**Bài tập Verilog**

Câu 1:

Sử dụng các toán tử để theo các yêu cầu từng câu

a/ Tính toán kết quả đầu ra:

a=6’b101100 và b=6’b101011

c = a|b =

d= a^b =

e= a&b =

b/ Tính toán kết quả đầu ra:

a=12’ha4b và b=12’h101

c = a|b =

d= a^b =

e= a&b =

c/ Viết biểu thức **y1**:

* có 2 tín hiệu điều khiển x0 và x1
* x0 là tín hiệu điều khiển lựa chọn giá trị giữa a0 và a1 để gán vào tín hiệu y0
* x1 là tín hiệu điều khiển lựa chọn giá trị giữa y0 và a2 để gán vào tín hiệu y1

y1 = ...

d/ Tính phép nhân hoặc phép chia sau: gợi ý sử dụng phép dịch trái hoặc phải

Giải thích tại sao dùng phép dịch đó ?

a = 4’d12 nhân với b = 2’b10

c = 8’hA9 nhân với d = 4’d8

f = 8’hB4 chia với e = 3’b10

e/ Cho một đoạn code mô tả một toán tử như sau:

|  |
| --- |
| module operators();  reg [3:0] a, b;  initial  begin  a = ...;  b = ...;  $display(a + b );  $display(a ? b: 1 );  $display(a \* b );  $display(a / b );  $display(b / a );  $display(a % b );  $display((a>>2) % 4 );  $display((a<<3) % (b<<1) );  end  endmodule |

Trong đó:

a là tổng các số trong năm sinh cộng lại (ví dụ bạn sinh năm 1999 => a= 1+9+9+9= 28)

b là tổng các số trong tháng sinh cộng lại (ví dụ bạn sinh tháng 09 => b= 0+9= 9)

Hãy ghi kết quả mà dòng lệnh display hiển thị ra

f/ Cho đoạn mã code như sau:

|  |
| --- |
| module example();  reg [3:0] a, b;  reg [7:0] c, d;  initial  begin  a = 4'b1110;  b = 4'b0110;  $display( "%b", a < b ); // Ket qua %b = 0 (a>b is true)  $display( "%b", a > 8 );  $display( "%b", a <= b );  $display( "%b", a >= 10 );  $display( "%b", a < 4'b1zzz );  $display( "%b", b < 4'b1x01 );  a = 4'b1100;  b = 4'b101x;  $display( "%b", a == 4'b1100 );  $display( "%b", a != 4'b1100 );  $display( "%b", a == 4'b1z10 );  $display( "%b", a != 4'b100x );  $display( "%b", b == 4'b101x );  $display( "%b", b != 4'b101x );  $display( "%b", b === 4'b101x );  $display( "%b", b !== 4'b101x );  a = 4'b1100;  b = 4'b0000;  $display( "%b", !a );  $display( "%b", !b );  $display( "%b", a && b );  $display( "%b", a || b );  c = 8'b1010xzxz;  d = 8'b10010011;  $display( "%b", c & d );  $display( "%b", c | d );  $display( "%b", c ^ d );  $display( "%b", c ~^ d );  $display( "%b", ~ c );  a = 4'b1111;  $display( "%b", a << 3 );  $display( "%b", a >> 3 );  $display( "%b", a << 1'bz );  $display( "%b", a >> 1'bx );  end  endmodule |

Hãy ghi các kết quả hiển thị (dòng lệnh display hiển thị kết quả như thế nào):

g/

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Value | & | ~& | | | ~| | ^ | ~^ | ^~ |
| 6’b000000 |  |  |  |  |  |  |  |
| 6’b000001 |  |  |  |  |  |  |  |
| 6’b010001 |  |  |  |  |  |  |  |
| 6’b100111 |  |  |  |  |  |  |  |
| 6’b010101 |  |  |  |  |  |  |  |
| 6’b110111 |  |  |  |  |  |  |  |
| 6’b111111 |  |  |  |  |  |  |  |
| 6’b01xx11 |  |  |  |  |  |  |  |
| 6’b00x101 |  |  |  |  |  |  |  |
| 6’b01zx0z |  |  |  |  |  |  |  |
| 6’b01z0z1 |  |  |  |  |  |  |  |
| 6’b0zx1zx |  |  |  |  |  |  |  |

Hãy điền các kết quả vào ô màu cam khi sử dụng các toán tử rút gọn (xem lại slide bài giảng về toán tử rút gọn) cho các giá trị ở cột Value

Câu 2:

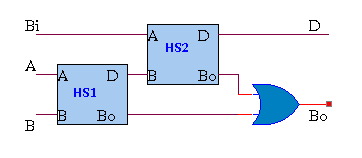
Tạo bộ cộng 2 bit dùng Half Adder với mô tả như sau:

* Bộ cộng cho 2 số, mỗi số có giá trị 2 bit (ví dụ a=2’b11 và b=2’b10 thì kết quả phải nhận là 3’b101)
* Bộ cộng sử dụng 4 bộ half adder
* Đầu vào là 2 số, mỗi số có giá trị 2 bit, không có giá trị bit nhớ riêng lẻ
* Đầu ra là giá trị 3 bit, trong đó 2 bit dành cho giá trị tổng và 1 bit dư

Hãy vẽ ra sơ đồ bộ cộng và viết một đoạn code để mô tả nó

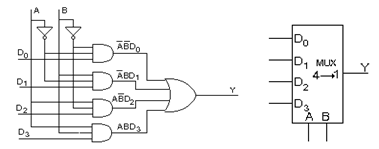
Ta có ví dụ như sau:

Bộ cộng full adder, sơ đồ hình vẽ và đoạn code mô tả:



|  |
| --- |
| module full\_adder(sum, cout, in1, in2, cin);  output sum, cout;  input in1, in2, cin;  wire sum, cout, in1, in2, cin;  wire I1, I2, I3;  half\_adder ha1(I1, I2, in1, in2);  half\_adder ha2(sum, I3, I1, cin);  assign cout = I2 || I3;  endmodule |

Câu 3: Viết lại bộ cộng MUX 4-1 theo 2 cách khác nhau:



Đoạn mẫu code sau đây:

|  |
| --- |
| module mux\_41(Y,D0,D1,D2,D3,A,B);  output Y;  input D0,D1,D2,D3,A,B ;  wire Abar, Bbar, T1, T2, T3, T4;  not (Abar, A);  not (Bbar, B);  and (T1, D0, Abar, Bbar);  and (T2, D1, Abar, B);  and (T3, D2, A, Bbar);  and (T4, D3, A, B);  or(Y, T1, T2, T3, T4);  endmodule |

Cách 1: Dùng assign và toán tử điều kiện

|  |
| --- |
| // Write code here |

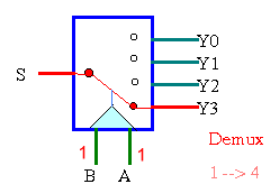
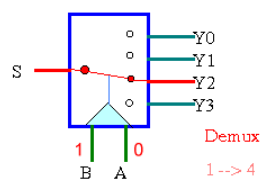
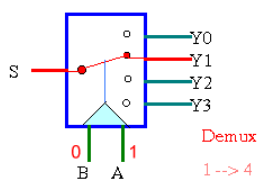
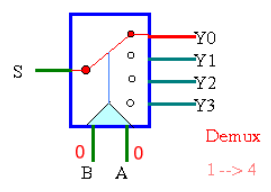
Cách 2: Dùng if ... else ... , trong đó sử dụng cách so sánh 1 bit

|  |
| --- |
| // Write code here |

Câu 4:

Ví dụ về bộ DEMUX 1-4:

Hình dưới mô tả cách hoạt động của bộ DEMUX 1-4, giá trị đầu ra sẽ phụ thuộc vào tín hiệu điều khiển A,B, sau đó sẽ chuyển giá trị S theo đầu ra tương ứng, các đầu ra còn lại bằng 0.

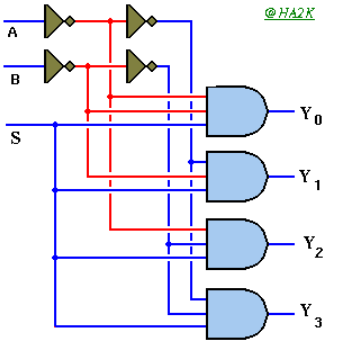


Bảng sự thật cho bộ DEMUX 1-4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | Y0 | Y1 | Y2 | Y3 |
| 0 | 0 | S | 0 | 0 | 0 |
| 0 | 1 | 0 | S | 0 | 0 |
| 1 | 0 | 0 | 0 | S | 0 |
| 1 | 1 | 0 | 0 | 0 | S |

Cấu tạo bộ DEMUX:

Từ 2 đầu A, B sẽ tách ra 4 tín hiệu: A và , B và . Sau đó dùng cổng AND 3 đầu vào: S và 2 tín hiệu chọn lựa từ 4 tín hiệu đã tách của A và B.



a/ Hãy viết lại đoạn code (mô hình dạng cổng) để mô tả lại bộ demux trên

|  |
| --- |
| // Write code here |

b/ Hãy ví dụ 3-6 trường hợp cụ thể về mối quan hệ đầu ra – đầu vào:

Ví dụ 1: A=0, B=0, S=1 => Chuỗi Y0-Y1-Y2-Y3 = 4’b1000

Câu 5:

Viết lại bộ MUX 4-1, MUX 8-1, MUX 16-1, trong đó:

a/ MUX 4-1: dùng procedure (always) để mô tả, trong đó nội dung dùng if ... else ...

|  |
| --- |
| // Write code here |

b/ MUX 8-1: dùng assign và toán tử điều kiện

|  |
| --- |
| // Write code here |

c/ MUX 8-1: dùng procedure (always) và dùng case()

|  |
| --- |
| // Write code here |

d/ MUX 16-1: dùng procedure (always) và dùng case()

|  |
| --- |
| // Write code here |