```
 1  library IEEE;
 2  use IEEE.std_logic_1164.all;
 3
 4  entity badd32 is
 5    port (a       : in  std_logic_vector(2 downto 0);  -- Booth multiplier
 6          b       : in  std_logic_vector(31 downto 0); -- multiplicand
 7          sum_in  : in  std_logic_vector(31 downto 0); -- sum input
 8          sum_out : out std_logic_vector(31 downto 0); -- sum output
 9          prod    : out std_logic_vector(1 downto 0)); -- 2 bits of product
10  end entity badd32;
11  library IEEE;
12  use IEEE.std_logic_1164.all;
13  entity add32 is             -- simple 32 bit ripple carry adder
14    port(a    : in  std_logic_vector(31 downto 0);
15         b    : in  std_logic_vector(31 downto 0);
16         cin  : in  std_logic;
17         sum  : out std_logic_vector(31 downto 0);
18         cout : out std_logic);
19  end entity add32;
20  library IEEE;
21  use IEEE.std_logic_1164.all;
22  entity bmul32 is   -- 32-bit by 32-bit two's complement multiplier
23    port (a : in  std_logic_vector(31 downto 0);   -- multiplier
24          b : in  std_logic_vector(31 downto 0);   -- multiplicand
25          p : out std_logic_vector(63 downto 0));  -- product
26  end entity bmul32;
27  library IEEE;
28  use IEEE.std_logic_1164.all;
29  entity fadd is                -- full adder stage, interface
30    port(a    : in  std_logic;
31         b    : in  std_logic;
32         cin  : in  std_logic;
33         s    : out std_logic;
34         cout : out std_logic);
35  end entity fadd;
36
37  architecture circuits of badd32 is
38    subtype word is std_logic_vector(31 downto 0);
39    signal bb        : word;
40    signal psum      : word;
41    signal b_bar     : word;
42    signal two_b     : word;
43    signal two_b_bar : word;
44    signal cout      : std_logic;
45    signal cin       : std_logic;
46    signal topbit    : std_logic;
47    signal topout    : std_logic;
48    signal nc1       : std_logic;
49  begin  -- circuits of badd32
50    b_bar <= not b;
51    two_b <= b(30 downto 0) & '0';
52    two_b_bar <= not two_b;
53    bb <= b when a="001" or a="010"           -- 5-input mux
54          else two_b when a="011"
55          else two_b_bar when a="100"         -- cin=1
56          else b_bar when a="101" or a="110"  -- cin=1
57          else x"00000000";
58    cin <= '1' when a="100" or a="101" or a="110"
59           else '0';
60    topbit <= b(31) when a="001" or a="010" or a="011"
61              else b_bar(31) when a="100" or a="101" or a="110"
62              else '0';
```

```
63
64      a1: entity WORK.add32 port map(sum_in, bb, cin, psum, cout);
65      a2: entity WORK.fadd port map(sum_in(31), topbit, cout, topout, nc1);
66
67      sum_out(29 downto 0) <= psum(31 downto 2);
68      sum_out(31) <= topout;
69      sum_out(30) <= topout;
70      prod <= psum(1 downto 0);
71    end architecture circuits;  -- of badd32
72
73    architecture circuits of bmul32 is
74      signal zer : std_logic_vector(31 downto 0) := x"00000000";     -- zeros
75      signal mul0: std_logic_vector(2 downto 0);
76      subtype word is std_logic_vector(31 downto 0);
77      type ary is array(0 to 15) of word;
78      signal s : ary;                         -- temp sums
79    begin  -- circuits of bmul32
80      mul0 <= a(1 downto 0) & '0';
81      a0:  entity WORK.badd32 port map(
82                   mul0,            b, zer,   s( 0), p( 1 downto  0));
83      a1:  entity WORK.badd32 port map(
84                   a(3  downto  1), b, s( 0), s( 1), p( 3 downto  2));
85      a2:  entity WORK.badd32 port map(
86                   a(5  downto  3), b, s( 1), s( 2), p( 5 downto  4));
87      a3:  entity WORK.badd32 port map(
88                   a(7  downto  5), b, s( 2), s( 3), p( 7 downto  6));
89      a4:  entity WORK.badd32 port map(
90                   a(9  downto  7), b, s( 3), s( 4), p( 9 downto  8));
91      a5:  entity WORK.badd32 port map(
92                   a(11 downto  9), b, s( 4), s( 5), p(11 downto 10));
93      a6:  entity WORK.badd32 port map(
94                   a(13 downto 11), b, s( 5), s( 6), p(13 downto 12));
95      a7:  entity WORK.badd32 port map(
96                   a(15 downto 13), b, s( 6), s( 7), p(15 downto 14));
97      a8:  entity WORK.badd32 port map(
98                   a(17 downto 15), b, s( 7), s( 8), p(17 downto 16));
99      a9:  entity WORK.badd32 port map(
100                  a(19 downto 17), b, s( 8), s( 9), p(19 downto 18));
101     a10: entity WORK.badd32 port map(
102                  a(21 downto 19), b, s( 9), s(10), p(21 downto 20));
103     a11: entity WORK.badd32 port map(
104                  a(23 downto 21), b, s(10), s(11), p(23 downto 22));
105     a12: entity WORK.badd32 port map(
106                  a(25 downto 23), b, s(11), s(12), p(25 downto 24));
107     a13: entity WORK.badd32 port map(
108                  a(27 downto 25), b, s(12), s(13), p(27 downto 26));
109     a14: entity WORK.badd32 port map(
110                  a(29 downto 27), b, s(13), s(14), p(29 downto 28));
111     a15: entity WORK.badd32 port map(
112                  a(31 downto 29), b, s(14), p(63 downto 32) , p(31 downto 30));
113   end architecture circuits;  -- of bmul32
114
115
116
117
118   architecture circuits of fadd is
119   begin
120     s <= a xor b xor cin after 1 ps;
121     cout <= (a and b) or (a and cin) or (b and cin) after 1 ps;
122   end architecture circuits; -- of fadd
123
124
```

```
125
126
127   architecture circuits of add32 is
128     signal c : std_logic_vector(0 to 30);
129   begin
130     a0: entity WORK.fadd port map(a(0), b(0), cin, sum(0), c(0));
131     stage: for I in 1 to 30 generate
132                 as: entity WORK.fadd port map(a(I), b(I), c(I-1) , sum(I), c(I));
133            end generate stage;
134     a31: entity WORK.fadd port map(a(31), b(31), c(30) , sum(31), cout);
135   end architecture circuits;  -- of add32
136
```