```
module cla64 (a, b, cin, sum, cout);        //64-bit carry look ahead adder: behavioral model
input [63:0] a, b;          //64-bit addends
input cin;                  //incoming carry at the lsb position
output [63:0] sum;          //64-bit sum output
output cout;                //carry output from the msb position
reg [63:0] sum;             //register corresponding to sum output port
reg cout;                   //register corresponding to cout output port
reg [63:0] g, p;            //1-bit adder level generate and propagate function values: 64 numbers each
reg [15:0] g1, p1;          //4-bit adder level generate and propagate function values: 16 numbers each
reg [3:0] g2, p2;           //16-bit adder level generate and propagate values: 4 numbers each
reg [3:0] c2;               //carry-in  values generated for the four no. 16-bit adders (L-2 components)
reg [15:0] c1;              //carry-in values generated for the sixteen no. 4-bit adders (L-1 component)
reg [63:0] c;               //carry-in values generated for the 64 no. 1-bit adders (L-0 component)
integer i, j;               //loop counters

always @ (a, b, cin)        //CLA combinational logic activation condition
begin
//hierarchy level-0: 1- bit adder component
// compute 1-bit adder level generate ‘g’ and propagate ‘p’ functions
for (i = 0; i <64; i = i + 1)
begin
g[i] = a[i] & b[i];
p[i] = a[i] ^ b[i];
end
//hierarchy level-1: 4-bit CLA adder component (made of 1-bit full adder components)
//compute 4-bit CLA adder component level generate ‘g1’ and propagate ‘p1’ functions
for (i = 0; i < 16; i = i + 4)
begin
j = i * 4;
```

```
g1[i] = g[j + 3] | p[j + 3] & g[j +2] | p[j + 3] & p[j + 2] & g[j + 1] | p[j + 3] & p[j + 2] & p[j + 1] g [j];

p1[i] = p[j + 3] & p[j + 2] & p[j + 1] & p[j];

end

//hierarchy level-2 (L-2 adder): 16-bit CLA adder (comprised of 4 x  4-bit CLA adders)

//compute L-2 adder level (16-bit CLA adder level) generate 'g2' and propagate 'p2' functions

for (i = 0; i < 4; i = i + 1)

begin

j = i * 4

g2[ i ] = g1[j + 3] | p1 [j + 3] & g1[j + 2] | p1[j + 3] & p1[j + 2] & g1 [j + 1] | p1[j + 3] & p1[j + 2] & p1[j + 1]
& g1[ j ];

p2 [ i ] = p1[j + 3] & p1[j + 2] & p1[j +1] & p1[ j ];

end

//look ahead computation of carry-in values for the

//four nos.  L-2 adder (16-bit CLA adder)

//used to realize the 64-bit adder

c2[ 0 ] = cin;

c2[ 1 ] = g2[ 0 ] |  p2[ 0 ] & c2[ 0 ];

c2[ 2 ] = g2[ 1 ] | p2[ 1 ] & g2[ 0 ] | p2[ 1 ] & p2[ 0 ] & c2[ 0 ];

c2[ 3 ] = g2[ 2 ] | p2[ 2 ] & g2[ 1 ] | p2[ 2 ] & p2[ 1 ] & g2 [ 0 ] | p2[ 2 ] & p2[ 1 ] & p2[ 0 ] & c2[ 0 ];

//compute carry-out from the most significant 16-bit adder

cout = g2[ 3 ] | p2[ 3 ] & g2[ 2 ] | p2[ 3 ] & p2[ 2] & g2[ 1 ] | p2[ 3 ] & p2[ 2 ] & p2[ 1 ] & g2 [ 0 ] |
        p2[ 3 ] & p2[ 2 ] & p2[ 1 ] & p2[ 0 ] & c2[ 0 ];

//look ahead computation of carry-in values for the

//sixteen nos. of L-1 adders (4-bit CLA adders)

for (i =0; i < 4; i = i + 1)

c1[4 * i] = c2[ i ];          //carry-in values c2[ i ] computed  above for the
                              //four nos. of L-2 adders (16-bit CLA adders) are also
                              // the carry-in values c1[4 * i] for their corresponding
                              //constituent L-1 adders (4-bit CLA adders) performing
```

```
                    //addition on their least significant four bits

//look ahead computations of carry-in values for the
//remaining twelve nos. of L-1 adders (4-bit CLA adders)
for ( i = 0; i < 16; i = i + 4)
begin
c1[i + 1] = g1[ i ] | p1[ i ] & c1[ i ];
c1[ i + 2] = g1[i + 1] | p1[i + 1] & g1[ i ] | p1[i + 1] & p1[ i ] & c1[ i ];
c1[i + 3] = g1[i + 2] | p1[i + 2] & g1[i + 1] | p1[i + 2} & p1[i + 1] & g1[ i]  | p1[i +2] & p1[i + 1] & p1[ i ] &
c1[ i ];
end
//look ahead computation of carry-in values for the sixty four L-0 adders (1-bit full adders)
for (i = 0; i < 16; i = i + 1)          //carry-in values c1[ i ] computed above for
                                        //sixteen nos. of L-1 adders (4-bit CLA adders) are also the
c[ 4 * i ] = c1[ i ];          //carry-in values c[ 4 * i ] for sixteen nos. of  L-0 adders (1-bit full adders)
//look ahead computation of carry-in values for the
//remaining forty eight L-0 adders (1-bit full adders)
for (i = 0; i < 64; i = i + 1)
begin
c[i + 1] = g[ i ] | p[ i ] & c[ i ];
c[i + 2] = g[i + 1] | p[i + 1] & g[ i ] | p[i + 1] & p[ i ] & c[ i ];
c[i + 3] = g[i + 2] | p[i + 2] &g[i + 1] | p[i +2] & p[i + 1] & g[ i ] | p[i + 2] & p[Ii + 1] & p[ i ] & c[ i ];
end
//now that carry-in values have been computed for
//all the sixty four L-0 adders (1-bit full adders)
//compute the sum value for all the sixty four L-0 adders (1-bit full adders)
for (i = 0; i < 64; i = i + 1)
sum[ i ] = a[ i ] ^ b[ i ] ^ c[ i ] ;
end
```

```
endmodule
```