```
----------------------------------------------------------------------------------
-- Engineer: Olasubomi Borishade
--
-- Create Date: 01/30/2026 08:55:31 PM
-- Design Name: Computation Unit Array
-- Module Name: Computation_Unit_Array - Behavioral
-- Project Name: EENG 5560 Homework 1
-- Description: A fully-connected 2x8 (2 rows, 8 columns) reconfigurable
architecture
-- where each CU in a row can send information to any of the CUs in the row below it
-- 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_CUSTOMPACK.ALL;

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

entity Computation_Unit_Array is
    Generic (d_w: integer := 4; -- parameterizing the data width
            rows: natural := 2; -- parameterizing the rows of the computation unit
array
            cols: natural := 8 -- parameterizing the columns of the computation unit
array
            );

    Port ( A, B : in VectorArray_1d (0 to cols - 1)(d_w - 1 downto 0); -- input
vector array for inputs A and B of each computation unit
            Opsel : in OperationArray_2d (0 to rows - 1)(0 to cols - 1); -- input
vector array for the operation select signals of each computation unit
            DataFlow1: in VectorArray_1d (0 to cols - 1)(2 downto 0); -- input
selector array for each A input of the second row of computation units
            DataFlow2: in VectorArray_1d (0 to cols - 1)(2 downto 0); -- input
selector array for each B input of the second row of computation units
            Y : out VectorArray_1d (0 to cols - 1)(d_w - 1 downto 0) -- output
vector array for the bottom row of computation units
            );

end Computation_Unit_Array;
```

```
architecture Structural of Computation_Unit_Array is
Signal As, Bs: VectorArray_1d (0 to cols - 1)(d_w - 1 downto 0); -- intermediate
input signal for the bottom row of computation units
Signal Ys: VectorArray_2d (0 to rows - 1)(0 to cols - 1)(d_w - 1 downto 0); --
output signal array of all computation units

begin
gen_rows: for r in 0 to rows - 1 generate
gen_cols: for c in 0 to cols - 1 generate

    dataflow1_proc: process(DataFlow1(c)) -- process is sensitive to each vector in
each column of the DataFlow1 array
    begin
        case DataFlow1(c) is -- if any vector in a specific column of array
DataFlow1 reads:
        when "000" => -- a value of "000"
            As(c) <= Ys(0)(0); -- assign the same column of signal As the output
value of CU (0,0)

        when "001" => -- a value of "001"
            As(c) <= Ys(0)(1); -- assign the same column of signal As the output
value of CU (0,1)

        when "010" => -- a value of "010"
            As(c) <= Ys(0)(2); -- assign the same column of signal As the output
value of CU (0,2)

        when "011" => -- a value of "011"
            As(c) <= Ys(0)(3); -- assign the same column of signal As the output
value of CU (0,3)

        when "100" => -- a value of "100"
            As(c) <= Ys(0)(4); -- assign the same column of signal As the output
value of CU (0,4)

        when "101" => -- a value of "101"
            As(c) <= Ys(0)(5); -- assign the same column of signal As the output
value of CU (0,5)

        when "110" => -- a value of "110"
            As(c) <= Ys(0)(6); -- assign the same column of signal As the output
value of CU (0,6)

        when "111" => -- a value of "111"
            As(c) <= Ys(0)(7); -- assign the same column of signal As the output
```

```
value of CU (0,7)

        when others =>
            As(c) <= (others => 'Z');
        end case;
    end process;

    dataflow2_proc: process(DataFlow2(c)) -- process is sensitive to each vector in
each column of the DataFlow2 array
    begin
        case DataFlow2(c) is -- if any vector in a specific column of array
DataFlow2 reads:
        when "000" => -- a value of "000"
            Bs(c) <= Ys(0)(0); -- assign the same column of signal Bs the output
value of CU (0,0)

        when "001" => -- a value of "001"
            Bs(c) <= Ys(0)(1); -- assign the same column of signal Bs the output
value of CU (0,1)

        when "010" => -- a value of "010"
            Bs(c) <= Ys(0)(2); -- assign the same column of signal Bs the output
value of CU (0,2)

        when "011" => -- a value of "011"
            Bs(c) <= Ys(0)(3); -- assign the same column of signal Bs the output
value of CU (0,3)

        when "100" => -- a value of "100"
            Bs(c) <= Ys(0)(4); -- assign the same column of signal Bs the output
value of CU (0,4)

        when "101" => -- a value of "101"
            Bs(c) <= Ys(0)(5); -- assign the same column of signal Bs the output
value of CU (0,5)

        when "110" => -- a value of "110"
            Bs(c) <= Ys(0)(6); -- assign the same column of signal Bs the output
value of CU (0,6)

        when "111" => -- a value of "111"
            Bs(c) <= Ys(0)(7); -- assign the same column of signal Bs the output
value of CU (0,7)

        when others =>
            Bs(c) <= (others => 'Z');
```

```
        end case;
    end process;

first: if r = 0 generate
    row0: entity work.Computation_Unit(Behavioral)
    Generic map (d_w => d_w)
    Port map (A => A(c), B => B(c), Opsel => Opsel(r)(c), Y => Ys(r)(c));
end generate first;

rest: if r > 0 generate
    rest_rows: entity work.Computation_Unit(Behavioral)
    Generic map(d_w => d_w)
    Port map (A => As(c), B => Bs(c), Opsel => Opsel(r)(c), Y => Ys(r)(c));
end generate rest;

end generate gen_cols;
end generate gen_rows;
Y <= Ys(rows - 1)(0 to cols - 1);

end Structural;
```