```
--------------------------------------------------------------------------------
-- EXAM QUESTION 2 --
--------------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY control_flow IS
    PORT (
        clock : IN std_logic;
        reset : IN std_logic;
        x0, x1, x2, x3, x4, x5, x6, x7, x8, x9 : IN std_logic_vector(7 DOWNTO
0);
        h0, h1, h2, h3, h4, h5, h6, h7, h8, h9 : IN std_logic_vector(7 DOWNTO
0);
        y : OUT std_logic_vector(7 DOWNTO 0)
    );
END control_flow;

ARCHITECTURE arch OF control_flow IS

    SIGNAL temp_output : std_logic_vector(7 DOWNTO 0) := (OTHERS => '0');
    SIGNAL term_0 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_1 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_2 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_3 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_4 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_5 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_6 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_7 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_8 : std_logic_vector(7 DOWNTO 0);
    SIGNAL term_9 : std_logic_vector(7 DOWNTO 0);

    SIGNAL counter : std_logic;

    FUNCTION MULTIPLY(x : std_logic_vector; h : std_logic_vector) RETURN std_l
ogic_vector IS
        VARIABLE result : std_logic_vector(15 DOWNTO 0);
        VARIABLE x_in : std_logic_vector(7 DOWNTO 0);
        VARIABLE h_in : std_logic_vector(7 DOWNTO 0);
    BEGIN

        x_in := std_logic_vector(unsigned(x));
        h_in := std_logic_vector(unsigned(h));
        result := (x_in * h_in);
```

```
        RETURN std_logic_vector(result);
    END FUNCTION MULTIPLY;

    FUNCTION ADD(term_1 : std_logic_vector; term_2 : std_logic_vector) RETURN
std_logic_vector IS
        VARIABLE result : std_logic_vector(7 DOWNTO 0);
        VARIABLE t1 : std_logic_vector(7 DOWNTO 0);
        VARIABLE t2 : std_logic_vector(7 DOWNTO 0);
    BEGIN

        t1 := std_logic_vector(unsigned(term_1));
        t2 := std_logic_vector(unsigned(term_2));

        result := t1 + t2;

        RETURN std_logic_vector(result);
    END FUNCTION ADD;

BEGIN

    PROCESS (clock, reset)
    BEGIN
        IF (reset = '1') THEN

            counter <= '0'; --multiplication hasn't been done yet
        ELSIF rising_edge(clock) THEN
            --rising edge half cycle is for multiplications

            term_0 <= MULTIPLY(x0, h0);
            term_0 <= MULTIPLY(x1, h1);
            term_0 <= MULTIPLY(x2, h2);
            term_0 <= MULTIPLY(x3, h3);
            term_0 <= MULTIPLY(x4, h4);
            term_0 <= MULTIPLY(x5, h5);
            term_0 <= MULTIPLY(x6, h6);
            term_0 <= MULTIPLY(x7, h7);
            term_0 <= MULTIPLY(x8, h8);
            term_0 <= MULTIPLY(x9, h9);
            counter <= '1'; --marks that multiplication is done
        END IF;
    END PROCESS;

    PROCESS (clock, reset)
    BEGIN
        --IF (reset = '1') THEN
            --temp_output <= (OTHERS => '0');
        IF falling_edge(clock) AND (counter = '1') THEN
            --falling edge half cycle is for additions
```

```
            temp_output <= ADD(term_0, term_1);
            temp_output <= ADD(temp_output, term_2);
            temp_output <= ADD(temp_output, term_3);
            temp_output <= ADD(temp_output, term_4);
            temp_output <= ADD(temp_output, term_5);
            temp_output <= ADD(temp_output, term_6);
            temp_output <= ADD(temp_output, term_7);
            temp_output <= ADD(temp_output, term_8);
            temp_output <= ADD(temp_output, term_9);
        END IF;
    END PROCESS;

    y <= temp_output;

END arch;
```