành phần cứng.

Ví dụ: Bộ mux n-bit sử dụng mô hình hành vi

```
module mux_21_nbit
    #(parameter n=8)
    (
        input [n-1:0] a,b,
        input s,
        output [n-1:0] y
    );

    always @(a or b or s)
    begin
        if (s==0) y =a;
        else y=b;
    end
endmodule
```