

```

1  /*
2     CSA_tree_16to2 is a 16-to-2 reducer for 64-bit operands, using 3-to-2 sum-carry reducers.
3     It is used to add the 16 augends produced from the Booth bit-pair recoding process for
4     32-bit multiplication.
5  */
6  // This module needed to be .sv to accept a 16-word wire vector.
7
8  module CSA_tree_16to2 (augends, reduced1, reduced2);
9
10     // 16-word wire vector holding the bit-pair recoding augends.
11     input [63:0] augends [15:0];
12
13     // Final two operands.
14     output [63:0] reduced1, reduced2;
15
16     // Wire vectors for sum and carry results from each CSA stage.
17     wire [63:0] stage1_sum [4:0];
18     wire [63:0] stage1_carry [4:0];
19
20     // SEE REPORT FOR 16-to-2 REDUCER TREE LAYOUT.
21
22     genvar i;
23     generate
24     for (i = 0; i < 5; i=i+1) begin : instances
25         reducer3to2 stage1 (augends[3*i+2], augends[3*i+1], augends[3*i], stage1_sum[i],
26         stage1_carry[i]);
27     end
28     endgenerate
29
30     wire [63:0] stage2_sum[2:0];
31     wire [63:0] stage2_carry[2:0];
32
33     reducer3to2 stage2_2 (augends[15], stage1_carry[4], stage1_sum[4], stage2_sum[2],
34     stage2_carry[2]);
35     reducer3to2 stage2_1 (stage1_carry[3], stage1_sum[3], stage1_carry[2], stage2_sum[1],
36     stage2_carry[1]);
37     reducer3to2 stage2_0 (stage1_sum[2], stage1_carry[1], stage1_sum[1], stage2_sum[0],
38     stage2_carry[0]);
39
40     wire [63:0] stage3_sum[1:0];
41     wire [63:0] stage3_carry[1:0];
42
43     reducer3to2 stage3_1 (stage2_carry[1], stage2_sum[1], stage2_carry[0], stage3_sum[1],
44     stage3_carry[1]);
45     reducer3to2 stage3_0 (stage2_sum[0], stage1_carry[0], stage1_sum[0], stage3_sum[0],
46     stage3_carry[0]);
47
48     wire [63:0] stage4_sum [1:0];
49     wire [63:0] stage4_carry [1:0];
50
51     reducer3to2 stage4_1 (stage2_carry[2], stage2_sum[2], stage3_carry[1], stage4_sum[1],
52     stage4_carry[1]);
53     reducer3to2 stage4_0 (stage3_sum[1], stage3_carry[0], stage3_sum[0], stage4_sum[0],
54     stage4_carry[0]);
55
56     wire [63:0] stage5_sum;
57     wire [63:0] stage5_carry;
58
59     reducer3to2 stage5_0 (stage4_sum[1], stage4_carry[0], stage4_sum[0], stage5_sum,
60     stage5_carry);
61
62     reducer3to2 stage6_0 (stage4_carry[1], stage5_carry, stage5_sum, reduced1, reduced2);
63
64 endmodule
65
66 // 3-to-2 Sum-Carry Reducer
67 module reducer3to2 (
68     input [63:0] x, y, z,
69     output [63:0] s,
70     output [63:0] c
71 );

```

```
65
66     assign s = x ^ y ^ z; // Bitwise XOR
67     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.
68
69 endmodule
70
```