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


entity EN_Adder is
    generic (N: natural := 64);
    port (
        A, B  : in std_logic_vector (N-1 downto 0);
        S : out std_logic_vector (N-1 downto 0);
        Cin : in std_logic;
        Cout, Ovfl : out std_logic
    );
end EN_Adder;

architecture ripple of EN_Adder is
    signal C : std_logic_vector (N-1 downto 0);
    signal P,G : std_logic_vector (N-1 downto 0);

    begin
    P(0) <= A(0) xor B(0);
    G(0) <= A(0) and B(0);
    S(0) <= P(0) xor Cin;
    C(0) <= G(0) or (P(0) and Cin);

    rippleNetwork: for i in 1 to N-1 generate
        P(i) <= A(i) xor B(i);
        G(i)<= A(i) and B(i);
        S(i) <= P(i) xor C(i-1);
        C(i) <= G(i) or (P(i) and C(i-1));
    end generate;
    Cout <= C(N-1);
    Ovfl <= (not (A(N-1) xor B(N-1))) and (A(N-1) xor S(N-1));


end ripple;

architecture FastRipple of EN_Adder is

signal temp : unsigned(N downto 0);
    begin
    -- Extend A and B to N+1 bits with a leading '0'
    -- Extend Cin to N+1 bits with N zeros in the upper bits

        temp <= unsigned('0' & A) + unsigned('0' & B) + unsigned(to_unsigned(0,N) &
        Cin);

        S    <= std_logic_vector(temp(N-1 downto 0));
    Cout <= temp(N);
     Ovfl <= (temp(N) xor temp(N-1));
end FastRipple;

architecture CSA of EN_Adder is
    constant N_half : integer := N / 2;
    signal Cout0, Cout1, Chalf : std_logic := '0';
    signal sum0, sum1 : std_logic_vector ((N_half-1) downto 0) := (others => '0');


    begin
        recur : if N > 2 generate
            begin
    CSAlow : entity work.EN_Adder(CSA)
      generic map (N => N_half)
      port map (
        A => A(N_half-1 downto 0),
        B => B(N_half-1 downto 0),
        S => S(N_half-1 downto 0),
        Cin => Cin,
        Cout => Chalf,
        Ovfl => open
      );
    CSAzero : entity work.EN_Adder(CSA)
```

```
73          generic map (N => N_half)
74          port map (
75            A    => A(N-1 downto N_half),
76            B    => B(N-1 downto N_half),
77            S    => sum0,
78            Cin  => '0',
79            Cout => Cout0,
80            Ovfl => open
81          );
82        CSAone : entity work.EN_Adder(CSA)
83          generic map (N => N_half)
84          port map (
85            A    => A(N-1 downto N_half),
86            B    => B(N-1 downto N_half),
87            S    => sum1,
88            Cin  => '1',
89            Cout => Cout1,
90            Ovfl => open
91          );
92            end generate recur;
93
94            leaf: if N = 2 generate
95                signal g, p : std_logic;
96                begin
97                g <= A(0) and B(0);
98                p <= A(0) xor B(0);
99                S(0) <= p xor Cin;
100               Chalf <= g or (Cin and p);
101
102               sum0(0) <= A(1) xor B(1);
103               sum1(0) <= not (A(1) xor B(1));
104               Cout0 <= A(1) and B(1);
105               Cout1 <= A(1) or B(1);
106           end generate leaf;
107
108
109           S((N-1)downto N_half) <= sum1 when Chalf = '1' else sum0;
110           Cout <= Cout1 when Chalf = '1' else Cout0;
111
112           Ovfl <= (not (A(N-1) xor B(N-1))) and (A(N-1) xor S(N-1));
113
114
115
116   end CSA;
```