```
/*
   CSA_tree_16to2 is a 16-to-2 reducer for 64-bit operands, using 3-to-2 sum-carry reducers.
   It is used to add the 16 augends produced from the Booth bit-pair recoding process for 
32-bit multiplication.
*/

// This module needed to be .sv to accept a 16-word wire vector.

module CSA_tree_16to2 (augends, reduced1, reduced2);

   // 16-word wire vector holding the bit-pair recoding augends.
   input [63:0] augends [15:0];

   // Final two operands.
   output [63:0] reduced1, reduced2;

   // Wire vectors for sum and carry results from each CSA stage.
   wire [63:0] stage1_sum [4:0];
   wire [63:0] stage1_carry [4:0];


   // SEE REPORT FOR 16-to-2 REDUCER TREE LAYOUT.

   genvar i;
   generate
   for (i = 0; i < 5; i=i+1) begin : instances
      reducer3to2 stage1 (augends[3*i+2], augends[3*i+1], augends[3*i], stage1_sum[i], 
stage1_carry[i]);
   end
   endgenerate

   wire [63:0] stage2_sum[2:0];
   wire [63:0] stage2_carry[2:0];

   reducer3to2 stage2_2 (augends[15], stage1_carry[4], stage1_sum[4], stage2_sum[2], 
stage2_carry[2]);
   reducer3to2 stage2_1 (stage1_carry[3], stage1_sum[3], stage1_carry[2], stage2_sum[1], 
stage2_carry[1]);
   reducer3to2 stage2_0 (stage1_sum[2], stage1_carry[1], stage1_sum[1], stage2_sum[0], 
stage2_carry[0]);

   wire [63:0] stage3_sum[1:0];
   wire [63:0] stage3_carry[1:0];

   reducer3to2 stage3_1 (stage2_carry[1], stage2_sum[1], stage2_carry[0], stage3_sum[1], 
stage3_carry[1]);
   reducer3to2 stage3_0 (stage2_sum[0], stage1_carry[0], stage1_sum[0], stage3_sum[0], 
stage3_carry[0]);

   wire [63:0] stage4_sum [1:0];
   wire [63:0] stage4_carry [1:0];

   reducer3to2 stage4_1 (stage2_carry[2], stage2_sum[2], stage3_carry[1], stage4_sum[1], 
stage4_carry[1]);
   reducer3to2 stage4_0 (stage3_sum[1], stage3_carry[0], stage3_sum[0], stage4_sum[0], 
stage4_carry[0]);

   wire [63:0] stage5_sum;
   wire [63:0] stage5_carry;

   reducer3to2 stage5_0 (stage4_sum[1], stage4_carry[0], stage4_sum[0], stage5_sum, 
stage5_carry);

   reducer3to2 stage6_0 (stage4_carry[1], stage5_carry, stage5_sum, reduced1, reduced2);

endmodule


// 3-to-2 Sum-Carry Reducer
module reducer3to2 (
   input [63:0] x, y, z,
   output [63:0] s,
   output [63:0] c
   );

```

```
    assign s = x ^ y ^ z; // Bitwise XOR
    assign c = (x & y | x & z | y & z) << 1; // Bitwise carry operation with shift. The 65th carry bit is not needed for a 64-bit result.

endmodule

```