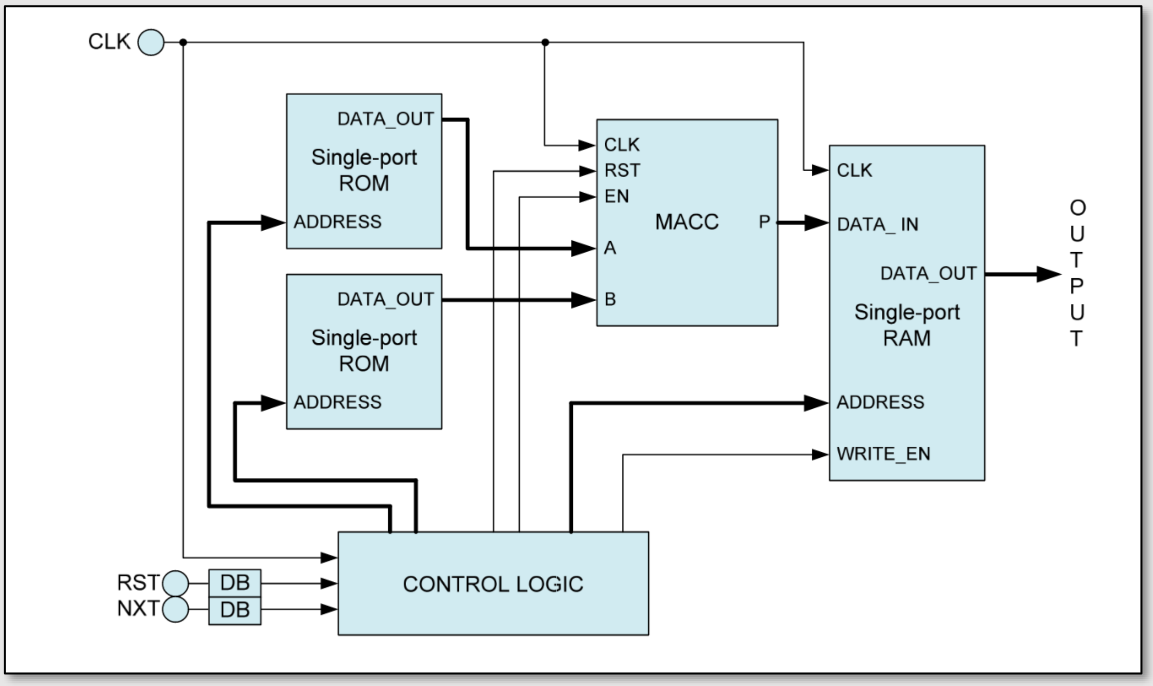
FINAL PROJECT

Matrix Multiplication Circuit Design



Digital Design with HDL
Y3859110 - Y3857872

```
__   ___  _ ___  _      ___                   _   
\ \ / / || |   \| |    | _ \___ _ __  ___ _ _| |_ 
 \ V /| __ | |) | |__  |   / -_) '_ \/ _ \ '_|  _|
  \_/ |_||_|___/|____| |_|_\___| .__/\___/_|  \__|
                               |_|                
```

Table of Contents

Technical Report

Q1. Datapath

The ‘Datapath’ acts on the instructions set by the ‘Control Logic’ and sends both data and addresses to the relevant memory components. As the system is parameterizable, the data width of the RAM and thus the coefficients of matrix ‘C’ must be adaptable to the operations performed by the MACC. In this case, the maximum width (in binary) of the coefficients is determined by the size of the two input matrices and by the data size of the input coefficients.

First, to find the maximum data width of the RAM we must first look at the largest magnitude number expressed by an N-bit number. For an unsigned number the largest magnitude for an N-bit number is $2^N - 1$, however for a 2’s complement signed number the largest magnitude is 2^{N-1} as the MSB is expressed negatively. As the MACC first multiplies these two magnitudes, the resulting product is $2^{(N-1)^2}$, which due to the laws of powers can be expressed as $2^{2(N-1)}$. As the MACC sums together ‘M’ number of products for one coefficient result, the total magnitude is then multiplied by the natural number ‘M’.

Expressing this total magnitude as a binary number, the magnitude must be expressed to a base-2 number, where the number of bits required can be calculated by finding the upper integer boundary of log_2 , or by using the ‘size’ function from the ‘DigEng’ library package provided. Finally, again due to 2’s complement, an extra bit is added to the size to offset the negative MSB when denoting the upper positive boundary, and thus the equation for the size of a coefficient is:

$$C = size(M \times \left(2^{2(data\_size-1)}\right) + 1)$$

[C = Data size of final calculated coefficient]

[M = Common size parameter of input matrices]

[data_size = Data size of input coefficient]

Q2. Memory

The ROMs 'A' and 'B' both have a memory width of the constant 'data_size', whereas the RAM has a memory width of $size(M \times (2^{2(data\_size-1)}) + 1)$, as derived in Q1.

The memory depths are specific to the size of each matrix. Respectively the data depths are:

$$\begin{gathered} Depth_{ROMA} = H \times M \\ Depth_{ROMB} = N \times M \\ Depth_{RAM} = H \times N \end{gathered}$$

The binary size of the address bus for each memory depth can be calculated by taking $log_2(depth)$ of each component in order for the counters to increment through the address values. For the input matrices used in the ROMs, see the 'Testbench Process' heading.

Q3. Control Logic

The limit of the counters is set by the generics values M, N and H respectively to their corresponding counter. The counters consist of unsigned values incrementing by 1, from 0, with their relative 'count enable' signal - each counter vector length is the upper integer limit of $'log_2(M, N, H) - 1\ downto\ 0'$.

The addresses of ROM A, ROM B and the RAM are set by a function of these counter values, where the coefficient address for the matrix 'C' are set by H and N, whereas the address for ROM A and B are set be N and M, H and M respectively. For example when '$Count_M$' is incremented, the address of ROM A increments by 1, whereas with ROM B, '$Count_M$' increments by integer 'N'. This ensures that the row of one matrix is always paired with the column of the other as '$Count_M$' increases, thus allowing the calculating of the each coefficient. The counter incriminations (count enable signals) are tied to the state-machine output assignments.

The combinational logic used for address generation are:

$$\begin{gathered} Addr_{ROM\ A} = (Count_H \times M) + Count_M \\ Addr_{ROM\ B} = (Count_M \times N) + Count_N \\ Addr_{RAM} = (Count_H \times N) + Count_N \end{gathered}$$

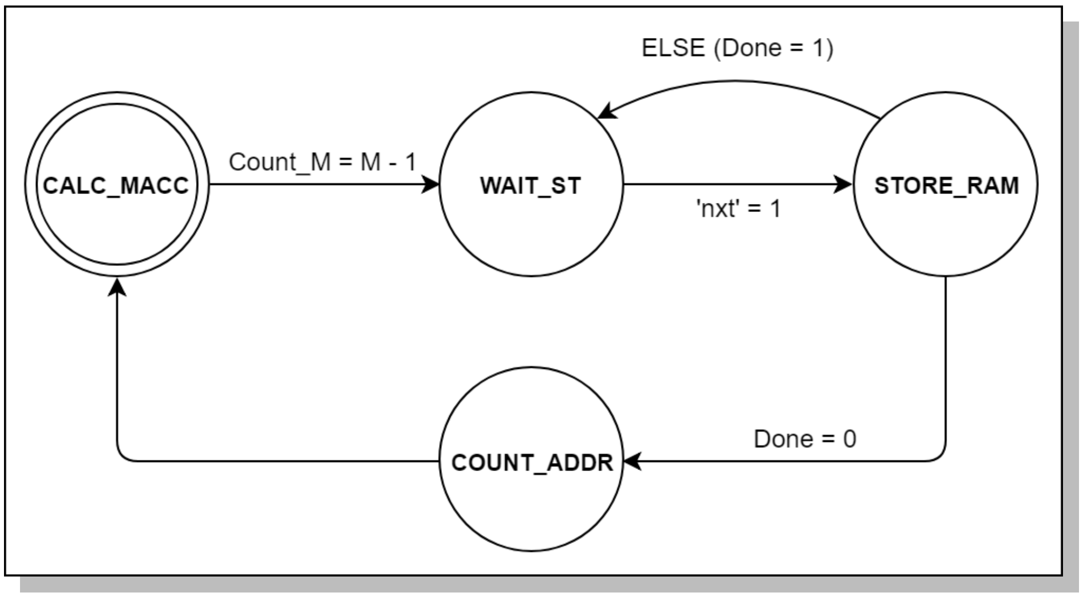
Beyond the global reset, each counter resets to '0' when the debounced user reset is pressed. The behaviour of each counter also wraps back round when the counter increments past their limit. Due to the operations of the 'IEEE.NUMERIC_STD' package, when multiplying a vector by an integer the resize function is required in order to rescale the vector length of the combinational logic to match the length of the memory address bus. The length of each address bus is set by the memory depth of the component, or $log_2(depth)$.

Q4. Finite-State Machine

The state machine is designed in a specific order to ensure that the output is always defined. With this design, four states are used in a cyclic behaviour. The next coefficient value is always calculated before each ‘nxt’ button press and as a result only requires ‘nxt’ to be pressed to store, output and display the value. As a design choice, it is chosen that when the ‘rst’ button is pressed, the counters (and therefore the address buses) are initialised to ‘0’ and the system returns to the ‘CALC_MACC’ state, where the first coefficient is calculated again before becoming idle again. The system then moves to ‘WAIT_ST’ and is idle until the next user input.

Furthermore, when the ‘done’ signal is high, the state transition is looped between ‘WAIT_ST’ and ‘STORE_RAM’ to “freeze” the system and prevent any more calculations. In this condition, each ‘nxt’ button press overwrites the final calculated value to the same address and the state machine returns to ‘WAIT_ST’, causing a loop until the next user reset.

FSM Transition Diagram



State Descriptions for FSM

State	Description
CALC_MACC	In this state, 'M_en' and 'macc_en' are enabled. The two ROM units are loaded into the MACC and the multiply-accumulate action repeats until the coefficient is calculated, or when 'Count_M' reaches its maximum value. The state then moves to 'WAIT_ST'.
WAIT_ST	In this state, all memories (including registers) are disabled and the system is idle. When the user presses 'nxt', the state moves to STORE_RAM.
STORE_RAM	In this state, 'ram_en' is high, allowing the coefficient to be stored to the RAM. The register within the MACC resets ('rst_reg' is high) to prepare for the next coefficient calculation. 'Count_N' increments, unless all numbers have been computed (ELSE 'done = 1'), in which case the FSM returns to 'WAIT_ST'. If 'done = 0', the state moves on to COUNT_ADDR.
COUNT_ADDR	In this state, 'Count_H' is checked by the condition 'Count_N' = (N - 1). If the condition is met, 'H_en' is enabled, incrementing by one, else it continues without enabling. It then proceeds back to CALC_MACC.

Output Assignments from 'Control Logic' to 'Datapath'

				Counters		
State	**macc_en**	**ram_en**	**rst_reg**	**M_en**	**N_en**	**H_en**
CALC_MACC	1	0	rst	1	0	0
WAIT_ST	0	0	rst	0	0	0
STORE_RAM	0	1	1	0	0	0
COUNT_ADDR	0	0	rst	0	1	'1' when: 'Count_N = (N - 1)'

ROM A

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     work.DigEng.ALL;

-- Asynchronous Read single-port ROM
    -- Memory Depth => (HxM)
    -- Memory Width => data_size
    -- Data_out signal is respectively Data_A.

ENTITY ROM_A IS
    GENERIC (
        data_size : NATURAL;
        H         : NATURAL;
        M         : NATURAL
    );

    PORT (
        -- Address Input
        Addr     : IN    UNSIGNED (log2(H * M) - 1 DOWNTO 0);
        -- Data Output
        Data_out : OUT   SIGNED (data_size - 1 DOWNTO 0)
    );

END ROM_A;

ARCHITECTURE Behavioral OF ROM_A IS

    TYPE rom_type IS ARRAY (0 TO (H * M) - 1) OF SIGNED(data_size - 1 DOWNTO 0);
        CONSTANT ROM_Matrix: rom_type := (

        -- Simple Computation
        --       1                2                3
        0 => B"00001", 1  => B"00010", 2  => B"00011",

        -- Max positive value
        --       15               15               15
        3 => B"01111", 4  => B"01111", 5  => B"01111",

        -- Max negative value
        --       -16              -16              -16
        6 => B"10000", 7  => B"10000", 8  => B"10000",

        -- Complex Computation
        --       11               -4               -9
        9 => B"01011", 10 => B"11100", 11 => B"10111",

        -- Catch all
        OTHERS => B"00000"
        );

BEGIN

-- Asynchronous read
Data_out <= ROM_Matrix(TO_INTEGER(Addr));

END Behavioral;
```

ROM B

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     work.DigEng.ALL;

-- Asynchronous Read single-port ROM
    -- Memory Depth => (NxM)
    -- Memory Width => data_size
    -- Data_out signal is respectively Data_B.

ENTITY ROM_B IS
    GENERIC (
        data_size : NATURAL;
        M         : NATURAL;
        N         : NATURAL
    );

    PORT (
        -- Address Input
        Addr     : IN    UNSIGNED (log2(M * N) - 1 DOWNTO 0);
        -- Data Output
        Data_out : OUT   SIGNED (data_size - 1 DOWNTO 0)
    );

END ROM_B;

ARCHITECTURE Behavioral OF ROM_B IS

TYPE rom_type IS ARRAY (0 TO (M * N) - 1) OF SIGNED(data_size - 1 DOWNTO 0);
CONSTANT ROM_Matrix: rom_type := (

-- Simple Comp.  Max +ve Values  Max -ve Values   Complex Comp.   Force Zeros
--       1                15               -16              3                0
0  => B"00001", 1  => B"01111", 2  => B"10000", 3  => B"00011", 4  => B"00000",
--       2                15               -16            -14                0
5  => B"00010", 6  => B"01111", 7  => B"10000", 8  => B"10010", 9  => B"00000",
--       3                15               -16              2                0
10 => B"00011", 11 => B"01111", 12 => B"10000", 13 => B"00010", 14 => B"00000",

-- Catch all
OTHERS => B"00000"
);

BEGIN

-- Asynchronous read
Data_out <= ROM_Matrix(TO_INTEGER(Addr));

END Behavioral;
```

MACC

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     work.DigEng.ALL;

-- Multiply-Accumulate Unit (MACC) for Matrix Multiplication.
    -- Consists of a multiplier coupled with an adder.
    -- Calculates the product-sum of the matrix coefficient as
    --   'Count_M' cycles through.
    -- Computation is complete when 'Count_M' reaches it's max value.


ENTITY MACC IS
    GENERIC (
        data_size : NATURAL;
        M         : NATURAL
    );

    PORT (
        clk      : IN  STD_LOGIC;
        -- Control Inputs
        rst_reg  : IN  STD_LOGIC; -- Register reset
        en       : IN  STD_LOGIC; -- MACC enable
        DataA_in : IN  SIGNED (data_size - 1 DOWNTO 0); -- ROM A data input
        DataB_in : IN  SIGNED (data_size - 1 DOWNTO 0); -- ROM B data input
        -- MACC Data Output
        MACC_out : OUT SIGNED (size(M*2**(2*(data_size-1))) DOWNTO 0)
    );

END MACC;

ARCHITECTURE Behavioral OF MACC IS

    -- Register I/O
    SIGNAL ACC_in  : STD_LOGIC_VECTOR(size(M*2**(2*(data_size-1))) DOWNTO 0);
    SIGNAL ACC_out : STD_LOGIC_VECTOR(size(M*2**(2*(data_size-1))) DOWNTO 0);

BEGIN

-- Parameterizable Register
ACC_reg: ENTITY work.param_register
    GENERIC MAP (
        data_size => size(M*2**(2*(data_size-1))) + 1
    )
    PORT MAP (
        clk      => clk,
        rst_reg  => rst_reg,
        en       => en,
        Data_in  => ACC_in,
        Data_out => ACC_out
    );

-- Product-Sum Operation
ACC_in   <= STD_LOGIC_VECTOR((DataA_in * DataB_in) + SIGNED(ACC_out));

MACC_out <= SIGNED(ACC_out);

END Behavioral;
```

ACC Register

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;

USE     work.DigEng.ALL;

-- N-bit D-Type flip flop with synchronous 'reset' and 'enable'.
    -- Used within the MACC to hold the products of coefficients for summing.

ENTITY param_register IS
    GENERIC (
        data_size : NATURAL
    );

    PORT (
        clk         : IN  STD_LOGIC;
        -- Inputs
        rst_reg     : IN  STD_LOGIC; -- Register reset
        en          : IN  STD_LOGIC; -- Register enable
        Data_in     : IN  STD_LOGIC_VECTOR(data_size - 1 DOWNTO 0);
        -- Output
        Data_out    : OUT STD_LOGIC_VECTOR(data_size - 1 DOWNTO 0)
    );

END param_register;

ARCHITECTURE Behavioral OF param_register IS

BEGIN

Reg: PROCESS (clk)
    BEGIN
        IF (rising_edge(clk)) THEN
            IF (rst_reg = '1') THEN
                Data_out <= (OTHERS => '0');
            ELSIF (en = '1') THEN
                Data_out <= Data_in;
            END IF;
        END IF;
END PROCESS Reg;

END Behavioral;
```

RAM

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     work.DigEng.ALL;

-- Synchronous Write / Asynchrounous Read (NxH)-bit single-port RAM
    -- Memory Depth => (NxH)
    -- Memory Width => size of Matrix_out [see report documentation].

-- Takes Data_out from MACC after coefficient has been calculated.

ENTITY RAM IS
    GENERIC (
        data_size : NATURAL;
        H         : NATURAL;
        M         : NATURAL;
        N         : NATURAL
    );

    PORT (
      clk        : IN  STD_LOGIC;
      -- Inputs
      write_en   : IN  STD_LOGIC; -- RAM write enable
      Addr       : IN  UNSIGNED (log2(H*N) - 1 DOWNTO 0); -- RAM address
      Data_in    : IN  SIGNED (size(M*2**(2*(data_size-1))) DOWNTO 0); -- MACC data in
      -- Output
      RAM_out    : OUT SIGNED (size(M*2**(2*(data_size-1))) DOWNTO 0)
    );

END RAM;

ARCHITECTURE Behavioral OF RAM IS

    -- Internal RAM signals
        -- Internal Address Signal:
            -- Only changes when write_en = '1' to prevent Data_out as undefined.
    SIGNAL Addr_int : UNSIGNED (log2(H*N) - 1 downto 0);
        -- RAM Array: Sets up an internal array.
    TYPE ram_type IS ARRAY (0 TO (N * H) - 1) OF SIGNED(size(M*2**(2*(data_size-1)))
    DOWNTO 0);
        SIGNAL ram_inst: ram_type;

BEGIN

-- Synchronous write (write enable signal)
PROCESS (clk)
    BEGIN
        IF (rising_edge(clk)) THEN
            IF (write_en = '1') THEN
                -- Writes 'Data_in' to memory array of index 'Addr'.
                ram_inst(TO_INTEGER(Addr)) <= Data_in;
                -- Assigns new address input to internal address signal.
                Addr_int <= Addr;
            END IF;
        END IF;
END PROCESS;

-- Asynchronous read
RAM_out <= ram_inst(TO_INTEGER(Addr_int));

END Behavioral;
```

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     work.DigEng.ALL;

-- Datapath for the Matrix Multiplier:
    -- Acts on the instructions sent by the control logic.
    -- Ties all data lines from all memory units (2xRom & 1xRAM)
    --   to the multiply-accumulate unit (MACC).

ENTITY Datapath IS
    GENERIC (
        data_size  : NATURAL;
        H          : NATURAL;
        M          : NATURAL;
        N          : NATURAL
    );

    PORT (
        clk              : IN  STD_LOGIC;
        -- Control Inputs
        macc_en          : IN  STD_LOGIC; -- Enables data to the ACC register
        ram_en           : IN  STD_LOGIC; -- Enables RAM 'write_en'
        rst_reg          : IN  STD_LOGIC; -- Resets ACC register
        Addr_ROM_A       : IN  UNSIGNED (log2(H * M) - 1 DOWNTO 0); -- Address ROM A
        Addr_ROM_B       : IN  UNSIGNED (log2(N * M) - 1 DOWNTO 0); -- Address ROM B
        Addr_RAM         : IN  UNSIGNED (log2(H * N) - 1 DOWNTO 0); -- Address RAM
        -- Matrix Product Output
        Matrix_Product   : OUT SIGNED  (size(M*2**(2*(data_size-1))) DOWNTO 0)
    );

END Datapath;

ARCHITECTURE Behavioral OF Datapath IS

    -- Internal Data Buses
    SIGNAL ROM_DataA   : SIGNED (data_size - 1 DOWNTO 0);
    SIGNAL ROM_DataB   : SIGNED (data_size - 1 DOWNTO 0);
    SIGNAL MACC_out    : SIGNED (size(M*2**(2*(data_size-1))) DOWNTO 0);

BEGIN

-- ROM A
MatrixA_ROM: ENTITY work.ROM_A
    GENERIC MAP (
        data_size => data_size,
        H         => H,
        M         => M
    )
    PORT MAP (
        Addr     => Addr_ROM_A,
        Data_out => ROM_DataA
    );

-- ROM B
MatrixB_ROM: ENTITY work.ROM_B
    GENERIC MAP (
        data_size => data_size,
        M         => M,
        N         => N
    )
    PORT MAP (
        Addr     => Addr_ROM_B,
        Data_out => ROM_DataB
    );

-- MACC
MACC: ENTITY work.MACC
    GENERIC MAP (
        data_size => data_size,
        M         => M
    )
    PORT MAP (
        clk       => clk,
        rst_reg   => rst_reg,
        en        => macc_en,
        DataA_in => ROM_DataA,
        DataB_in => ROM_DataB,
        MACC_out => MACC_out
    );

-- RAM
Matrix_RAM: ENTITY work.RAM
    GENERIC MAP (
        data_size => data_size,
        H          => H,
        M          => M,
        N          => N
    )
    PORT MAP (
        clk       => clk,
        write_en => ram_en,
        Data_in   => MACC_out,
        Addr      => Addr_RAM,
        RAM_out   => Matrix_Product
    );

END Behavioral;
```

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     work.DigEng.ALL;

-- Control Logic for Matrix Multiplier:
    -- Sets the instructions to be sent to the Datapath.
    -- Contains finite-state-machine, parameter counters, and output assignments.

ENTITY Control_Logic IS
    GENERIC (
        data_size  : NATURAL;
        H          : NATURAL;
        M          : NATURAL;
        N          : NATURAL
    );

    PORT (
        clk        : IN    STD_LOGIC;
        -- User Inputs
        rst        : IN    STD_LOGIC;
        nxt        : IN    STD_LOGIC;
        -- Control Outputs
        macc_en    : OUT   STD_LOGIC; -- MACC register enable
        rst_reg    : OUT   STD_LOGIC; -- MACC register reset
        ram_en     : OUT   STD_LOGIC; -- RAM write enable
        -- Address Outputs
            -- Size is computed using the log2 function.
        Addr_ROM_A : OUT   UNSIGNED (log2(H * M) - 1 DOWNTO 0);
        Addr_ROM_B : OUT   UNSIGNED (log2(N * M) - 1 DOWNTO 0);
        Addr_RAM   : OUT   UNSIGNED (log2(H * N) - 1 DOWNTO 0)
    );

END Control_Logic;

ARCHITECTURE Behavioral OF Control_Logic IS

    -- FSM State Definitions
    TYPE Control_states IS (
        WAIT_ST,     -- Wait State: Wait and hold until 'nxt' signal.
        COUNT_ADDR,  -- Count Address: Sets the matrix to the next address.
        CALC_MACC,   -- Calculate: MACC computation for product sum.
        STORE_RAM    -- Store State: Stores data_out from MACC to RAM.
        );

    -- State Type Signals
    SIGNAL state, next_state : Control_states;

    -- Control Signals
    SIGNAL done : STD_LOGIC; -- State "Freeze"
    SIGNAL M_en, H_en, N_en : STD_LOGIC; -- Counter enables

    -- Count Signals
    SIGNAL Count_M : UNSIGNED (log2(M) - 1 DOWNTO 0);
    SIGNAL Count_H : UNSIGNED (log2(H) - 1 DOWNTO 0);
    SIGNAL Count_N : UNSIGNED (log2(N) - 1 DOWNTO 0);

BEGIN

-- State Reset Process
    -- Resets FSM to state = CALC_MACC to compute first coefficient.
state_rst: PROCESS (clk) IS
    BEGIN
        IF RISING_EDGE(clk) THEN
            IF (rst = '1') THEN
                state <= CALC_MACC;
            ELSE
                state <= next_state;
            END IF;
        END IF;
END PROCESS state_rst;


```

```vhdl
-- Counter Entities
    -- Counter Address M
AddrM_Counter: ENTITY work.Param_counter
    GENERIC MAP (
        LIMIT => M )
    PORT MAP (
        clk => clk,
        rst => rst,
        en  => M_en,
        count_out => Count_M
    );

    -- Counter Address H
AddrH_Counter: ENTITY work.Param_counter
    GENERIC MAP (
        LIMIT => H )
    PORT MAP (
        clk => clk,
        rst => rst,
        en  => H_en,
        count_out => Count_H
    );

    -- Counter Address N
AddrN_Counter: ENTITY work.Param_counter
    GENERIC MAP (
        LIMIT => N )
    PORT MAP (
        clk => clk,
        rst => rst,
        en  => N_en,
        count_out => Count_N
    );

-- Transition parameters between FSM states.
    -- Coefficient is calculated on each 'nxt' button press
    --  and freezes in 'WAIT_ST' when matrix is complete.
state_transitions: PROCESS (state, nxt, Count_M) IS
    BEGIN
        CASE state IS
            WHEN WAIT_ST =>
                IF (nxt = '1') THEN
                    next_state <= STORE_RAM;
                ELSE
                    next_state <= state;
                END IF;

            WHEN STORE_RAM =>
                IF (done = '0') THEN
                    next_state <= COUNT_ADDR;
                ELSE -- ELSE IF done = '1', go back to WAIT_ST
                    next_state <= WAIT_ST;
                END IF;

            WHEN COUNT_ADDR =>
                next_state <= CALC_MACC;

            WHEN CALC_MACC =>
                IF (Count_M = M - 1) THEN
                    next_state <= WAIT_ST;
                ELSE
                    next_state <= state;
                END IF;

        END CASE;
END PROCESS state_transitions;

-- OUTPUT ASSIGNMENTS

-- M_Counter Enable
M_en    <=  '0' WHEN state = WAIT_ST    ELSE
            '0' WHEN state = STORE_RAM  ELSE
            '0' WHEN state = COUNT_ADDR ELSE
```

```vhdl
                '1' WHEN state = CALC_MACC  ELSE
                'U';

   -- H_Counter Enable
           -- Count_H increments with Count_N reaches limit.
   H_en    <=   '0' WHEN state = WAIT_ST    ELSE
                '0' WHEN state = STORE_RAM  ELSE
                '1' WHEN state = COUNT_ADDR AND
                         Count_N  = (N - 1) ELSE
                '0' WHEN state = COUNT_ADDR AND
                         Count_N /= (N - 1) ELSE
                '0' WHEN state = CALC_MACC  ELSE
                'U';

   -- N_Counter Enable
   N_en    <=   '0' WHEN state = WAIT_ST    ELSE
                '0' WHEN state = STORE_RAM  ELSE
                '1' WHEN state = COUNT_ADDR ELSE
                '0' WHEN state = CALC_MACC  ELSE
                'U';

   -- Done/Freeze Signal
       -- Dependent on count limits of Count_H and Count_N.
   done    <=   '1' WHEN Count_H = (H - 1)  AND
                         Count_N = (N - 1)  ELSE
                '0';

   -- MACC Calculation Enable
   macc_en <=   '0' WHEN state = WAIT_ST    ELSE
                '0' WHEN state = STORE_RAM  ELSE
                '0' WHEN state = COUNT_ADDR ELSE
                '1' WHEN state = CALC_MACC  ELSE
                'U';

   -- Register Reset
       -- On 'rst' button press and after computation is complete.
   rst_reg <=   rst WHEN state = WAIT_ST    ELSE
                rst WHEN state = STORE_RAM  ELSE
                '1' WHEN state = COUNT_ADDR ELSE
                rst WHEN state = CALC_MACC  ELSE
                'U';

   -- RAM Write Enable
   ram_en  <=   '0' WHEN state = WAIT_ST    ELSE
                '1' WHEN state = STORE_RAM  ELSE
                '0' WHEN state = COUNT_ADDR ELSE
                '0' WHEN state = CALC_MACC  ELSE
                'U';

   -- Address Combinational Logic
   Addr_ROM_A <= RESIZE((Count_H * TO_UNSIGNED(M, log2(M))) + Count_M, log2(M * H));
   Addr_ROM_B <= RESIZE((Count_M * TO_UNSIGNED(N, log2(N))) + Count_N, log2(M * N));
   Addr_RAM   <= RESIZE((Count_H * TO_UNSIGNED(N, log2(N))) + Count_N, log2(N * H));

   END Behavioral;
```

Top Level

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     WORK.DigEng.ALL;

--  __  __      _       _
-- |  \/  |__ _| |_ _ _(_)_ __
-- | |\/| / _` |  _| '_| \ \ /
-- |_|  |_\__,_|\__|_| |_/_\_\
--     __  __      _ _   _      _ _
--    |  \/  |_  _| | |_(_)_ __| (_)___ _ _
--    | |\/| | || | |  _| | '_ \ | / -_) '_|
--    |_|  |_|\_,_|_|\__|_| .__/_|_\___|_|
--                        |_|
--
-- Top Level Source for Matrix Multiplier:
--      Inputs two matrices of sizes [HxM] and [MxN] and outputs
--        the resulting matrix of size [HxN].
--      The next incremental coefficient product is displayed on
--        each user 'nxt' input.
--      Component ties together the user inputs to the
--        Control Logic and Datapath.
--
-- STD_LOGIC signals are expressed as lower case
--    while vector/bus signals are capitalized.

ENTITY Matrix_Multiplier IS
    GENERIC (
        data_size  : NATURAL := 5;
        H          : NATURAL := 4;
        M          : NATURAL := 3;
        N          : NATURAL := 5
    );

    PORT (
        -- Master Clock
        clk             : in  STD_LOGIC;
        -- User Inputs
        rst             : in  STD_LOGIC;
        nxt             : in  STD_LOGIC;
        -- Coefficient Output
        Matrix_Product  : out SIGNED (size(M*2**(2*(data_size-1))) downto 0)
    );

END Matrix_Multiplier;

ARCHITECTURE Behavioral OF Matrix_Multiplier IS

    SIGNAL deb_rst, deb_nxt  : STD_LOGIC;  -- Debounced "reset" and "next" signals.
    SIGNAL macc_en, ram_en, rst_reg   : STD_LOGIC; -- Control/Datapath signal links.
    SIGNAL Addr_ROM_A  : UNSIGNED (log2(H * M) - 1 downto 0); -- Datapath ROM_A address.
    SIGNAL Addr_ROM_B  : UNSIGNED (log2(M * N) - 1 downto 0); -- Datapath ROM_B address.
    SIGNAL Addr_RAM    : UNSIGNED (log2(H * N) - 1 downto 0); -- Datapath RAM address.

BEGIN

-- Debouncer for 'RST' signal
Rst_Debouncer: ENTITY work.Debouncer
    PORT MAP (
        clk     => clk,
        Sig     => rst,
        Deb_Sig => deb_rst
    );

-- Debouncer for 'NXT' signal
Next_Debouncer: ENTITY work.Debouncer
    PORT MAP (
        clk     => clk,
        Sig     => nxt,
        Deb_Sig => deb_nxt
    );
```

```
-- Control Logic
Control_Logic: ENTITY work.Control_Logic
    GENERIC MAP (
        data_size  => data_size,
        H          => H,
        M          => M,
        N          => N
    )
    PORT MAP (
        CLK        => CLK,
        RST        => deb_rst,
        NXT        => deb_nxt,
        macc_en    => macc_en,
        ram_en     => ram_en,
        rst_reg    => rst_reg,
        Addr_ROM_A => Addr_ROM_A,
        Addr_ROM_B => Addr_ROM_B,
        Addr_RAM   => Addr_RAM
    );

-- Datapath
Datapath: ENTITY work.Datapath
    GENERIC MAP (
        data_size => data_size,
        H         => H,
        M         => M,
        N         => N
    )
    PORT MAP (
        CLK            => CLK,
        macc_en        => macc_en,
        ram_en         => ram_en,
        rst_reg        => rst_reg,
        Addr_ROM_A     => Addr_ROM_A,
        Addr_ROM_B     => Addr_ROM_B,
        Addr_RAM       => Addr_RAM,
        Matrix_Product => Matrix_Product
    );

END Behavioral;
```

Matrix Multiplier Testbench

Testbench Process

The test process is designed to test the matrix multiplier in extreme cases of computation, but also to ensure correct operation generally. Our test methods include verifying (in order) a complete matrix computation, a user system reset when idle, a mid-calculation state reset, and a system “freeze” loop. Each test reports in the TCL Console whether it has failed or was successful and displays any relevant data needed to debug.

To ensure the self-checking testbench, the coefficient values on the bottom row are one higher than the true calculated values (coefficients 15 to 19). These coefficients will appear as errors within the TCL Console, however as long as the expected values are one higher than what is observed, the user can refer a correct computation and verify that the testbench is functioning correctly. The calculation and final resulting matrix which is verified in the self-checking testbench is shown below.

Self-Checking Testbench Calculation

ROM A ROM B

$$\boxed{\begin{bmatrix} 1 & 2 & 3 \\ 15 & 15 & 15 \\ -16 & -16 & -16 \\ 11 & -4 & -9 \end{bmatrix} \times \begin{bmatrix} 1 & 15 & -16 & 3 & 0 \\ 2 & 15 & -16 & -14 & 0 \\ 3 & 15 & -16 & 2 & 0 \end{bmatrix} =}$$

$$\downarrow$$

$$\boxed{\begin{bmatrix} 1\times1+2\times2+3\times3 & 1\times15+2\times15+3\times15 & 1\times-16+2\times-16+3\times-16 & 1\times3+2\times-14+3\times2 & 1\times0+2\times0+3\times0 \\ 15\times1+15\times2+15\times3 & 15\times15+15\times15+15\times15 & 15\times-16+15\times-16+15\times-16 & 15\times3+15\times-14+15\times2 & 15\times0+15\times0+15\times0 \\ -16\times1+-16\times2+-16\times3 & -16\times15+-16\times15+-16\times15 & -16\times-16+-16\times-16+-16\times-16 & -16\times3+-16\times-14+-16\times2 & -16\times0+-16\times0+-16\times0 \\ 11\times1+-4\times2+-9\times3 & 11\times15+-4\times15+-9\times15 & 11\times-16+-4\times-16+-9\times-16 & 11\times3+-4\times-14+-9\times2 & 11\times0+-4\times0+-9\times0 \end{bmatrix}}$$

$$\downarrow$$

$$\boxed{= \begin{bmatrix} 14 & 90 & -96 & -19 & 0 \\ 90 & 675 & -720 & -135 & 0 \\ -96 & -720 & 768 & 144 & 0 \\ -24 & -30 & 32 & 71 & 0 \end{bmatrix}}$$

RAM

Testbench

```vhdl
LIBRARY IEEE;
USE     IEEE.STD_LOGIC_1164.ALL;
USE     IEEE.NUMERIC_STD.ALL;
USE     work.DigEng.ALL;
--
--  _____         _   _                   _
-- |_   _|___  ___| |_ | |__  ___  _ _  __ | |_
--   | | / -_)(_-<|  _|| '_ \/ -_)| ' \ / _|| ' \
--   |_| \___|/__/ \__||_.__/\___||_||_|\__||_||_|
--
-- Self-Checking Testbench for Matrix Multiplier Project:
--     Test method includes checking a test array of matrix outputs against
--       the output of the RAM.
--     The 'VERIFY' process tests the functions of 'Reset', 'Next', 'Hold',
--       'Shift Left' and 'Shift Right'.
--
-- Information on each function's tests are displayed in the TCL Console with
--   any relevant data on failures.

ENTITY Matrix_Multiplier_tb IS

END Matrix_Multiplier_tb;

ARCHITECTURE Behavioral OF Matrix_Multiplier_tb IS

    -- Generic/Constant Values
    CONSTANT data_size   : NATURAL := 5;
    CONSTANT H           : NATURAL := 4;
    CONSTANT M           : NATURAL := 3;
    CONSTANT N           : NATURAL := 5;

    -- Internal Signals
    SIGNAL   clk, rst, nxt      : STD_LOGIC; -- Clock & user inputs
    SIGNAL   Matrix_Product     : SIGNED (size(M*2**(2*(data_size-1))) DOWNTO 0);
    -- Clock Period
    CONSTANT clk_period         : TIME := 10 ns;


    -- Test Vector Array
    TYPE Output_Test IS ARRAY (0 TO (H * N) - 1) OF SIGNED
    (size(M*2**(2*(data_size-1))) DOWNTO 0);
        SIGNAL Matrix_Test: Output_Test := (
            -- Verify Computation of matrices A and B.
            0  => (B"00000001110"), 1  => (B"00001011010"), 2  => (B"11110100000"),
            3  => (B"11111101101"), 4  => (B"00000000000"),
            5  => (B"00001011010"), 6  => (B"01010100011"), 7  => (B"10100110000"),
            8  => (B"11101111001"), 9  => (B"00000000000"),
            10 => (B"11110100000"), 11 => (B"10100110000"), 12 => (B"01100000000"),
            13 => (B"00010010000"), 14 => (B"00000000000"),
            -- Verify Testbench (Expected outputs are 1 higher than observed outputs).
            15 => (B"11111101001"), 16 => (B"11111100011"), 17 => (B"00000100001"),
            18 => (B"00001001000"), 19 => (B"00000000001")
        );

BEGIN

-- Unit Under Test: Matrix_Multiplier
UUT: ENTITY work.Matrix_Multiplier
    GENERIC MAP (
        data_size => data_size,
        H         => H,
        M         => M,
        N         => N
    )
    PORT MAP (
        -- Master Clock
        clk            => clk,
        -- User Inputs
        rst            => rst,
        nxt            => nxt,
        -- Outputs
        Matrix_Product => Matrix_Product
    );

```

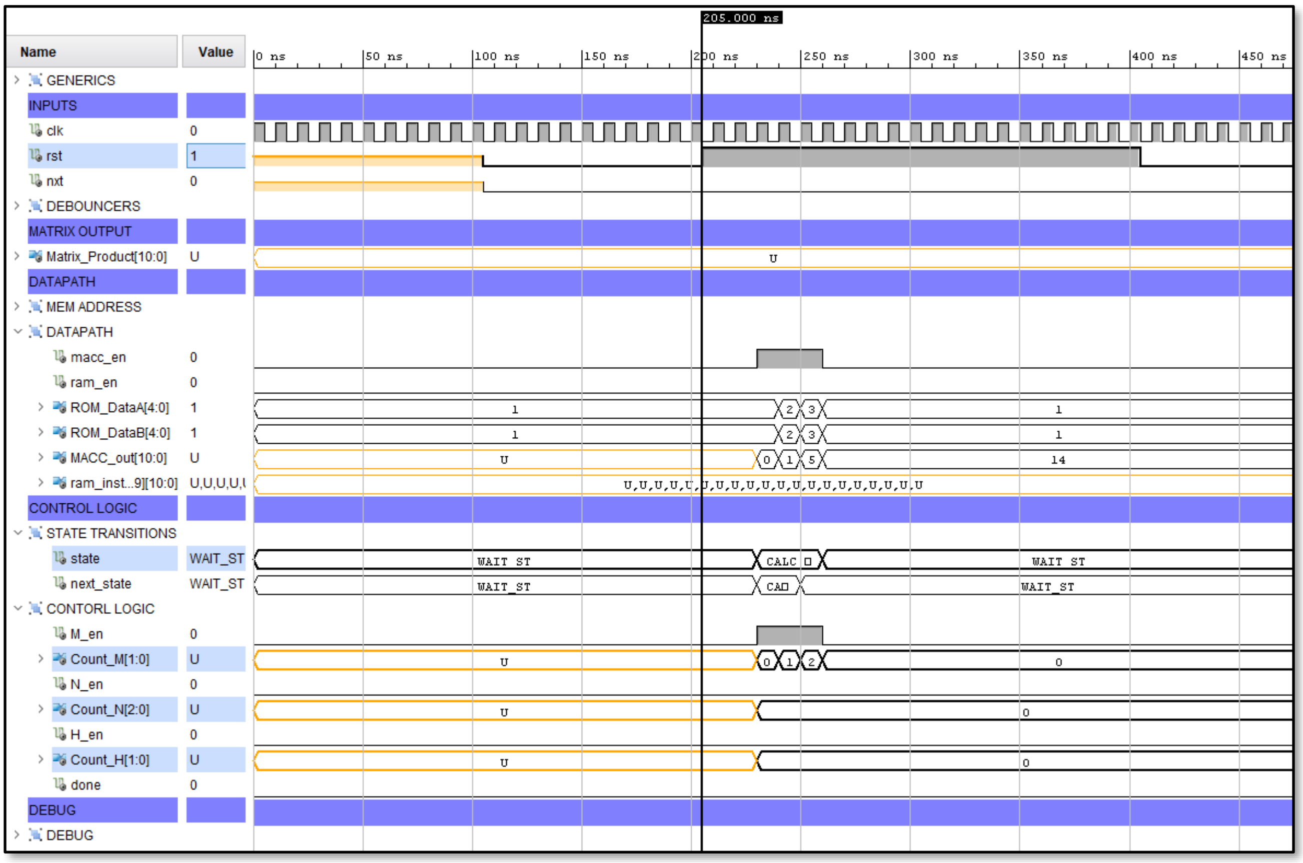
```vhdl
-- Clock Process (Sequential)
clk_process: PROCESS
    BEGIN
        clk <= '1';
        WAIT FOR clk_period/2;
        clk <= '0';
        WAIT FOR clk_period/2;
END PROCESS;

-- Test Method (Sequential)
VERIFY: PROCESS
    BEGIN
        -- Global reset...
        WAIT FOR 100 ns;

        -- Sync to falling edge.
        WAIT UNTIL FALLING_EDGE(clk);

        -- Set internal signals to '0'.
        rst <= '0';
        nxt <= '0';
        WAIT FOR clk_period * 10;

        -- Reset toggle to initialise components.
        rst <= '1';
        WAIT FOR clk_period * 20;
        rst <= '0';
        WAIT FOR clk_period * 20;

-- TEST 1: TESTING COMPLETE MATRIX VALUES SIGNAL:

        -- Verify matrix coefficient 'i' computation.
        FOR i IN Output_Test'RANGE LOOP

            nxt <= '1';
            WAIT FOR clk_period * 20;

                -- Test Failed
            ASSERT (Matrix_Test(i) = Matrix_Product)
            REPORT "TEST 1: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { " & INTEGER'image(i) &
            " }, Expected Output { " & INTEGER'image(TO_INTEGER(Matrix_Test(i))) &
            " }, Observed Output { " & INTEGER'image(TO_INTEGER(Matrix_Product)) & " }"
            SEVERITY error;
                -- Test Successful
            ASSERT (Matrix_Test(i) /= Matrix_Product)
            REPORT "TEST 1: Computation Successful for Coefficient " & INTEGER'image(i) & "!"
            SEVERITY note;

            nxt <= '0';
            WAIT FOR clk_period * 20; -- Wait period to imitate real user input.

        END LOOP;

-- TEST 2: TESTING MACC REGISTER AND COUNTER ADDRESS RESET:

        -- Reset.
        rst <= '1';
        WAIT FOR clk_period * 20;
        rst <= '0';
        WAIT FOR clk_period * 20;

        -- Check system returns to first coefficient.
        nxt <= '1';
        WAIT FOR clk_period * 20;
            -- Test Failed
        ASSERT (Matrix_Test(0) = Matrix_Product)
        REPORT "TEST 2: *** MATRIX RESET FAILED *** =>" &
        " Expected Output { " & INTEGER'image(TO_INTEGER(Matrix_Test(0))) &
        " }, Observed Output { " & INTEGER'image(TO_INTEGER(Matrix_Product)) & " }"
        SEVERITY error;
```

```vhdl
                -- Test Successful
            ASSERT (Matrix_Test(0) /= Matrix_Product)
            REPORT "TEST 2: Reset Successful!"
            SEVERITY note;

            nxt <= '0';
            WAIT FOR clk_period * 20;

    -- TEST 3: TESTING MID-STATE CALCULATION RESET:

            -- Reset.
            rst <= '1';
            WAIT FOR clk_period * 20;
            rst <= '0';
            WAIT FOR clk_period * 20;

            -- Verify first 5 matrix coefficient 'j' computation.
            FOR j IN 0 TO 4 LOOP

                nxt <= '1';
                WAIT FOR clk_period * 20;

                    -- Test Failed
                ASSERT (Matrix_Test(j) = Matrix_Product)
                REPORT "TEST 3: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { "
                & INTEGER'image(j) &
                " }, Expected Output { " & INTEGER'image(TO_INTEGER(Matrix_Test(j))) &
                " }, Observed Output { " & INTEGER'image(TO_INTEGER(Matrix_Product)) & "
                }"
                SEVERITY error;
                    -- Test Successful
                ASSERT (Matrix_Test(j) /= Matrix_Product)
                REPORT "TEST 3: Computation Successful for Coefficient " &
                INTEGER'image(j) & "!"
                SEVERITY note;

                nxt <= '0';
                WAIT FOR clk_period * 20; -- Wait period to imitate real input.

            END LOOP;

            -- Mid-state (CALC_MACC) reset.
            nxt <= '1';
            WAIT FOR clk_period * 4;
            rst <= '1';
            WAIT FOR clk_period * 20;

            nxt <= '0';
            WAIT FOR clk_period * 20;
            rst <= '0';
            WAIT FOR clk_period * 20;

            -- Check system returns to first coefficient.
            nxt <= '1';
            WAIT FOR clk_period * 20;
                -- Test Failed
            ASSERT (Matrix_Test(0) = Matrix_Product)
            REPORT "TEST 3: *** MATRIX MID-STATE RESET FAILED *** =>" &
            " Expected Output { " & INTEGER'image(TO_INTEGER(Matrix_Test(0))) &
            " }, Observed Output { " & INTEGER'image(TO_INTEGER(Matrix_Product)) & " }"
            SEVERITY error;
                -- Test Successful
            ASSERT (Matrix_Test(0) /= Matrix_Product)
            REPORT "TEST 3: Mid-State Reset Successful!"
            SEVERITY note;

            nxt <= '0';
            WAIT FOR clk_period * 20;

    -- TEST 4: TEST "FREEZE" BEHAVIOUR WHEN 'DONE' SIGNAL HIGH

            -- Reset.
            rst <= '1';
```

```vhdl
        WAIT FOR clk_period * 20;
        rst <= '0';
        WAIT FOR clk_period * 20;

        -- Verify all matrix coefficients.
        FOR k IN Output_Test'RANGE LOOP

            nxt <= '1';
            WAIT FOR clk_period * 20;

                -- Test Failed
            ASSERT (Matrix_Test(k) = Matrix_Product)
            REPORT "TEST 4: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { " & INTEGER'image(k) &
            " }, Expected Output { " & INTEGER'image(TO_INTEGER(Matrix_Test(k))) &
            " }, Observed Output { " & INTEGER'image(TO_INTEGER(Matrix_Product)) & " }"
            SEVERITY error;
                -- Test Successful
            ASSERT (Matrix_Test(k) /= Matrix_Product)
            REPORT "TEST 4: Computation Successful for Coefficient " & INTEGER'image(k) & "!"
            SEVERITY note;

            nxt <= '0';
            WAIT FOR clk_period * 20; -- Wait period to imitate real input.

        END LOOP;

        -- Verify 'Done' Signal
        nxt <= '1';
        WAIT FOR clk_period * 20;

            -- Test Failed (Coefficient '19' is 1 higher than true output, thus '- 1')
        ASSERT (TO_INTEGER(Matrix_Test(19) - 1) = TO_INTEGER(Matrix_Product))
        REPORT "TEST 4: *** MATRIX FREEZE FAILED *** => " &
        " Expected Output { " & INTEGER'image(TO_INTEGER(Matrix_Test(19) - 1)) &
        " }, Observed Output { " & INTEGER'image(TO_INTEGER(Matrix_Product)) & " }"
        SEVERITY error;
            -- Test Successful
        ASSERT (TO_INTEGER(Matrix_Test(19) - 1) /= TO_INTEGER(Matrix_Product))
        REPORT "TEST 4: Freeze Behaviour Successful!"
        SEVERITY note;

        -- End Testing
        REPORT "*** TEST COMPLETE ***";
        WAIT; -- Wait forever...

END PROCESS;

END Behavioral;
```

Screenshots of Matrix Multiplier Simulation

Figure 1 - Global and Initial Reset



Name	Value
GENERICS	
INPUTS	
clk	0
rst	1
nxt	0
DEBOUNCERS	
MATRIX OUTPUT	
Matrix_Product[10:0]	U
DATAPATH	
MEM ADDRESS	
DATAPATH	
macc_en	0
ram_en	0
ROM_DataA[4:0]	1
ROM_DataB[4:0]	1
MACC_out[10:0]	U
ram_inst...9][10:0]	U,U,U,U,(
CONTROL LOGIC	
STATE TRANSITIONS	
state	WAIT_ST
next_state	WAIT_ST
CONTORL LOGIC	
M_en	0
Count_M[1:0]	U
N_en	0
Count_N[2:0]	U
H_en	0
Count_H[1:0]	U
done	0
DEBUG	
DEBUG	

Figure 2 - First Coefficient

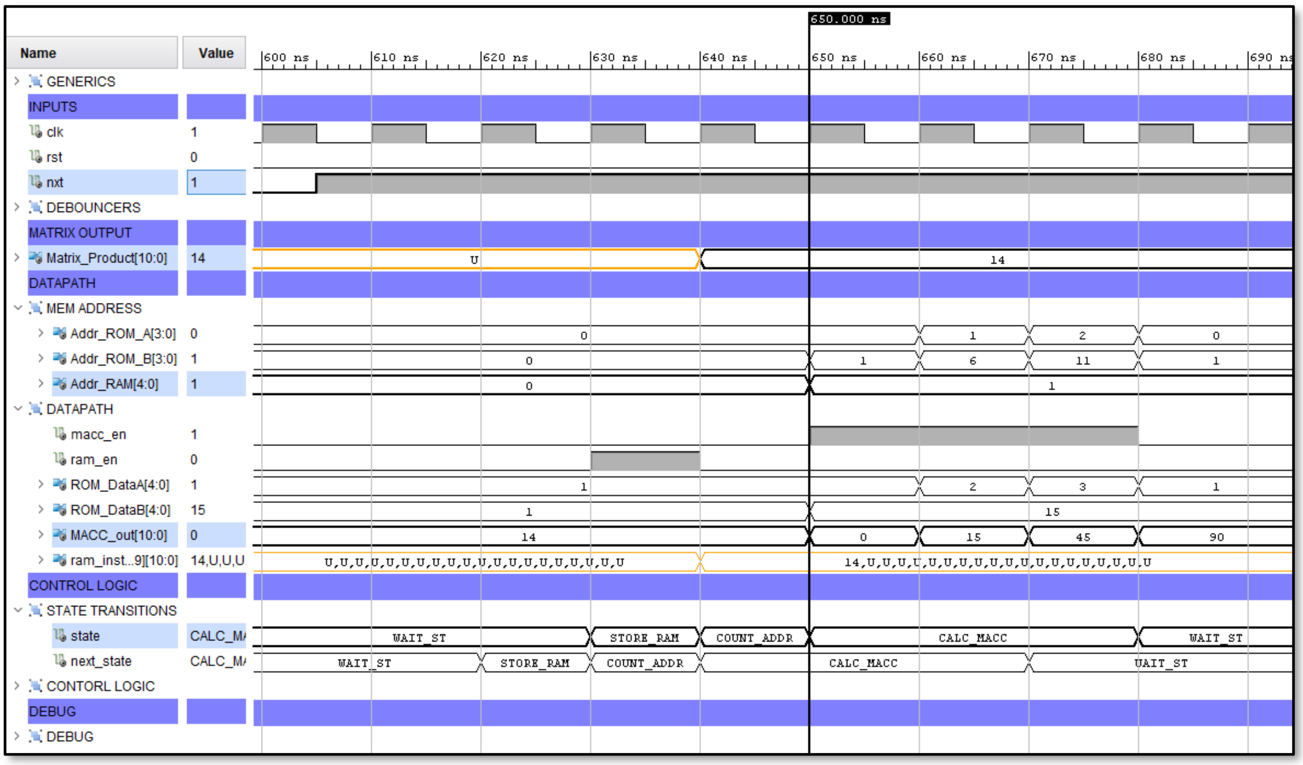


Figure 3 - Control Combinational Logic for Datapath (Address Buses)

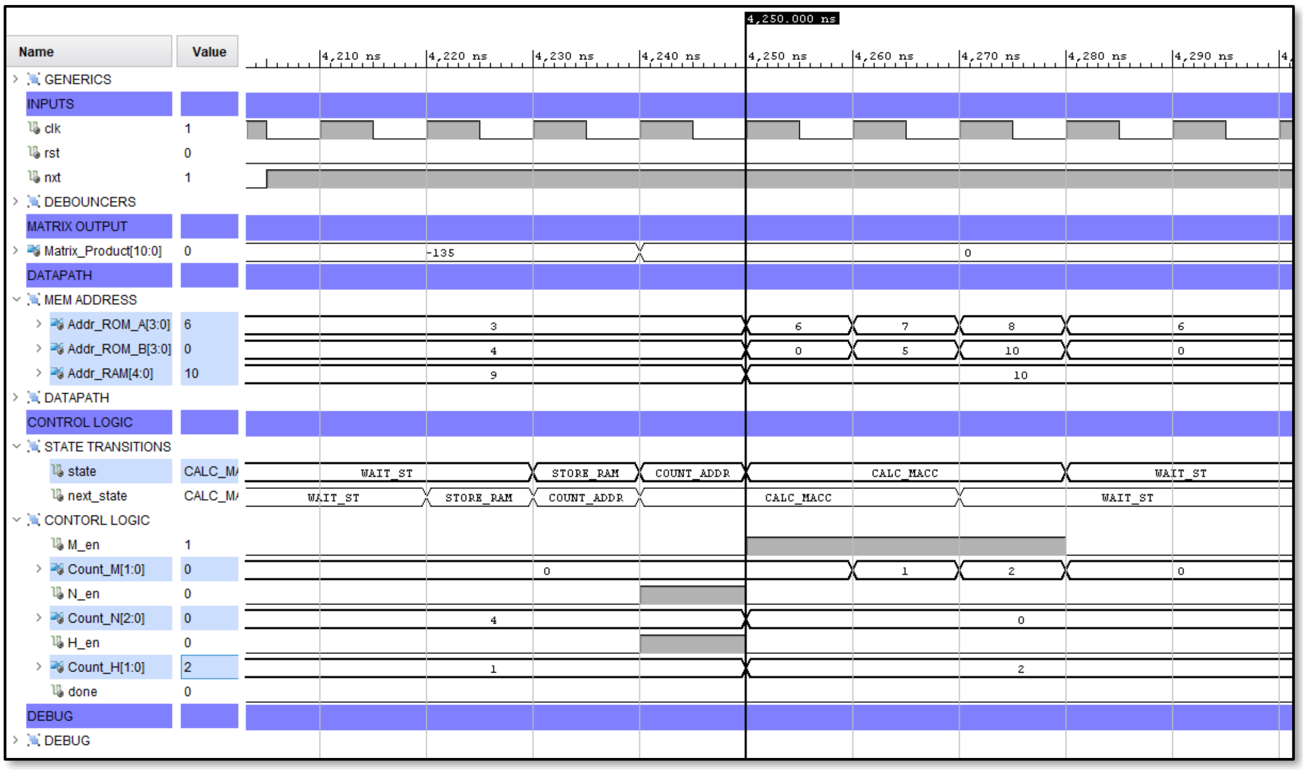


Figure 4 - Done Signal

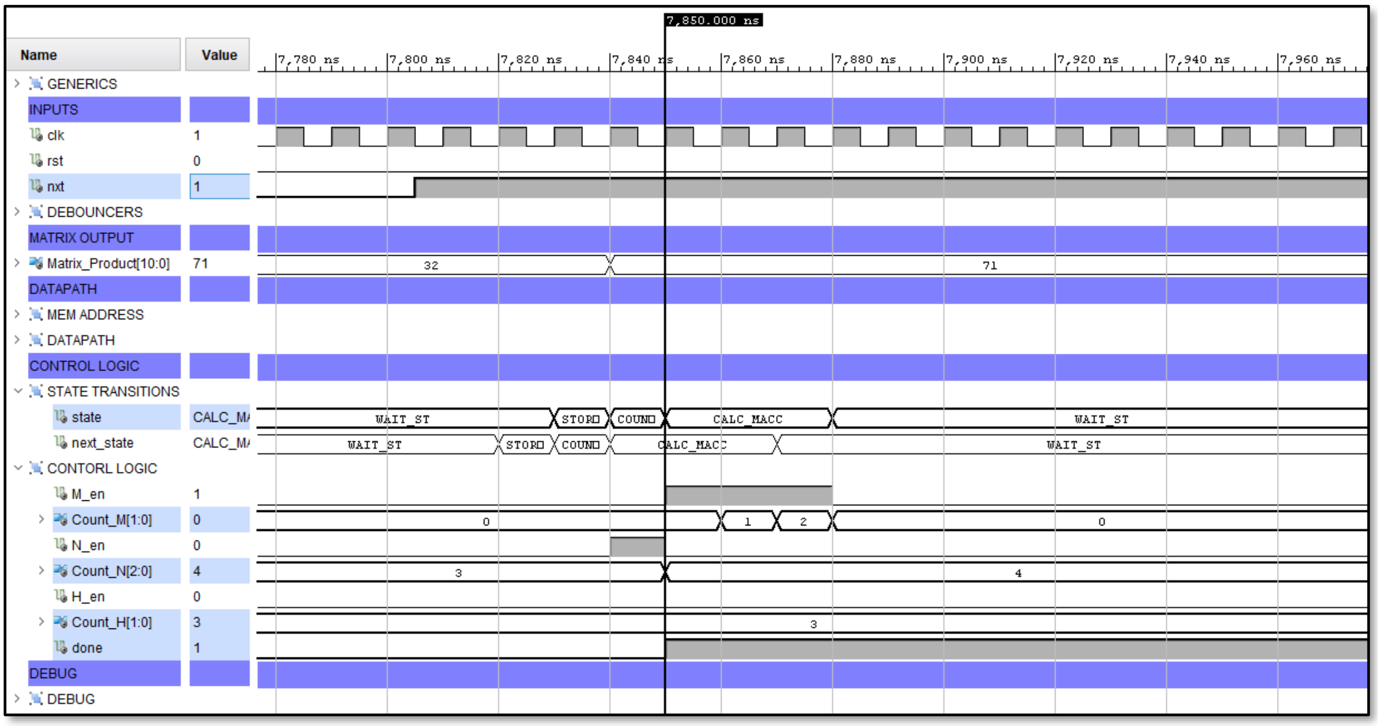


Figure 5 - Last Coefficient

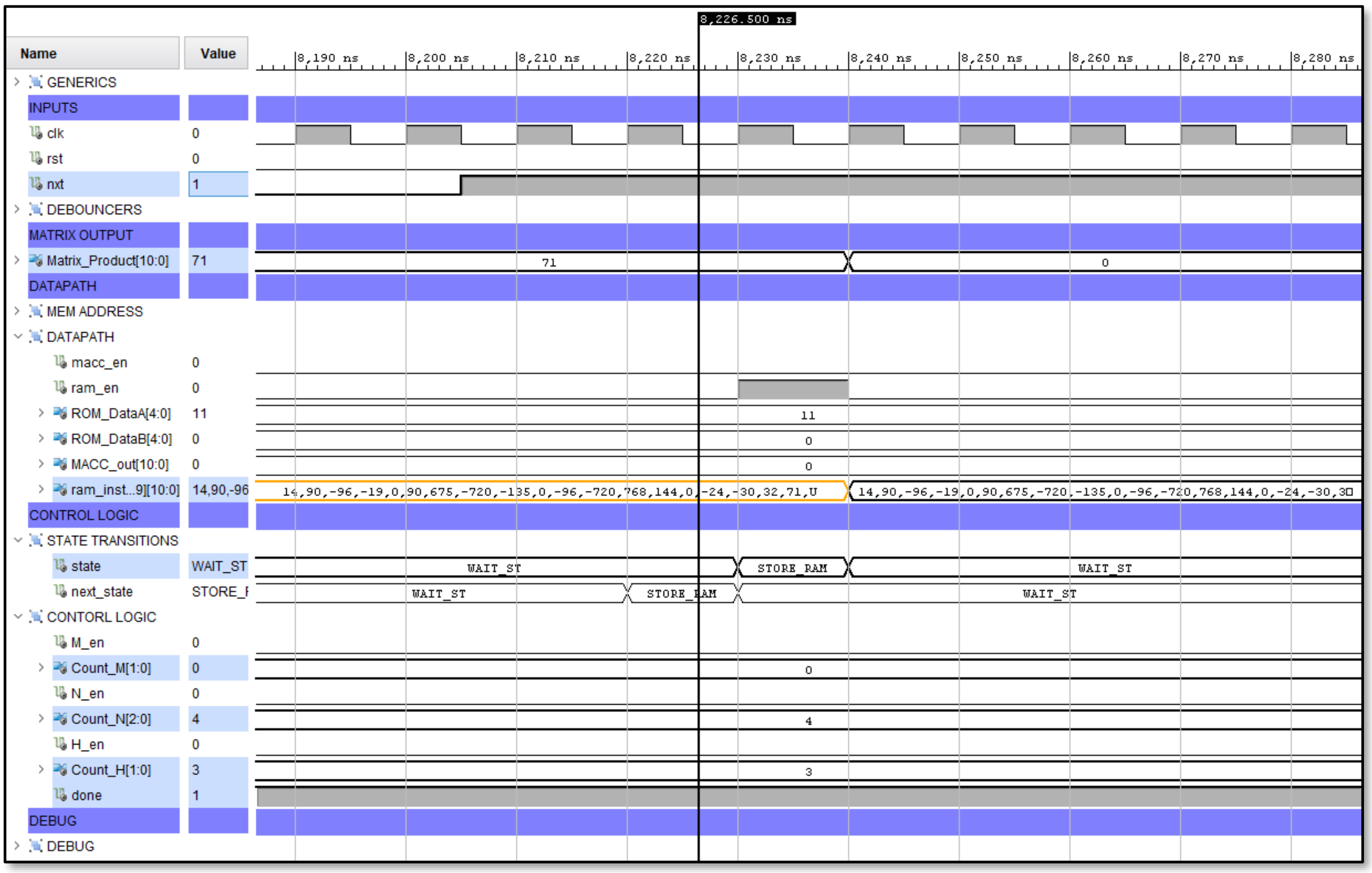


Figure 6 - RAM Memory Overview

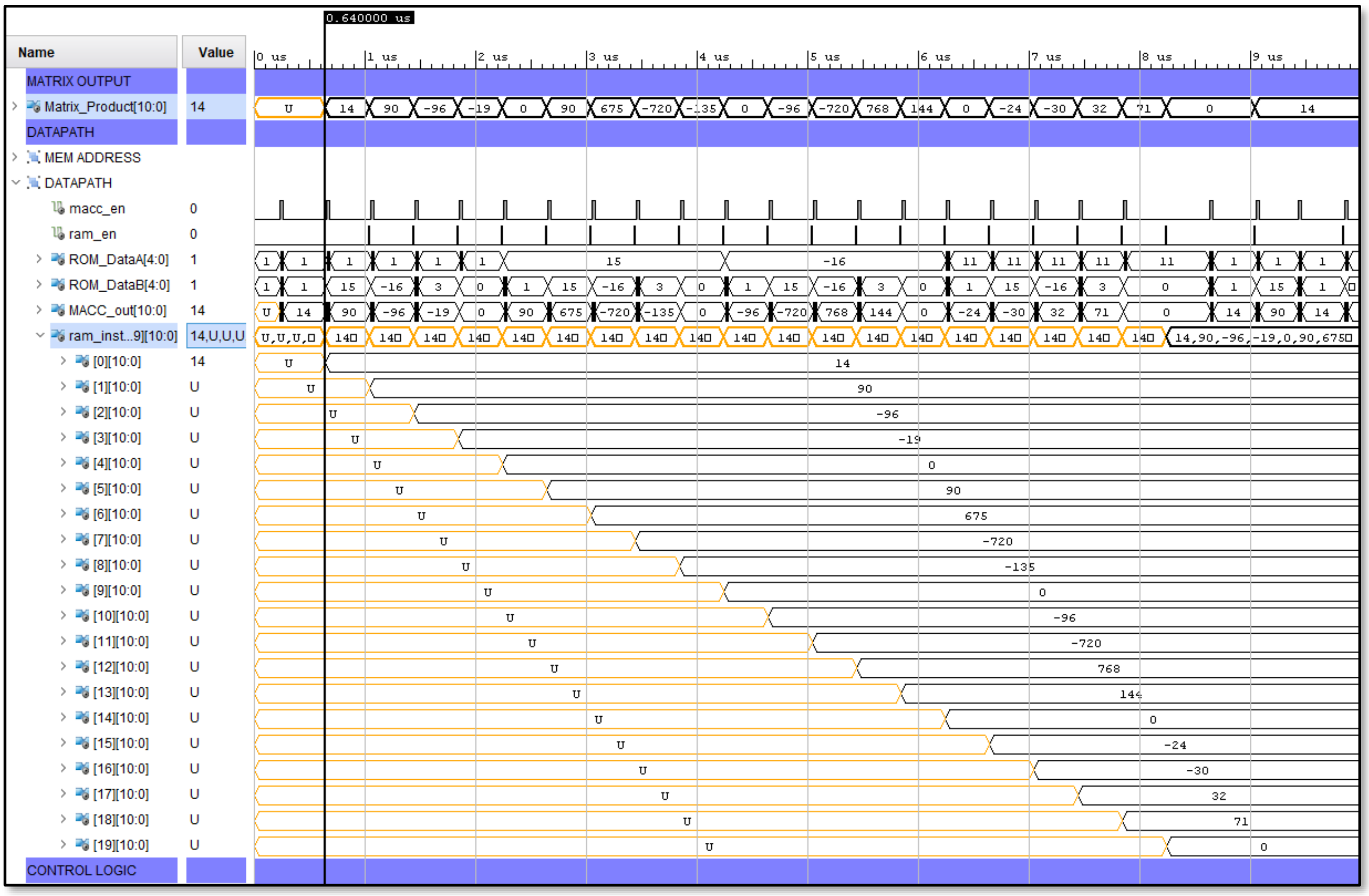


Figure 7 - Complete Matrix Computation (TEST 1)

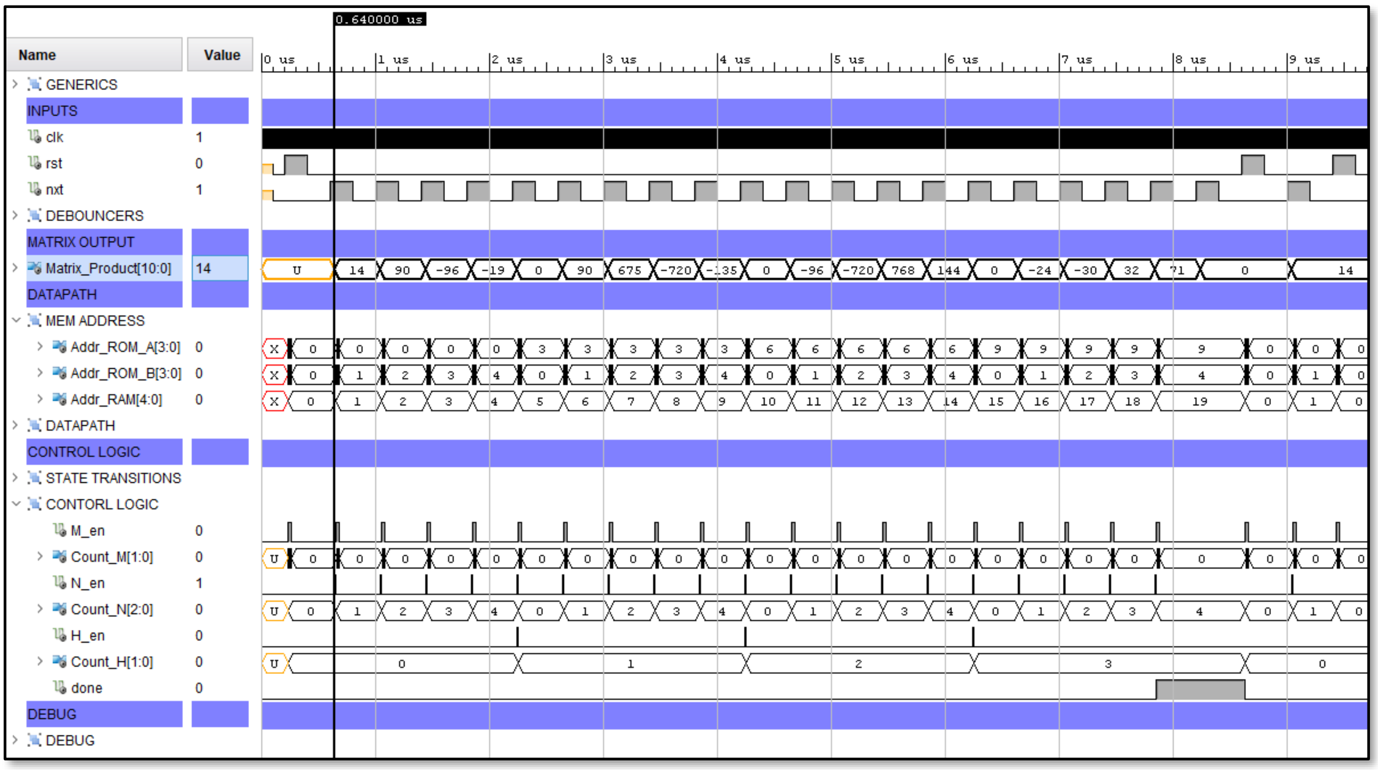


Figure 8 - User Reset (TEST 2)

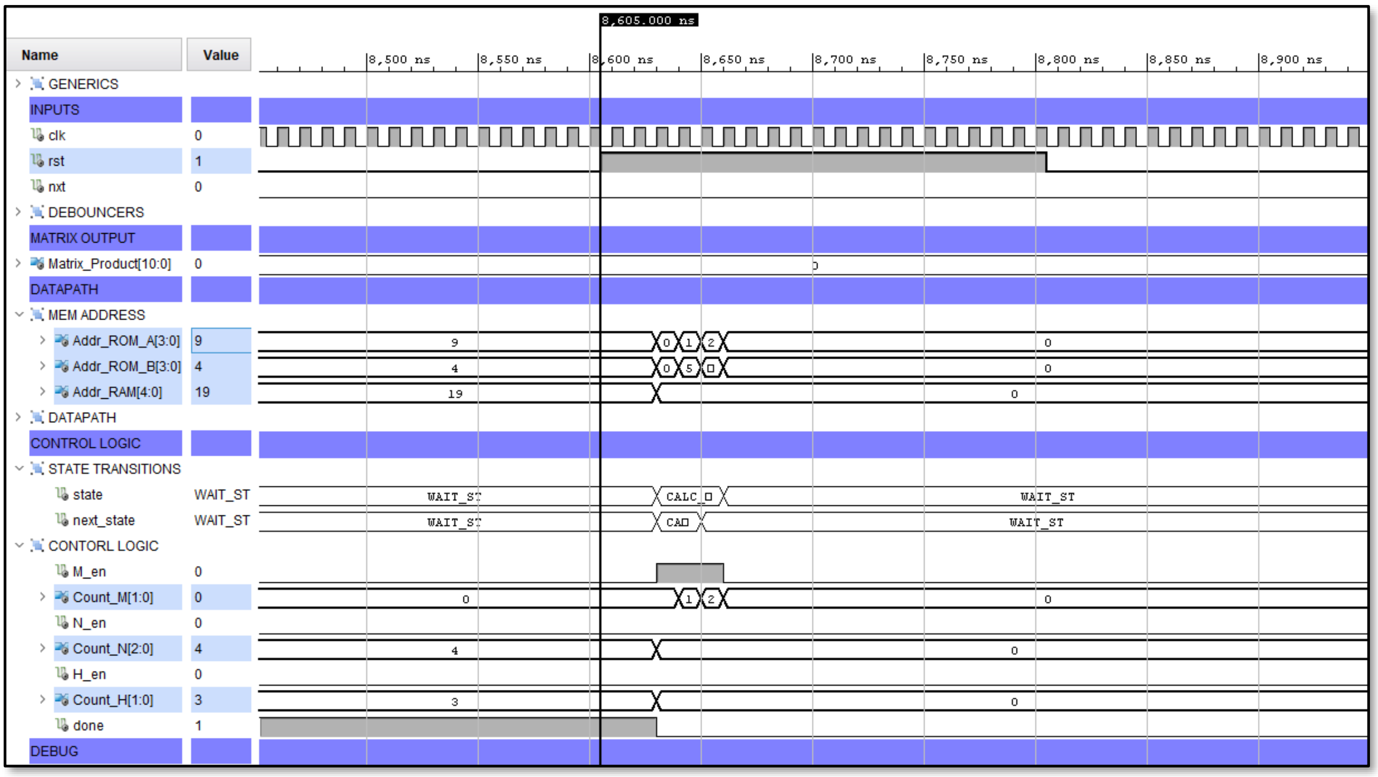


Figure 9 - Mid-State Reset (TEST 3)

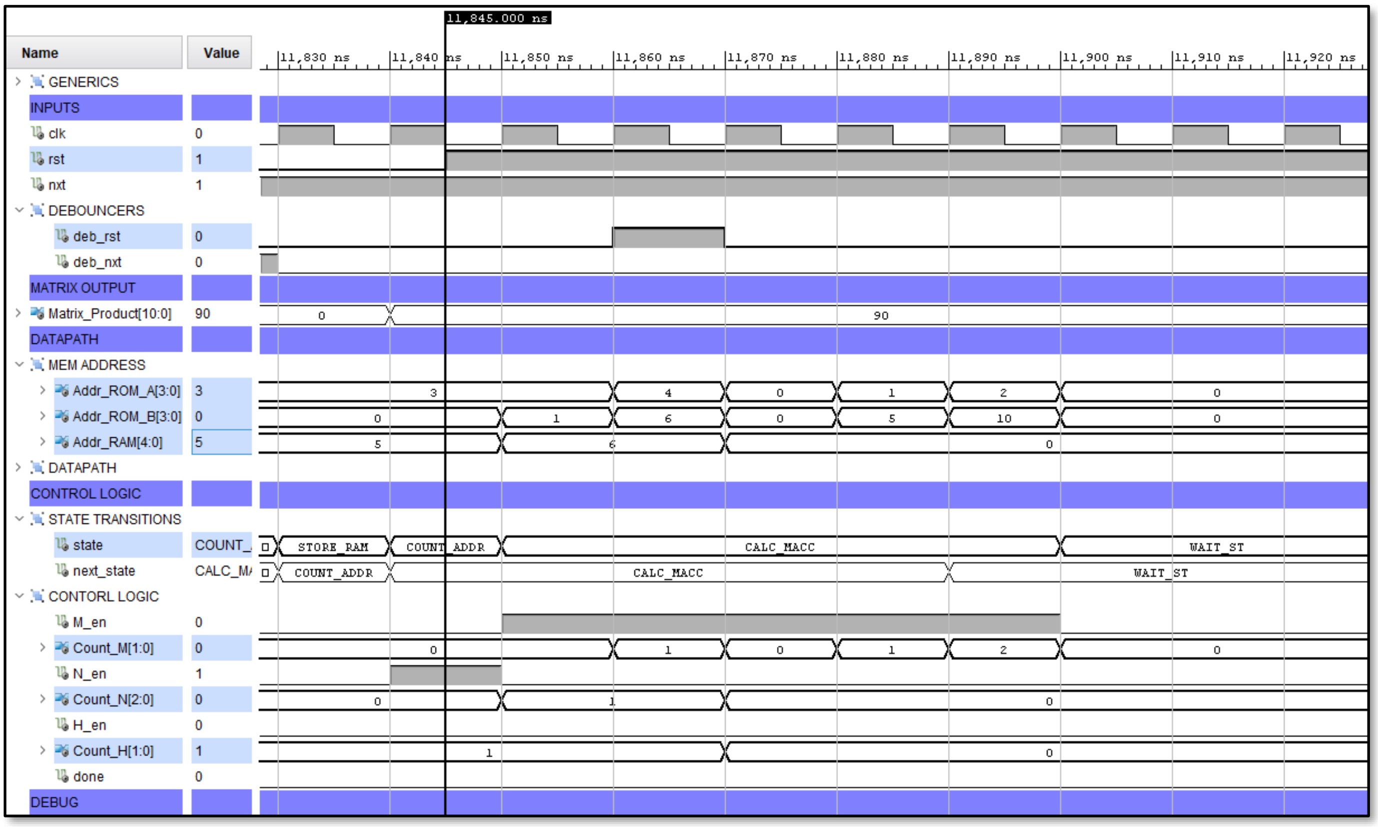


Figure 10 - “Freeze” Behaviour (TEST 4)

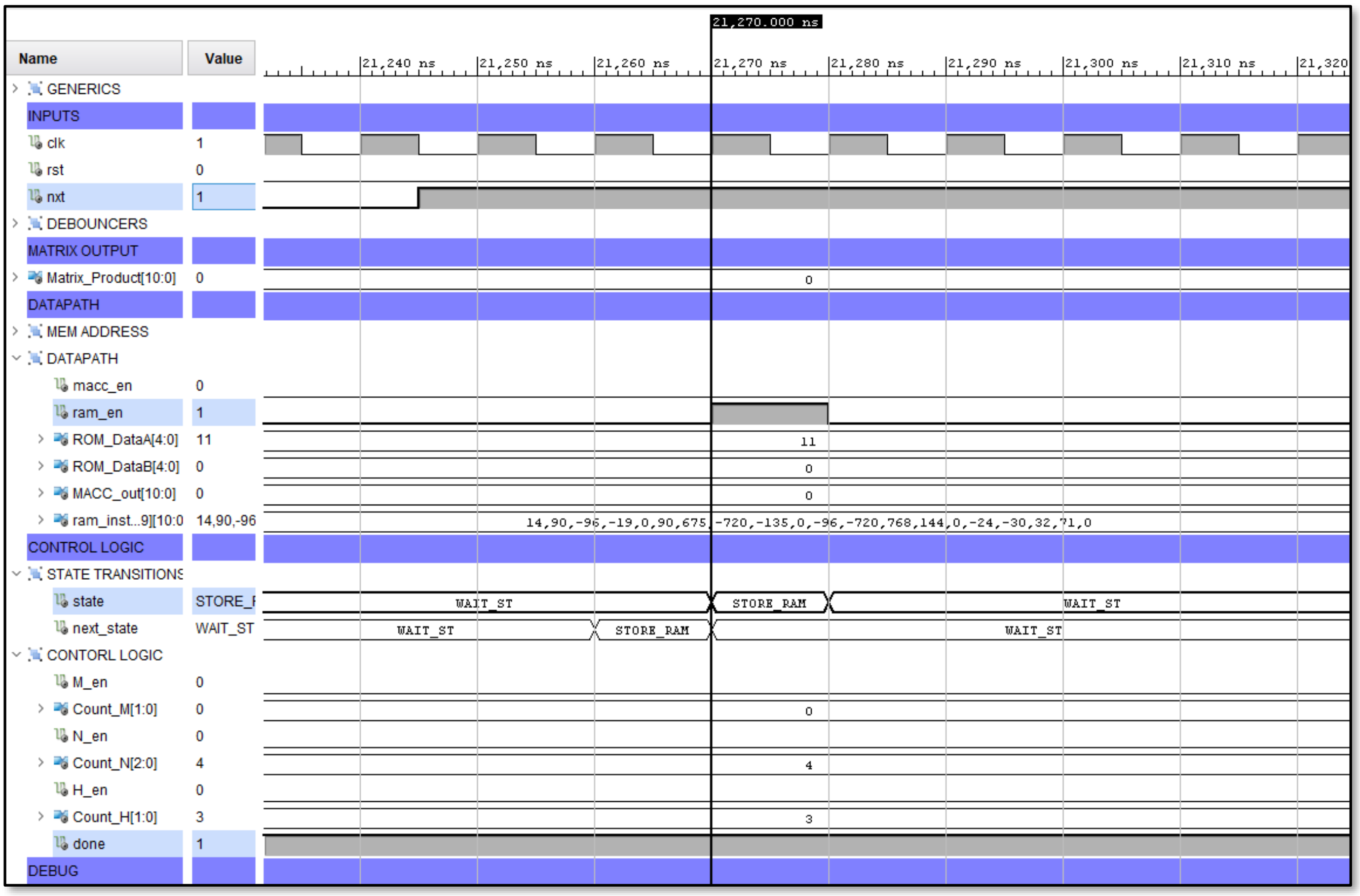
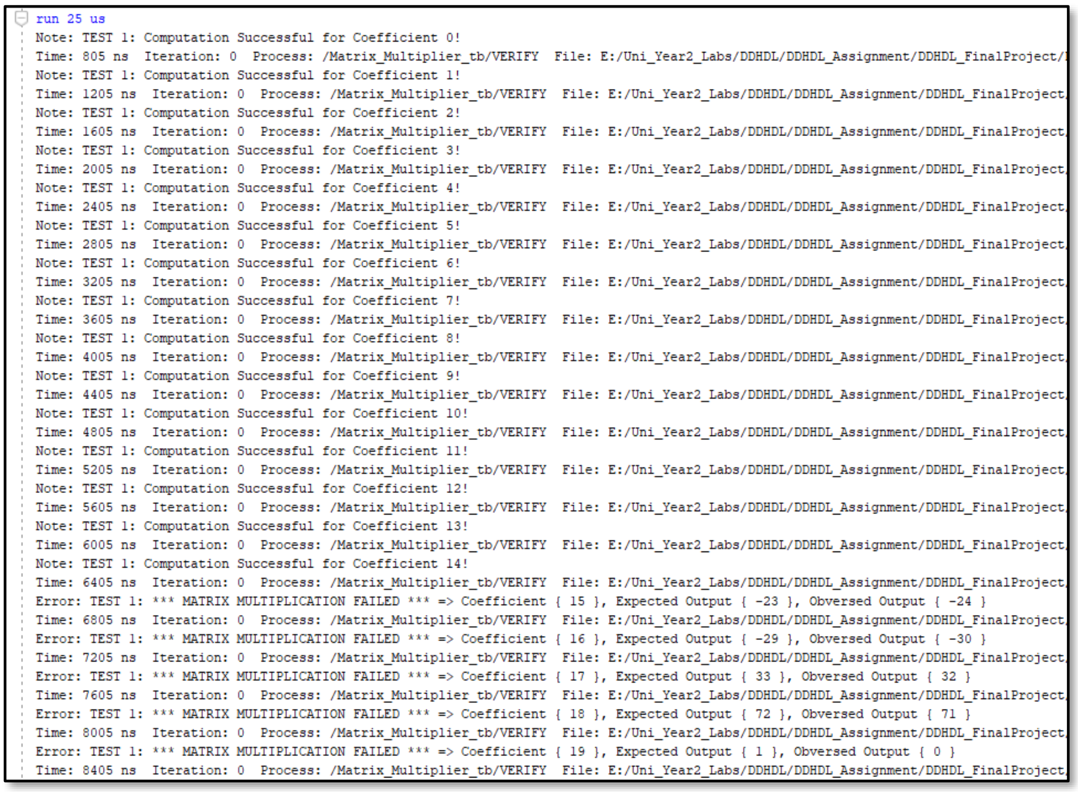
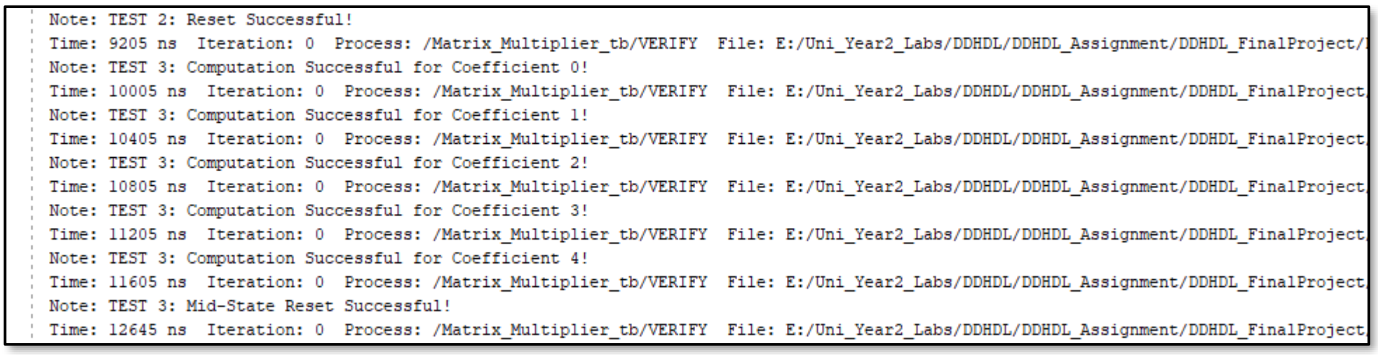


Figure 11 - TCL Console Log (TEST 1: Complete Matrix Computation)



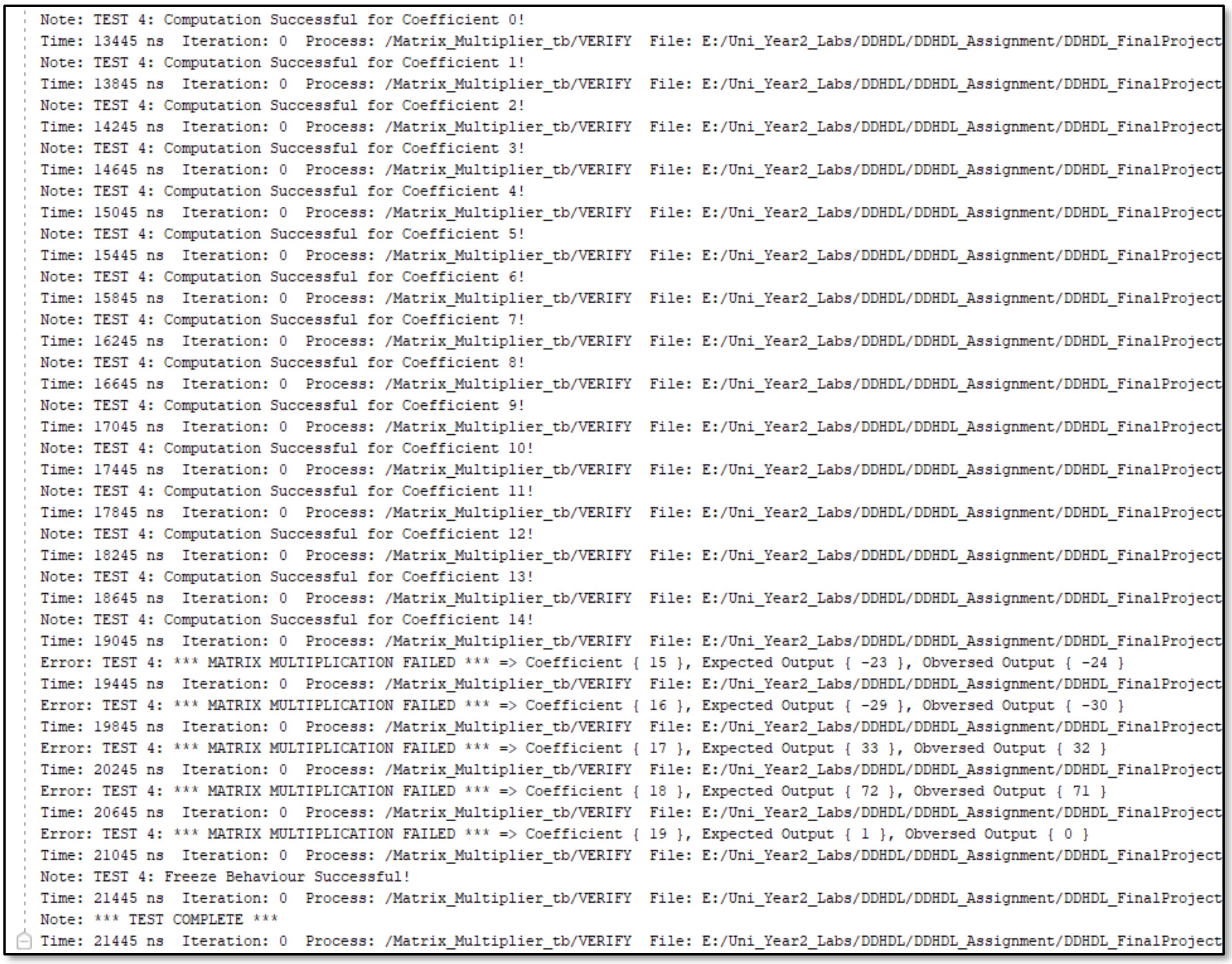
```
run 25 us
Note: TEST 1: Computation Successful for Coefficient 0!
Time: 805 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject/
Note: TEST 1: Computation Successful for Coefficient 1!
Time: 1205 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 2!
Time: 1605 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 3!
Time: 2005 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 4!
Time: 2405 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 5!
Time: 2805 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 6!
Time: 3205 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 7!
Time: 3605 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 8!
Time: 4005 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 9!
Time: 4405 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 10!
Time: 4805 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 11!
Time: 5205 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 12!
Time: 5605 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 13!
Time: 6005 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 1: Computation Successful for Coefficient 14!
Time: 6405 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 1: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 15 }, Expected Output { -23 }, Obversed Output { -24 }
Time: 6805 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 1: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 16 }, Expected Output { -29 }, Obversed Output { -30 }
Time: 7205 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 1: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 17 }, Expected Output { 33 }, Obversed Output { 32 }
Time: 7605 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 1: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 18 }, Expected Output { 72 }, Obversed Output { 71 }
Time: 8005 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 1: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 19 }, Expected Output { 1 }, Obversed Output { 0 }
Time: 8405 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
```

Figure 12 - TCL Console Log (TEST 2 & 3: Reset and Mid-State Reset)



```
Note: TEST 2: Reset Successful!
Time: 9205 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject/
Note: TEST 3: Computation Successful for Coefficient 0!
Time: 10005 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 3: Computation Successful for Coefficient 1!
Time: 10405 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 3: Computation Successful for Coefficient 2!
Time: 10805 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 3: Computation Successful for Coefficient 3!
Time: 11205 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 3: Computation Successful for Coefficient 4!
Time: 11605 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 3: Mid-State Reset Successful!
Time: 12645 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
```

Figure 13 - TCL Console Log (TEST 4: “Freeze” Behaviour)



```
Note: TEST 4: Computation Successful for Coefficient 0!
Time: 13445 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 1!
Time: 13845 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 2!
Time: 14245 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 3!
Time: 14645 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 4!
Time: 15045 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 5!
Time: 15445 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 6!
Time: 15845 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 7!
Time: 16245 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 8!
Time: 16645 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 9!
Time: 17045 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 10!
Time: 17445 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 11!
Time: 17845 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 12!
Time: 18245 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 13!
Time: 18645 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Computation Successful for Coefficient 14!
Time: 19045 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 4: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 15 }, Expected Output { -23 }, Obversed Output { -24 }
Time: 19445 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 4: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 16 }, Expected Output { -29 }, Obversed Output { -30 }
Time: 19845 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 4: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 17 }, Expected Output { 33 }, Obversed Output { 32 }
Time: 20245 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 4: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 18 }, Expected Output { 72 }, Obversed Output { 71 }
Time: 20645 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Error: TEST 4: *** MATRIX MULTIPLICATION FAILED *** => Coefficient { 19 }, Expected Output { 1 }, Obversed Output { 0 }
Time: 21045 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: TEST 4: Freeze Behaviour Successful!
Time: 21445 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
Note: *** TEST COMPLETE ***
Time: 21445 ns  Iteration: 0  Process: /Matrix_Multiplier_tb/VERIFY  File: E:/Uni_Year2_Labs/DDHDL/DDHDL_Assignment/DDHDL_FinalProject
```

RTL Component Statistics

```
Start RTL Component Statistics
---------------------------------------------------------------------------
Detailed RTL Component Info :
+---Adders :
       2 Input      3 Bit       Adders := 1
       2 Input      2 Bit       Adders := 2
+---Registers :
                   11 Bit    Registers := 1
                    5 Bit    Registers := 1
                    3 Bit    Registers := 1
                    2 Bit    Registers := 3
                    1 Bit    Registers := 6
+---RAMs :
                  220 Bit         RAMs := 1
+---Muxes :
      12 Input      5 Bit        Muxes := 1
      15 Input      5 Bit        Muxes := 1
       2 Input      3 Bit        Muxes := 1
       2 Input      2 Bit        Muxes := 3
       4 Input      2 Bit        Muxes := 1
       5 Input      1 Bit        Muxes := 1
       2 Input      1 Bit        Muxes := 1
       4 Input      1 Bit        Muxes := 3
       3 Input      1 Bit        Muxes := 2
---------------------------------------------------------------------------
Finished RTL Component Statistics
```