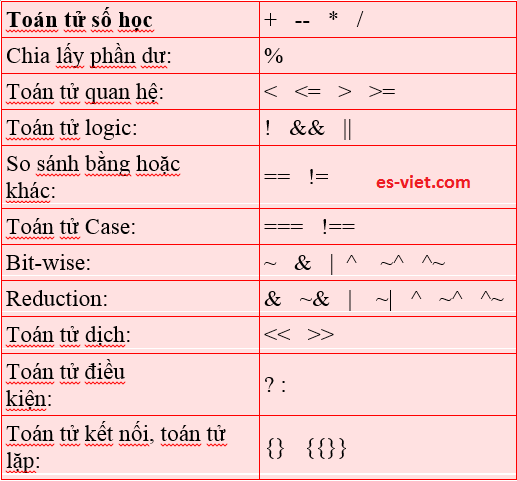
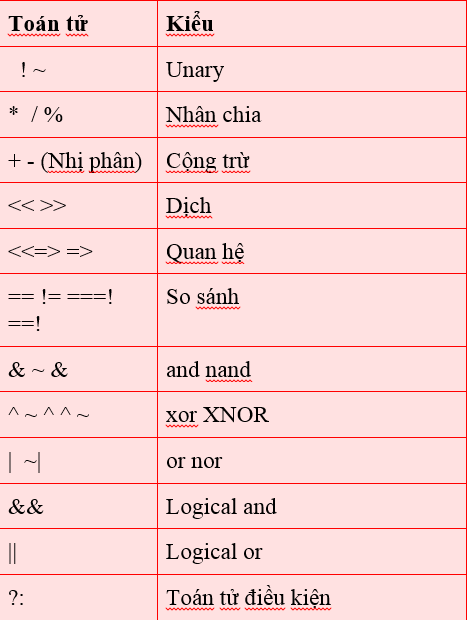
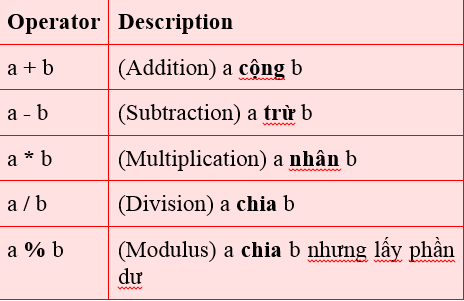
**I,Giới thiệu  
--Toán tử trong Verilog không khác trong C là mấy. Nên nếu không hiểu phần nào các bạn có thể xem phần giải thích toán tử trong C để hiểu rõ hơn.**  
  
**II,Các toán tử  
Danh sách toán tử trong verilog:**  
  
**Thứ tự ưu tiên:**  
  
  
**A, Toán tử số học**  
Bảng sau mô tả các toán tử số học trong verilog.  
  
  
**-- Toán tử modulus không được phép có biến kiểu dữ liệu thực. Toán tử khác có thể được sử dụng tất cả loại dữ liệu. Đối với toán tử module, kết quả lấy dấu của toán hạng đầu tiên.  
  
^_^\*\*ví dụ:  
Code sau giải thích việc sử dụng của một số toán tử cơ bản:**

**MÃ:**

**module operators();**

**reg [3:0] a, b;**

**initial**

**begin**

**a = 3;**

**b = 5;**

**$display(a + b ); // 8**

**$display(a ? b: 1 ); // 5**

**$display(a \* b ); // 15**

**$display(a / b ); // 0**

**$display(b / a ); // 1**

**$display(a % b ); // 3**

**$display(7 % 3 ); // 1**

**$display(7 % -3 ); // 1**

**$display(-7 % 3 ); // -1**

**$display(-7 % -3 ); // -1**

**end**

**endmodule**

**Output:**

**8**

**5**

**15**

**0**

**1**

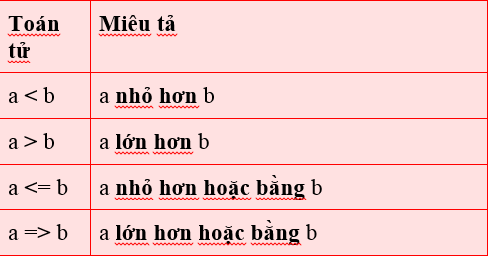
**3**

**1**

**1**

**-1**

**-1**

**B, Toán tử quan hệ  
-- Các toán tử quan hệ được sử dụng để so sánh các biểu thức. Giá trị trả về của toán tử quan hệ là 0 nếu biểu thức là false và 1 nếu biểu thức là true. Nếu mỗi toán hạng trong một toán tử quan hệ chứa giá trị không xác định (x) hoặc giá trị trở kháng cao (z), thì kết quả sẽ là một bit có giá trị không xác định (x). Các toán tử được mô tả như bảng dưới:**  
  
**-- Khi một hoặc cả hai toán hạng của một toán tử quan hệ là không dấu, biểu thức được hiểu như là so sánh giữa hai giá trị không dấu. Nếu toán hạng không bằng nhau về chiều dài bit, thì toán hạng có số bit nhỏ hơn sẽ thêm bit 0, để có độ dài bằng toán hạng có số bit lớn hơn.  
  
^_^\*\*ví dụ**

**MÃ:**

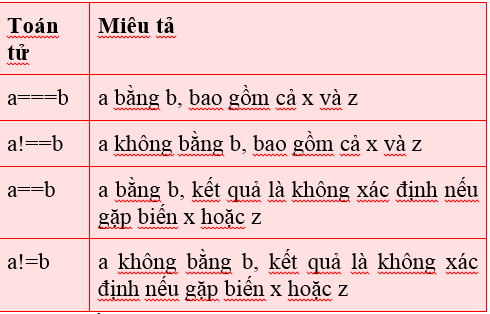
**// so sánh hai vector a, b**

**reg [3:0] a, b;**

**if (a[3] = =b [3])**

**a[2:0] >b[2:0];**

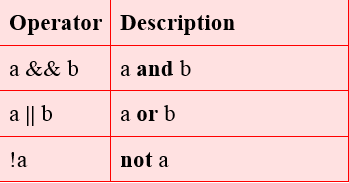
**else b[3];**

**C, Toán tử so sánh**  
  
**-- Trong toán tử == và !=, nếu toán hạng là không xác định (x) hoặc trở kháng cao (z) thì quan hệ là không xác định, và kết quả sẽ là một bit có giá trị không xác định (x).  
-- Trong toán tử === và !==, sự so sánh như là một câu lệnh *case*. Bit x hoặc z trong toán hạng sẽ được so sánh và cho kết quả là bằng nếu giống nhau. Kết quả của toán tử sẽ là một giá trị xác định 0 hoặc 1.  
^_^\*\*ví dụ**

**MÃ:**

**assign a==b;**

**assign (a!=b)==c;**

**D, Toán tử logic  
-- Các toán tử logic được sử dụng để liên kết các biểu thức.**  
  
**-- Kết quả của toán tử sẽ trả về 0 khi false, về 1 khi true và Z không xác định. Toán tử ! sẽ đảo giá trị true (1) hoặc một giá trị nào đó khác 0 thành 0. Và đảo giá trị false (0) thành 1,kết quả không rõ rang là X.  
  
^_^\*\*ví dụ:**

**MÃ:**

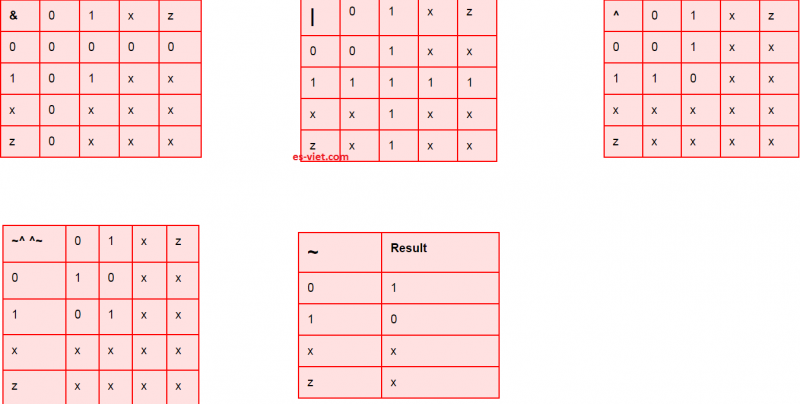
**Wire [7:0] x, y, z;**

**Reg a;**

**…**

**if ((x= = y)&&(z)) a=1;**

**else a=! x;**

**E, Toán tử bit\_wire.  
-- So sánh từng bit hai toán toán hạng,  
[](http://vimach.net/attachments/hj-png.20/)  
F, Toán tử dịch  
-- Đây là hai loại toán tử dịch, toán tử *dịch logic*(<< và >>), và toán tử dịch số học (<<< và >>>). Quy tắc hoạt động của các loại toán tử này như sau:  
\*\*\* Toán tử dịch trái: << và <<<, sẽ dịch toán hạng bên trái của chúng sang trái một số vị trí bit được đưa ra trong toán hạng bên phải. Trong cả hai trường hợp, bit ở vị trí trống sẽ được điền vào bằng số 0.  
\*\*\* Toán tử dịch phải: >> và >>>, sẽ dịch toán hạng bên trái của chúng sang phải một số vị trí bit được đưa ra trong toán hạng bên phải.  
-- Trong toán tử dịch phải logic >>: sẽ điền vào vị trí bit trống là bit 0.  
-- Trong toán tử dịch phải toán học >>>: sẽ điền vào vị trí bit trống số 0 nếu kết quả là loại không dấu, ngược lại, nếu kết quả là loại có dấu, nó sẽ điền vào vị trí bit trống, giá trị bit có ý nghĩa nhất của toán hạng bên trái.  
  
^_^\*\*ví dụ:**

**MÃ:**

**assign c = a<<2; c = a dịch trái 2 bit các chỗ trống được điền với**

**những số 0.**

**G, Toán tử điều kiện (?:)  
- Giống như C/C++. Chúng định giá một trong hai biểu thức cơ bản  
trong một điều kiện. Nó sẽ tổng hợp thành bộ đa cộng (MUX).  
- Cú pháp  
Kết quả = (điều kiện) ? kết quả khi điều kiện đúng : kết quả khi điều kiện sai.  
  
=D\*\*ví dụ:**

**MÃ:**

**assign a = (b) ? x : y;**

**assign a = (b = =2) ? x: y; // nếu b = 2 thì a=x ngược lại a=y;**

**H, Toán tử ghép nối ({})  
-- Là kết quả của việc nối các bit từ một hay nhiều biểu thức lại với nhau. Toán tử ghép nối sử dụng ký hiệu ngoặc nhọn ({}) và dùng dấu phảy (,) để ngăn cách các biểu thức. Một số hằng số không xác định kích thước, sẽ không được phép sử dụng trong toán tử ghép nối. Đó là vì kích thước của mỗi toán hạng trong toán tử ghép nối, cần phải tính toán cho phù hợp với kích thước của kết quả toán tử ghép nối.  
  
^_^\*\*ví dụ:**

**MÃ:**

**assign x = {1’b0, a}; // x[2] = 0, x[1] = a[1], x[0] = a[0].**

**assign y = {a, b}; // y[3]= a[1], y[2] = a[0], y[1] = b[1], y[0] = b[0].**

**MÃ:**

**Ví dụ:**

**module example();**

**reg [3:0] a, b;**

**reg [7:0] c, d;**

**initial**

**begin**

**a = 4'b1110; //14**

**b = 4'b0110; //5**

**$display( "%b", a < b );// false - 0**

**$display( "%b", a > 8 );// true - 1**

**$display( "%b", a <= b );// false - 0**

**$display( "%b", a >= 10 );// true - 1**

**$display( "%b", a < 4'b1zzz );// unknown - x**

**$display( "%b", b < 4'b1x01 );// unknown - x**

**a = 4'b1100;**

**b = 4'b101x;**

**$display( "%b", a == 4'b1100 ); // true - 1**

**$display( "%b", a != 4'b1100 );// false - 0**

**$display( "%b", a == 4'b1z10 );// false - 0**

**$display( "%b", a != 4'b100x );// true ? 1**

**$display( "%b", b == 4'b101x );// unknown - x**

**$display( "%b", b != 4'b101x );// unknown - x**

**$display( "%b", b === 4'b101x );// true - 1**

**$display( "%b", b !== 4'b101x );// false - 0**

**a = 4'b1100;**

**b = 4'b0000;**

**$display( "%b", !a );// 0 - false**

**$display( "%b", !b );// 1 - true**

**$display( "%b", a && b ); // 0 - false**

**$display( "%b", a || b );// 1 ? true**

**c = 8'b1010xzxz;**

**d = 8'b10010011;**

**$display( "%b", c & d ); //= 8'b100000xx;**

**$display( "%b", c | d ); //= 8'b1011xx11;**

**$display( "%b", c ^ d ); //= 8'b0011xxxx;**

**$display( "%b", c ~^ d ); //= 8'b1100xxxx;**

**$display( "%b", ~ c ); //= 8'b0101xxxx;**

**a = 4'b1111;**

**$display( "%b", a << 3 ); //= 4'b1000**

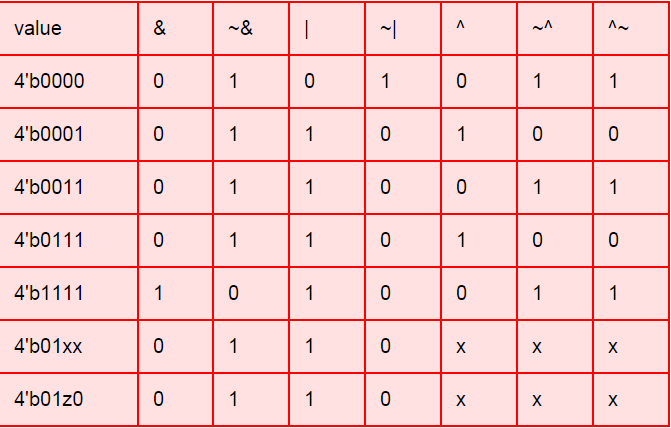
**$display( "%b", a >> 3 ); //= 4'b0001**

**$display( "%b", a << 1'bz ); //= 4'bxxxx**

**$display( "%b", a >> 1'bx ); //= 4'bxxxx**

**end**

**endmodule**

**[](http://vimach.net/attachments/u-png.21/)**