```
----------------------------------------------------------------------------------
--------------
-- Company:
-- Engineer:
--
-- Create Date: 02/01/2024 02:05:09 PM
-- Design Name:
-- Module Name: compunit - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
--------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.std_logic_unsigned.ALL;
USE IEEE.numeric_std.ALL;

--use IEEE.std_logic_unsigned.all;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity compunit is
generic( dw: integer:=4);
 Port (a,b:in std_logic_vector(dw-1 downto 0);
  s: in std_logic_vector(dw downto 0);
  o:out std_logic_vector(dw-1 downto 0));
end compunit;

architecture Behavioral of compunit is

begin

process (A,B,s)
begin

case s is

when"00000"=>o<=A and B;
when"00001"=>o<=A or B;
when"00010"=>o<=A nand B;
when"00011"=>o<=A nor B;
when"00100"=>o<=A xor B;
when"00101"=>o<=A xnor B;
when"00110"=>o<=A + B;
when"00111"=>o<=A - B;
when"01000"=>o<=
std_logic_vector(to_unsigned(to_integer(unsigned(a)) *
to_integer(unsigned(b)),4));
when"01001"=> if(A > B) then  o <= (others => '1'); else o <=
(others => '0');end if;
when"01010"=> if(A < B) then o <= (others => '1'); else o <=
(others => '0');end if;
when"01011"=> if(A = B) then o <= (others => '1'); else o <=
(others => '0');end if;
when"01100"=> if(A >= B) then o <= (others => '1'); else o <=
(others => '0');end if;
when"01101"=> if(A <= B) then o <= (others => '1'); else o <=
(others => '0');end if;
```

```
when"01110"=> if(A /= B) then o <= (others => '1'); else o <=
(others => '0');end if;
when"01111"=> o <=(others => '0');
when "10000" => o <= to_stdlogicvector(to_bitvector (a) sla
TO_INTEGER(unsigned( b)));
when "10001" => o <= to_stdlogicvector(to_bitvector (a) sra
TO_INTEGER(unsigned( b)));
when "10010" => o <= to_stdlogicvector(to_bitvector (a) srl
TO_INTEGER(unsigned( b)));
when "10011" => o <= to_stdlogicvector(to_bitvector (a) sll
TO_INTEGER(unsigned( b)));
when "10100" => o <= to_stdlogicvector(to_bitvector (a) ror
TO_INTEGER(unsigned( b)));
when "10101" => o <= to_stdlogicvector(to_bitvector (a) rol
TO_INTEGER(unsigned( b)));
when others => o <= (others => '0');
end case;

end process;
end Behavioral;
```