```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.textio.all;


entity TB_Adder_CSA is
end TB_Adder_CSA;

architecture behavior of TB_Adder_CSA is
  constant N : integer := 64;

  -- DUT ports
  signal TBA, TBB : std_logic_vector(N-1 downto 0) := (others => '0');
  signal TBCin    : std_logic := '0';
  signal TBS      : std_logic_vector(N-1 downto 0);
  signal TBCout   : std_logic;
  signal TBOvfl   : std_logic;

  -- Test-vector file
  constant TestVectorFile : string := "TestVectors/Adder00.tvs";
  constant PreStimTime    : time   := 1 ns;
  constant PostStimTime   : time   := 100 ns;  -- adjust after experimenting

begin
  ---------------------------------------------------------------------
  -- Device Under Test
  ---------------------------------------------------------------------
  dut: entity work.EN_Adder(CSA)
    generic map (N => N)
    port map (
      A => TBA,
      B => TBB,
      Cin => TBCin,
      S => TBS,
      Cout => TBCout,
      Ovfl => TBOvfl
    );

  ---------------------------------------------------------------------
  -- Stimulus Process with per-vector line printing
  ---------------------------------------------------------------------
  stimulus : process
    file      tvf : text;
    variable  L, L2 : line;
    constant  MAXLEN : natural := 2048;
    variable  s : string(1 to MAXLEN);
    variable  vA, vB, vS : std_logic_vector(N-1 downto 0);
    variable  vCin, vCout, vOvfl : std_logic;
    variable  skip_line  : boolean;
    variable  idx        : natural := 0;   -- measurement index
    variable  pass       : boolean;
    variable  OUTL       : line;          -- for printing one summary line
  begin
    file_open(tvf, TestVectorFile, read_mode);
    report "Using test vectors from file: " & TestVectorFile;

    while not endfile(tvf) loop
      readline(tvf, L);

      -- Skip blank or comment lines (comments start with "--")
      if L'length = 0 then
        next;
      end if;

      skip_line := false;

      if L'length > MAXLEN then
        report "Input line exceeds MAXLEN=" & integer'image(MAXLEN) severity failure;
      end if;

      s := (others => ' ');
      s(1 to L'length) := L.all;  -- length-safe copy
```

```
74
75          -- Check if the first two non-space characters are "--"
76          for i in s'range loop
77            if s(i) > ' ' then
78              if i < s'high and s(i) = '-' and s(i + 1) = '-' then
79                skip_line := true;
80              end if;
81              exit;
82            end if;
83          end loop;
84          if skip_line then
85            next;
86          end if;
87
88          -- Rebuild the line to parse values
89          L2 := null;
90          write(L2, s(1 to L'length));
91
92          -- Parse: A B Cin S Cout Ovfl
93          HREAD(L2, vA);
94          HREAD(L2, vB);
95          read (L2, vCin);
96          HREAD(L2, vS);
97          read (L2, vCout);
98          read (L2, vOvfl);
99
100         -- 1) Drive 'X' for PreStimTime (per spec)
101         TBA   <= (others => 'X');
102         TBB   <= (others => 'X');
103         TBCin <= 'X';
104         wait for PreStimTime;
105
106         -- 2) Apply inputs
107         TBA   <= vA;
108         TBB   <= vB;
109         TBCin <= vCin;
110
111         -- 3) Wait for outputs to settle (experiment to choose PostStimTime)
112         wait for PostStimTime;
113
114         -- 4) Compute pass/fail and (optionally) assert
115         pass := (TBS = vS) and (TBCout = vCout) and (TBOvfl = vOvfl);
116
117         assert pass
118           report "Mismatch: i=" & integer'image(idx) &
119                  " A=" & to_hstring(TBA) &
120                  " B=" & to_hstring(TBB) &
121                  " Cin=" & std_logic'image(TBCin) &
122                  "  got S=" & to_hstring(TBS) & " Cout=" & std_logic'image(TBCout) & "
                     Ovfl=" & std_logic'image(TBOvfl) &
123                  "  exp S=" & to_hstring(vS)  & " Cout=" & std_logic'image(vCout)  & "
                     Ovfl=" & std_logic'image(vOvfl)
124           severity error;
125
126         -- 5) Print one concise summary line (goes to ModelSim transcript)
127         OUTL := null;
128         write(OUTL, idx);
129         write(OUTL, string'(" A="));              write(OUTL, to_hstring(TBA));
130         write(OUTL, string'(" B="));              write(OUTL, to_hstring(TBB));
131         write(OUTL, string'(" Cin="));            write(OUTL, TBCin);
132         write(OUTL, string'("  |  S="));         write(OUTL, to_hstring(TBS));
133         write(OUTL, string'(" Cout="));           write(OUTL, TBCout);
134         write(OUTL, string'(" Ovfl="));           write(OUTL, TBOvfl);
135         write(OUTL, string'("  status="));
136         if pass then write(OUTL, string'("PASS"));
137                 else write(OUTL, string'("FAIL"));
138         end if;
139         writeline(output, OUTL);
140
141         idx := idx + 1;
142       end loop;
143
144       report "Simulation completed: reached end of " & TestVectorFile;
```

```
145       file_close(tvf);
146       wait;
147     end process;
148   end architecture;
149
150
```