```vhdl
----------------------------------------------------------------------------------
-- Engineer: Olasubomi Borishade
--
-- Create Date: 02/03/2026 04:42:08 PM
-- Design Name: Arithmetic Logic Block
-- Module Name: Arithmetic_Logic_Block - Behavioral
-- Project Name: EENG 5560 Homework 2

-- Description: Logic block capable of adding,
-- subtracting and multiplying to inputs A and B

-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.OPERATIONS_ARRAY_CUSTOM_PACK.ALL;

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

entity Arithmetic_Logic_Block is
    Generic (d_w: integer:= 4);

    Port ( A : in STD_LOGIC_VECTOR (d_w - 1 downto 0); -- Declaring A as a input
vector
           B : in STD_LOGIC_VECTOR (d_w - 1 downto 0); -- Declaring B as a input
vector
           OpSelA : in ALB_Operation; -- Declaring the operation selection signal as
an input vector
           Y : out STD_LOGIC_VECTOR (d_w - 1 downto 0) -- Declaring Y as a vector
           );

end Arithmetic_Logic_Block;

architecture Behavioral of Arithmetic_Logic_Block is

begin
ALB_proc: process (A, B, OpSelA)
```

```
    variable Full_Addition : STD_LOGIC_VECTOR (d_w - 1 downto 0); -- Defining 
full-bit width intermediate addition signal
    variable Full_Subtraction : STD_LOGIC_VECTOR (d_w - 1 downto 0); -- Defining 
full-bit width intermediate subtraction signal
    variable Full_Multiplication : STD_LOGIC_VECTOR (2 * d_w - 1 downto 0); -- 
Defining full-bit width intermediate multiplication signal

begin
    case OpSelA is
    when aADD =>
        Full_Addition := STD_LOGIC_VECTOR (Unsigned(A) + Unsigned(B)); -- Computes 
the full-bit width arithmetic addition of inputs A and B if the operation selection 
signal is 00
        Y <= Full_Addition (d_w - 1 downto 0); -- Truncates the sum to our specified 
bit width

    when aSUB =>
        Full_Subtraction := STD_LOGIC_VECTOR (Unsigned(A) - Unsigned(B)); -- 
Computes the full-bit width arithmetic sutraction of inputs A and B if the operation 
selection signal is 01
        Y <= Full_Subtraction (d_w - 1 downto 0); -- Truncates the difference to our 
specified bit width

    when aMULT =>
        Full_Multiplication := STD_LOGIC_VECTOR (Unsigned(A) * Unsigned (B)); -- 
Computes the full-bit width arithmetic multiplication of inputs A and B if the 
operation selection signal is 100
        Y <= Full_Multiplication (d_w - 1 downto 0); -- Truncates the difference to 
our specified bit width

    when NoOp =>
        Y <= "0000";

    when others =>
        Y <= (others => 'Z');

    end case;
end process;

end Behavioral;
```