1 absBlock.vhd

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

--The functionality of this block is the following:

--In input the number can be positive or negative.
--The input is positive if the leftest bit is 0, otherwise is negative.

--When the input is a negative number, it will be complemented and then 1 will
 be added.
--Using this procedure it's possibile to compute the negative number module.

--The positive case is trivial: it is necessary consider only the N-1 rightest
 bits.
--The above case is implemented in the absolutePositiveNumber function.

entity absBlock is
             generic(N: Integer:=9);
    port (
    input   : in std_logic_vector (0 to N);
    output  : out std_logic_vector (0 to N-1));
END absBlock;

architecture behav of absBlock is

function sumPlusOne(a: in std_logic_vector)
    return std_logic_vector is
    variable sum: std_logic_vector(0 to N-1);
    variable oneNBit: std_logic_vector(0 to N-1);
    variable carry: std_logic;

    begin
    carry:='0';
    for  i in 0 to N-1 loop
        oneNBit(i):='0';
        sum(i):='0';
    end loop;

    oneNBit(N-1):='1';
    for  i in N-1 downto 0 loop
        sum(i):=    a(i) xor oneNBit(i) xor carry;
        carry:= (a(i) and oneNBit(i)) or (carry and a(i)) or (carry and oneNBi
t(i));
    end loop;

    return sum;
    end sumPlusOne;


function absolutePositiveNumber(a: in std_logic_vector)
    return std_logic_vector is
    variable tmp: std_logic_vector(0 to N-1);
    begin
    for  i in 1 to N loop
        tmp(i-1):=a(i);
    end loop;
    return tmp;
    end   absolutePositiveNumber;

function complement2(a: in std_logic_vector)
```

```
    return std_logic_vector is
    variable tmp: std_logic_vector(0 to N-1);
    begin
    for  i in 1 to N loop
        tmp(i-1):=not a(i);
    end loop;
    tmp:=sumPlusOne(tmp);
    return tmp;

    end complement2;


signal tmp:std_logic_vector (0 to N-1);

begin
    tmp<=absolutePositiveNumber(input) when(input(0)='0')else complement2(inpu
t);
    output<=tmp;

    end behav;
```