

module cla\_64bit( a, b, cin, sum, cout);

input [63:0] a, b;

input cin;

output [63:0] sum;

output cout;

wire [63:0] p, g;

wire [16:0] c;

wire [4:0] c16;

wire [15:0] gG\_4,gP\_4;

wire [3:0] gG\_16,gP\_16;

// 小 p 小 g

pg\_generator pg1(a[3:0], b[3:0], p[3:0], g[3:0]);

pg\_generator pg2(a[7:4], b[7:4], p[7:4], g[7:4]);

pg\_generator pg3(a[11:8], b[11:8], p[11:8], g[11:8]);

pg\_generator pg4(a[15:12], b[15:12], p[15:12], g[15:12]);

pg\_generator pg5(a[19:16], b[19:16], p[19:16], g[19:16]);

pg\_generator pg6(a[23:20], b[23:20], p[23:20], g[23:20]);

pg\_generator pg7(a[27:24], b[27:24], p[27:24], g[27:24]);

pg\_generator pg8(a[31:28], b[31:28], p[31:28], g[31:28]);

pg\_generator pg9(a[35:32], b[35:32], p[35:32], g[35:32]);

pg\_generator pg10(a[39:36], b[39:36], p[39:36], g[39:36]);

pg\_generator pg11(a[43:40], b[43:40], p[43:40], g[43:40]);

pg\_generator pg12(a[47:44], b[47:44], p[47:44], g[47:44]);

pg\_generator pg13(a[51:48], b[51:48], p[51:48], g[51:48]);

pg\_generator pg14(a[55:52], b[55:52], p[55:52], g[55:52]);

pg\_generator pg15(a[59:56], b[59:56], p[59:56], g[59:56]);

pg\_generator pg16(a[63:60], b[63:60], p[63:60], g[63:60]);

// 四個一組的PG會算出來，小c也會

group\_CLA\_4bit carry\_generator1 (p[3:0], g[3:0] , cin , gG\_4[0] , gP\_4[0] , , sum[3:0], );

group\_CLA\_4bit carry\_generator2 (p[7:4], g[7:4] , c[1] , gG\_4[1] , gP\_4[1] , , sum[7:4], );

group\_CLA\_4bit carry\_generator3 (p[11:8], g[11:8] , c[2] , gG\_4[2] , gP\_4[2] , , sum[11:8], );

group\_CLA\_4bit carry\_generator4 (p[15:12],g[15:12] ,c[3] , gG\_4[3] , gP\_4[3] , , sum[15:12], );

group\_CLA\_4bit carry\_generator5 (p[19:16],g[19:16] ,c[4] , gG\_4[4] , gP\_4[4] , , sum[19:16], );

group\_CLA\_4bit carry\_generator6 (p[23:20],g[23:20] ,c[5] , gG\_4[5] , gP\_4[5] , , sum[23:20], );

group\_CLA\_4bit carry\_generator7 (p[27:24],g[27:24] ,c[6] , gG\_4[6] , gP\_4[6] , , sum[27:24], );

group\_CLA\_4bit carry\_generator8 (p[31:28],g[31:28] ,c[7] , gG\_4[7] , gP\_4[7] , , sum[31:28], );

group\_CLA\_4bit carry\_generator9 (p[35:32],g[35:32] ,c[8] , gG\_4[8] , gP\_4[8] , , sum[35:32], );

group\_CLA\_4bit carry\_generator10(p[39:36],g[39:36] ,c[9] , gG\_4[9] , gP\_4[9] , , sum[39:36], );

group\_CLA\_4bit carry\_generator11(p[43:40],g[43:40] ,c[10] ,gG\_4[10] ,gP\_4[10] , ,sum[43:40], );

group\_CLA\_4bit carry\_generator12(p[47:44],g[47:44] ,c[11] ,gG\_4[11] ,gP\_4[11] , ,sum[47:44], );

group\_CLA\_4bit carry\_generator13(p[51:48],g[51:48] ,c[12] ,gG\_4[12] ,gP\_4[12] , ,sum[51:48], );

group\_CLA\_4bit carry\_generator14(p[55:52],g[55:52] ,c[13] ,gG\_4[13] ,gP\_4[13] , ,sum[55:52], );

group\_CLA\_4bit carry\_generator15(p[59:56],g[59:56] ,c[14] ,gG\_4[14] ,gP\_4[14] , ,sum[59:56], );

group\_CLA\_4bit carry\_generator16(p[63:60],g[63:60] ,c[15] ,gG\_4[15] ,gP\_4[15] , ,sum[63:60], );

// 16個一組的 P, G

group\_CLA\_4bit carry\_generator17(gP\_4[3:0], gG\_4[3:0] , , gG\_16[0] ,gP\_16[0] , ,, );

group\_CLA\_4bit carry\_generator18(gP\_4[7:4], gG\_4[7:4] , , gG\_16[1] ,gP\_16[1] , ,, );

group\_CLA\_4bit carry\_generator19(gP\_4[11:8], gG\_4[11:8] , , gG\_16[2] ,gP\_16[2] , ,, );

group\_CLA\_4bit carry\_generator20(gP\_4[15:12],gG\_4[15:12] , ,gG\_16[3] ,gP\_16[3] , ,, );

// overflow bit

// overflowBit carry\_generator26(gP\_16[3:0], gG\_16[3:0] ,cin , cout);

// get all

group\_CLA\_4bit carry\_generator21(gP\_16[3:0],gG\_16[3:0] ,cin , , ,c16[3:0] , , cout);

group\_CLA\_4bit carry\_generator22(gP\_4[3:0], gG\_4[3:0] , c16[0] , , ,c[3:0] ,, );

group\_CLA\_4bit carry\_generator23(gP\_4[7:4], gG\_4[7:4] , c16[1] , , ,c[7:4] ,, );

group\_CLA\_4bit carry\_generator24(gP\_4[11:8], gG\_4[11:8] , c16[2] , , ,c[11:8] ,, );

group\_CLA\_4bit carry\_generator25(gP\_4[15:12],gG\_4[15:12] ,c16[3] , , ,c[15:12] ,, );

endmodule

module group\_CLA\_4bit( p, g, cin, gG, gP, c, sum, out);

input [3:0] p, g;

input cin;

output gG, gP;

output [3:0] c;

output [3:0] sum;

output out;

assign gG = (g[0] & p[1] & p[2] & p[3])|(g[1]&p[2]&p[3])|(g[2]&p[3])|g[3];

assign gP = p[0] & p[1] & p[2] & p[3];

assign c[0] = cin;

assign c[1] = g[0]|(p[0] & cin);

assign c[2] = g[1]|(p[1] & g[0])|(p[1] & p[0] & cin);

assign c[3] = g[2]|(p[2] & g[1])|(p[2] & p[1] & g[0])|(p[2] & p[1] & p[0] & cin);

assign out = g[3]|(p[3] & g[2])|(p[3] & p[2] & g[1])|(p[3] & p[2] & p[1] & g[0])|(p[3] & p[2] & p[1] & p[0] & cin);

assign sum[0] = c[0] ^ p[0]; // c[0] ^ a[0] ^ b[0]

assign sum[1] = c[1] ^ p[1]; // c[1] ^ a[1] ^ b[1]

assign sum[2] = c[2] ^ p[2]; // c[2] ^ a[2] ^ b[2]

assign sum[3] = c[3] ^ p[3]; // c[3] ^ a[3] ^ b[3]

endmodule

/\*

module overflowBit( p, g, cin, cout);

input [3:0] p, g;

input cin;

output cout;

assign cout = g[3]|(p[3] & g[2])|(p[3] & p[2] & g[1])|(p[3] & p[2] & p[1] & g[0])|(p[3] & p[2] & p[1] & p[0] & cin);

endmodule

\*/

module pg\_generator( a, b, p, g);

input [3:0] a, b;

output [3:0] p, g;

assign p[0] = a[0] ^ b[0] ;

assign p[1] = a[1] ^ b[1] ;

assign p[2] = a[2] ^ b[2] ;

assign p[3] = a[3] ^ b[3] ;

assign g[0] = a[0] & b[0] ;

assign g[1] = a[1] & b[1] ;

assign g[2] = a[2] & b[2] ;

assign g[3] = a[3] & b[3] ;

endmodule