```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity min_bit_value is
    generic (
        N : positive := 8;  -- Set the number of bits for the input
values
        W : positive := 4   -- Set the number of bits for the
multiplier
    );
    port (
        clk : in std_logic;
        reset : in std_logic;
        x_in : in std_logic_vector(N-1 downto 0);
        wp : in std_logic_vector(W-1 downto 0);
        y_out : out std_logic_vector(N-1 downto 0)
    );
end entity min_bit_value;

architecture rtl of min_bit_value is
    signal product : signed(N+W-1 downto 0);
    signal sum : signed(N+W downto 0) := (others => '0');
    signal min_value : signed(N-1 downto 0) := (others => '1');
    signal i : natural range 0 to N-1 := 0;
begin
    process (clk, reset)
    begin
        if reset = '1' then
            sum <= (others => '0');
            min_value <= (others => '1');
            i <= 0;
        elsif rising_edge(clk) then
            -- Compute the product
            product <= signed(x_in) * signed(wp);

            -- Accumulate the sum
```

```
            sum <= sum + product;

            -- Update the minimum value
            if product(N-1 downto 0) < min_value then
                min_value <= product(N-1 downto 0);
            end if;

            -- Increment the index
            i <= i + 1;
            if i = N-1 then
                i <= 0;
            end if;
        end if;
    end process;

    -- Output the minimum value
    y_out <= std_logic_vector(min_value);
end architecture rtl;
```