אורט בראודה
המכללה האקדמית להנדסה

התקנים לוגיים מתוכנתים 31551

דוח מסכם

מגישות:

היבא חמדאן

גנא שלבי

מריה נחלה

תאריך הגשה : 6/09/2024

מחלקה : הנדסת חשמל ואלקטרוניקה

שם המדריך: ד"ר פאדל טריף

תוכן העניינים

1. דרישות תכן :

המערכת מופעלת משעון יחיד של MHz50.

כניסה : לחצן RESET — active low — מגיע מהלחצן שברכבה.

יציאה : תצוגת מצב המערכת. מצב כל ה-LEDs ותצוגת seg-n7 יהיו כבויים למעט LEDG(1) שנמצא בכל אחד ממצבי המערכת.

לחיצה ראשונה על לחצן STARTn גורמת לבלוק data_generator להתחיל להוציא את שני המטריצות.

קצב הצגת 16 האיברים בכל מטריצה הוא תוצאה ללחיצה הראשונה ולאחר כל 16 האיברים של המטריצה קדימה או אחורה המערכת עוצרת ומחכה ללחיצה נוספת.

בזמן קבלת המטריצות המערכת מתעלמת מלחיצות על הלחצנים DISPLAYn-1 STARTn.

גודל המטריצות x44 כאשר כל איבר במטריצה הוא מספר של 8 bits signed (מספרים מוסומנים בין -127 ל+127).

התוצאה תרשם בזיכרון בסדר שהוכן בזיכרון לאחר קבלת תוצאות המטריצות, המערכת תעבור לתצוגת התוצאות.

אם לחצן STARTn נלחץ שוב במהלך תצוגת התוצאות, המערכת תמשיך להתעלם מלחיצות על הלחצנים DISPLAYn-1 ו-STARTn.

המערכת תפעל בסבב של שתי מטריצות ותתקבלנה תוצאות גם אם היא לא לחוצה בזמן כתיבת תוצאות.

לחיצה על לחצן אחר תגרום לשינוי תצוגת מצב המערכת על ה-LED ותצוגת seg-n7.

במהלך קבלת המטריצות, לחיצה על לחצן אחר לא תגרום לשינוי תצוגת מצב המערכת על ה-LED ותצוגת seg-n7.

לאחר שהתוצאה הראשונה של המטריצות תתקבל ותוצג, תופעל לחיצה נוספת שתפעיל את התצוגה הבאה.

במצב תצוגת המטריצות המערכת חוזרת לתצוגת מצב המערכת על DISPLAYn-1 STARTn.

המערכת תמשיך בתצוגה זו עד אשר תתקבל תוצאה ברורה (אין אפשרות לבצע קריאת זיכרון וכתיבה לזיכרון בו זמנית).

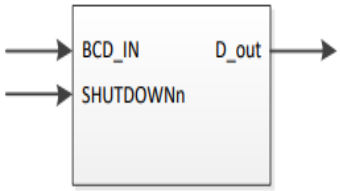
במהלך הצגת התוצאה המערכת מציגה את התוצאה באמצעות הלחצנים במטריצה של seg-n7.

לחיצה על כל לחצן אחר תגרום להצגת התוצאה באופן אוטומטי.

לחיצה נוספת על לחצן STARTn תגרום להפסקת התצוגה ולהפעלת המערכת מחדש לתחילת התהליך.

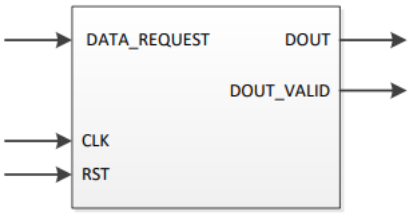
2. קיצור לפעולת הבלוקים שקיבלנו מד"ר פאדל :

מימוש הבלוק bcd_to_7seg :



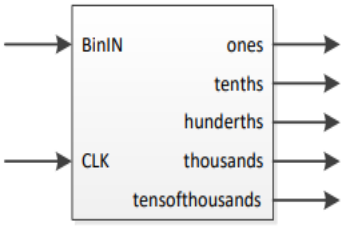
הקוד ממיר ערך BCD בן 4 ביט (0-9) לתצוגת 7 סגמנטים על ידי קביעת אילו סגמנטים ידלקו עבור כל ספרה.

מימוש הבלוק data_generator :



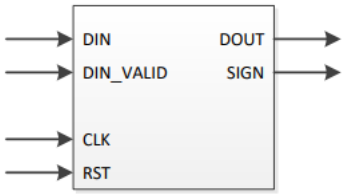
הבלוק מתפקד כגנרטור נתונים שמייצר ומעביר נתונים מחוץ לרכיב בצורה מאורגנת ומבוקרת, כאשר כל פעם שמתקבל DATA_REQUEST, הוא שולח סדרה של נתונים מהמטריצות שלו דרך האות DOUT.

מימוש הבלוק bin2bcd_12bit:



ממיר מספר בינארי בגודל 16 סיבית (12 סיביות לוגיות בלבד) לפורמט BCD עם 5 ספרות (כולל עשרות אלפים, אלפים, מאות, עשרות, ויחידות).

מימוש הבלוק num_convert:

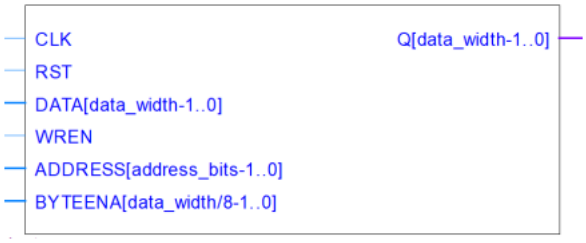


הרכיב לוקח מספר בינארי עם סימן, מפריד את הסימן וממיר את המספר לחיובי, ומוציא את התוצאה יחד עם הסימן המקורי.

מימוש הבלוק synic_diff:

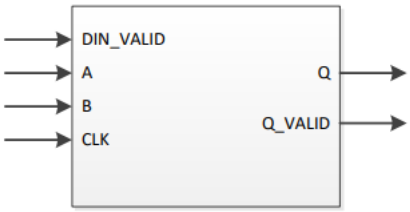
הרכיב מסנכרן אות כניסה אסינכרוני עם שעון המערכת ומפיק פלט שמציין שינוי באות בהתאם להגדרה.

מימוש הבלוק matrix_ram:



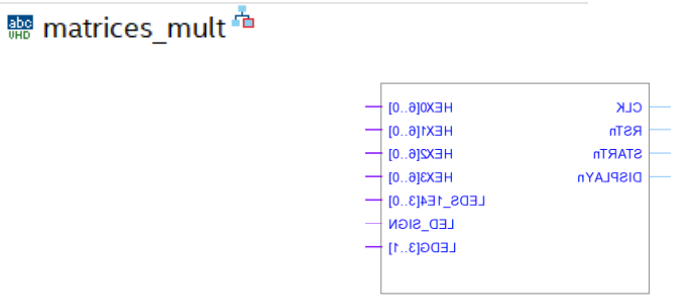
הקוד מייצג רכיב זיכרון RAM שניתן לכתוב ולקרוא ממנו נתונים על פי כתובת.

מימוש הבלוק my_multiplier:



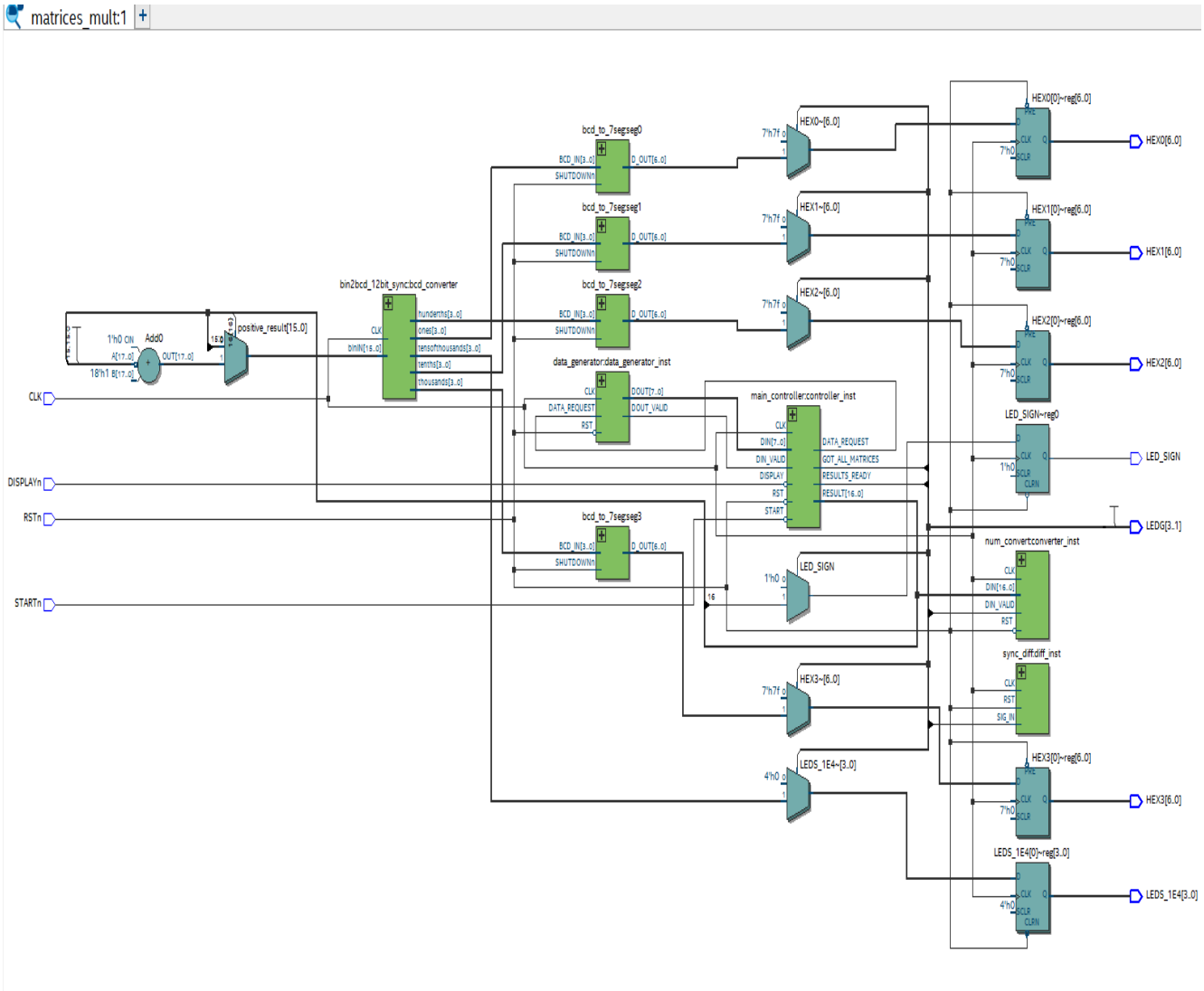
הרכיב מבצע כפל בין שני מספרים בינאריים.

3. ממיוש הרמה העליונה matrices_mult :



הקוד מגדיר מערכת בשם matrices_mult, שמבצעת כפל מטריצות ומציגה את התוצאה על תצוגות 7-segment7 ו-LEDs. המערכת כוללת מספר רכיבים עיקריים, כולל בקר ראשי (main controller), מחולל נתונים (data generator), ממיר בינארי ל-BCD, וממיר BCD לתצוגת 7-segment

3.1 מבנה לוגי :



3.2 קובץ matrices_mult.vhd :

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- Entity Declaration: matrices_mult
-- This module handles matrix multiplication and displays the result on 7-segment displays and LEDs.
entity matrices_mult is
    port (
        RSTn        : in std_logic;                    -- Asynchronous reset (active low)
        CLK         : in std_logic;                    -- System clock
        STARTn      : in std_logic;                    -- Start signal (active low)
        DISPLAYn    : in std_logic;                    -- Display enable signal (active low)
        HEX0        : out std_logic_vector(6 downto 0);-- 7-segment display for units digit
        HEX1        : out std_logic_vector(6 downto 0);-- 7-segment display for tens digit
        HEX2        : out std_logic_vector(6 downto 0);-- 7-segment display for hundreds digit
        HEX3        : out std_logic_vector(6 downto 0);-- 7-segment display for thousands digit
        LEDS_1E4    : out std_logic_vector(3 downto 0);-- LED display for ten thousands digit
        LED_SIGN    : out std_logic;                   -- LED to indicate the sign of the result
        LEDG        : out std_logic_vector(3 downto 1) -- General-purpose LEDs
    );
end entity matrices_mult;

-- Architecture Declaration: structured
-- This architecture implements the matrix multiplication and its display logic.
architecture structured of matrices_mult is

    -- Intermediate signals for component connections
    signal req_data        : std_logic;                     -- Data request signal
    signal input_data      : std_logic_vector(7 downto 0);  -- Input data from data generator
    signal valid_data      : std_logic;                     -- Valid data signal
    signal output_result   : std_logic_vector(16 downto 0); -- Output result from controller
    signal result_ready    : std_logic;                     -- Result ready signal
    signal matrices_loaded : std_logic;                     -- Matrices loaded signal
    signal bcd_units       : std_logic_vector(3 downto 0);  -- BCD digit for units
    signal bcd_tens        : std_logic_vector(3 downto 0);  -- BCD digit for tens
    signal bcd_hundreds    : std_logic_vector(3 downto 0);  -- BCD digit for hundreds
    signal bcd_thousands   : std_logic_vector(3 downto 0);  -- BCD digit for thousands
    signal bcd_ten_thousands: std_logic_vector(3 downto 0); -- BCD digit for ten thousands
    signal hex_display0    : std_logic_vector(6 downto 0);  -- 7-segment output for units digit
    signal hex_display1    : std_logic_vector(6 downto 0);  -- 7-segment output for tens digit
    signal hex_display2    : std_logic_vector(6 downto 0);  -- 7-segment output for hundreds digit
    signal hex_display3    : std_logic_vector(6 downto 0);  -- 7-segment output for thousands digit
    signal num_sign        : std_logic;                     -- Signal for the sign of the number
    signal processed_data  : std_logic_vector(15 downto 0); -- Processed data signal
    signal signal_out      : std_logic;                     -- Synchronized signal output
    signal is_negative     : std_logic;                     -- Sign bit for the result
    signal positive_result : std_logic_vector(15 downto 0); -- Positive part of the result

    -- Component Declaration: main_controller
    -- Handles the core functionality of matrix multiplication.
    component main_controller
        port (
            RST             : in std_logic;                     -- Reset signal
            CLK             : in std_logic;                     -- System clock
            START           : in std_logic;                     -- Start signal
            DISPLAY         : in std_logic;                     -- Display enable signal
            DIN             : in std_logic_vector(7 downto 0);  -- Data input
            DIN_VALID       : in std_logic;                     -- Data valid signal
            DATA_REQUEST    : out std_logic;                    -- Data request signal
```

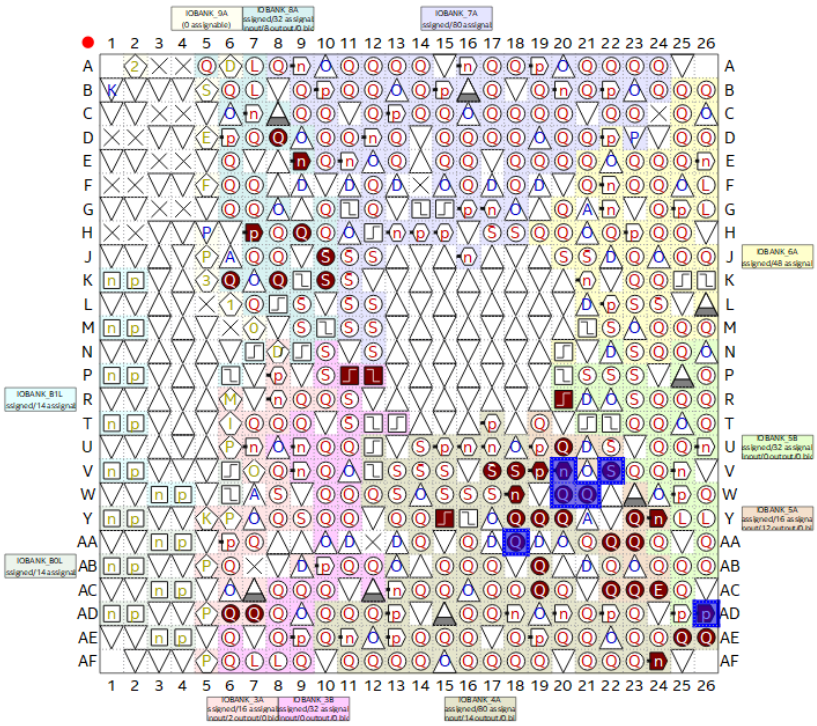
```
            DATA_REQUEST     : out std_logic;                      -- Data request signal
            RESULT           : out std_logic_vector(16 downto 0);-- Multiplication result
            RESULTS_READY    : out std_logic;                      -- Result ready signal
            GOT_ALL_MATRICES: out std_logic                        -- Signal indicating all matrices are loaded
        );
    end component;

    -- Component Declaration: data_generator
    -- Generates the input data for matrix multiplication.
    component data_generator
        port (
            RST          : in std_logic;                       -- Reset signal
            CLK          : in std_logic;                       -- System clock
            DATA_REQUEST: in std_logic;                        -- Data request signal
            DOUT         : out std_logic_vector(7 downto 0); -- Data output
            DOUT_VALID   : out std_logic                       -- Data valid signal
        );
    end component;

    -- Component Declaration: bin2bcd_12bit_sync
    -- Converts a 12-bit binary number to BCD format.
    component bin2bcd_12bit_sync
        port (
            CLK              : in std_logic;                       -- System clock
            binIN            : in std_logic_vector(15 downto 0); -- Binary input
            ones             : out std_logic_vector(3 downto 0); -- BCD output for ones
            tenths           : out std_logic_vector(3 downto 0); -- BCD output for tens
            hunderths        : out std_logic_vector(3 downto 0); -- BCD output for hundreds
            thousands        : out std_logic_vector(3 downto 0); -- BCD output for thousands
            tensofthousands : out std_logic_vector(3 downto 0)  -- BCD output for ten thousands
        );
    end component;

    -- Component Declaration: bcd_to_7seg
    -- Converts a BCD digit to a 7-segment display format.
    component bcd_to_7seg
        port (
            BCD_IN      : in std_logic_vector(3 downto 0); -- BCD input
            SHUTDOWNn   : in std_logic;                      -- Shutdown signal (active low)
            D_OUT       : out std_logic_vector(6 downto 0) -- 7-segment display output
        );
    end component;

    -- Component Declaration: num_convert
    -- Converts a 17-bit signed number to a 16-bit signed number and extracts the sign.
    component num_convert
        port (
            RST        : in std_logic;                       -- Reset signal
            CLK        : in std_logic;                       -- System clock
            DIN        : in std_logic_vector(16 downto 0);  -- Data input
            DIN_VALID : in std_logic;                       -- Data valid signal
            DOUT       : out std_logic_vector(15 downto 0); -- Data output
            SIGN       : out std_logic                       -- Sign output
        );
    end component;

    -- Component Declaration: sync_diff
    -- Synchronizes the result ready signal.
    component sync_diff
```

```
        generic (
            G_DERIVATE_RISING_EDGE  : boolean := true;          -- Derivative on rising edge
            G_SIG_IN_INIT_VALUE     : std_logic := '0';         -- Initial value of the signal input
            G_RESET_ACTIVE_VALUE    : std_logic := '0'          -- Reset active value
        );
        port (
            CLK     : in std_logic; -- System clock
            RST     : in std_logic; -- Reset signal
            SIG_IN  : in std_logic; -- Signal input
            SIG_OUT : out std_logic -- Signal output
        );
    end component;

begin

    -- Data Generator Instantiation
    data_generator_inst: data_generator
        port map (
            RST          => not RSTn,  -- Active-low reset
            CLK          => CLK,       -- System clock
            DATA_REQUEST=> req_data,  -- Data request signal
            DOUT         => input_data,-- Data output
            DOUT_VALID   => valid_data -- Data valid signal
        );

    -- Main Controller Instantiation
    controller_inst: main_controller
        port map (
            RST              => not RSTn,       -- Active-low reset
            CLK              => CLK,            -- System clock
            START            => not STARTn,     -- Active-low start signal
            DISPLAY          => not DISPLAYn,   -- Active-low display enable signal
            DATA_REQUEST     => req_data,       -- Data request signal
            DIN              => input_data,     -- Data input
            DIN_VALID        => valid_data,     -- Data valid signal
            RESULT           => output_result,  -- Multiplication result
            RESULTS_READY    => result_ready,   -- Result ready signal
            GOT_ALL_MATRICES=> matrices_loaded -- Matrices loaded signal
        );

    -- Process to handle sign detection and result conversion
    process(output_result)
    begin
        is_negative <= output_result(16);  -- Assign the sign bit to the is_negative signal
        case output_result(16) is
            when '1' =>  -- If the sign bit is 1, the result is negative
                positive_result <= std_logic_vector((-signed(output_result(15 downto 0)))); -- Convert to positive
            when others =>  -- If the sign bit is 0, the result is positive
                positive_result <= output_result(15 downto 0); -- No modification needed
        end case;
    end process;

    -- Binary to BCD Conversion
    bcd_converter: bin2bcd_12bit_sync
        port map (
            CLK              => CLK,            -- System clock
            binIN            => positive_result,-- Binary input
            ones             => bcd_units,      -- BCD output for units
            tenths           => bcd_tens,       -- BCD output for tens
```

```
            hunderths       => bcd_hundreds,   -- BCD output for hundreds
            thousands       => bcd_thousands,  -- BCD output for thousands
            tensofthousands => bcd_ten_thousands -- BCD output for ten thousands
        );

    -- Convert Result to Displayable Number
    converter_inst: num_convert
        port map (
            RST       => not RSTn,           -- Active-low reset
            CLK       => CLK,                -- System clock
            DIN       => output_result,      -- Data input
            DIN_VALID => result_ready,       -- Data valid signal
            DOUT      => processed_data,     -- Processed data output
            SIGN      => num_sign            -- Sign output
        );

    -- BCD to 7-Segment Display Conversion
    seg0: bcd_to_7seg
        port map (
            BCD_IN     => bcd_units,         -- BCD input for units
            SHUTDOWNn  => RSTn,              -- Shutdown signal (active low)
            D_OUT      => hex_display0       -- 7-segment display output for units
        );

    seg1: bcd_to_7seg
        port map (
            BCD_IN     => bcd_tens,          -- BCD input for tens
            SHUTDOWNn  => RSTn,              -- Shutdown signal (active low)
            D_OUT      => hex_display1       -- 7-segment display output for tens
        );

    seg2: bcd_to_7seg
        port map (
            BCD_IN     => bcd_hundreds,      -- BCD input for hundreds
            SHUTDOWNn  => RSTn,              -- Shutdown signal (active low)
            D_OUT      => hex_display2       -- 7-segment display output for hundreds
        );

    seg3: bcd_to_7seg
        port map (
            BCD_IN     => bcd_thousands,     -- BCD input for thousands
            SHUTDOWNn  => RSTn,              -- Shutdown signal (active low)
            D_OUT      => hex_display3       -- 7-segment display output for thousands
        );

    -- Synchronize Result Ready Signal
    diff_inst: sync_diff
        generic map (
            G_DERIVATE_RISING_EDGE  => true,  -- Derivative on rising edge
            G_SIG_IN_INIT_VALUE     => '0',   -- Initial value of signal input
            G_RESET_ACTIVE_VALUE    => '0'    -- Reset active value
        )
        port map (
            CLK     => CLK,         -- System clock
            RST     => RSTn,        -- Reset signal
            SIG_IN  => result_ready, -- Signal input
            SIG_OUT => signal_out   -- Signal output
        );

    -- Process for handling the display and LED output assignments
    process(CLK, RSTn)
    begin
        if RSTn = '0' then
            -- Reset condition: disable all outputs
            HEX0 <= (others => '1');  -- Turn off display (all segments high)
            HEX1 <= (others => '1');
            HEX2 <= (others => '1');
            HEX3 <= (others => '1');
            LEDS_1E4 <= (others => '0');
            LED_SIGN <= '0';
        elsif rising_edge(CLK) then
            if result_ready = '1' then
                -- Assign display values when the result is ready
                HEX0 <= hex_display0;
                HEX1 <= hex_display1;
                HEX2 <= hex_display2;
                HEX3 <= hex_display3;
                LEDS_1E4 <= bcd_ten_thousands;
                LED_SIGN <= is_negative; -- Show the sign based on is_negative signal
            else
                -- Default to turning off the display if the result is not ready
                HEX0 <= (others => '1');
                HEX1 <= (others => '1');
                HEX2 <= (others => '1');
                HEX3 <= (others => '1');
                LEDS_1E4 <= (others => '0');
                LED_SIGN <= '0';
            end if;
        end if;
    end process;

    -- Assign LED status indicators
    LEDG(3) <= result_ready;     -- LED indicating the result is ready
    LEDG(2) <= matrices_loaded;  -- LED indicating matrices are loaded
    LEDG(1) <= '1';              -- Always on LED (indicates system is operational)

end architecture structured;
```

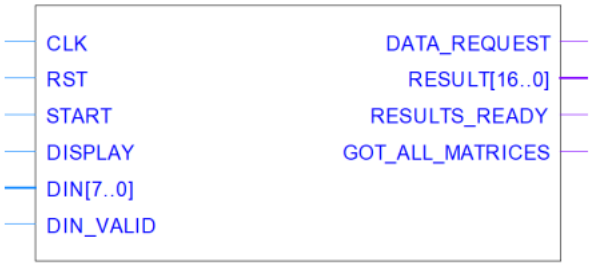
: Pin Planner 3.3

Top View - Wire Bond
Cyclone V - 5CGXFC5C6F27C7



CLK	Input	PIN_R20	5B	B5B_N0	PIN_R20	3.3-...VTTL		16mA...ult)	
DISPLAYn	Input	PIN_Y15	4A	B4A_N0	PIN_Y15	1.2 V		8mA ...ult)	
HEX0[6]	Output	PIN_Y18	4A	B4A_N0	PIN_Y18	1.2 V		8mA ...ult)	1 (default)
HEX0[5]	Output	PIN_Y19	4A	B4A_N0	PIN_Y19	1.2 V		8mA ...ult)	1 (default)
HEX0[4]	Output	PIN_Y20	4A	B4A_N0	PIN_Y20	1.2 V		8mA ...ult)	1 (default)
HEX0[3]	Output	PIN_W18	4A	B4A_N0	PIN_W18	1.2 V		8mA ...ult)	1 (default)
HEX0[2]	Output	PIN_V17	4A	B4A_N0	PIN_V17	1.2 V		8mA ...ult)	1 (default)
HEX0[1]	Output	PIN_V18	4A	B4A_N0	PIN_V18	1.2 V		8mA ...ult)	1 (default)
HEX0[0]	Output	PIN_V19	4A	B4A_N0	PIN_V19	1.2 V		8mA ...ult)	1 (default)
HEX1[6]	Output	PIN_AF24	4A	B4A_N0	PIN_AF24	1.2 V		8mA ...ult)	1 (default)
HEX1[5]	Output	PIN_AC19	4A	B4A_N0	PIN_AC19	1.2 V		8mA ...ult)	1 (default)
HEX1[4]	Output	PIN_AE25	4A	B4A_N0	PIN_AE25	1.2 V		8mA ...ult)	1 (default)
HEX1[3]	Output	PIN_AE26	4A	B4A_N0	PIN_AE26	1.2 V		8mA ...ult)	1 (default)
HEX1[2]	Output	PIN_AB19	4A	B4A_N0	PIN_AB19	1.2 V		8mA ...ult)	1 (default)
HEX1[1]	Output	PIN_AD26	4A	B4A_N0	PIN_AD26	1.2 V		8mA ...ult)	1 (default)
HEX1[0]	Output	PIN_AA18	4A	B4A_N0	PIN_AA18	1.2 V		8mA ...ult)	1 (default)
HEX2[6]	Output	PIN_W20	5A	B5A_N0	PIN_W20	3.3-...VTTL		16mA...ult)	1 (default)
HEX2[5]	Output	PIN_W21	5A	B5A_N0	PIN_W21	3.3-...VTTL		16mA...ult)	1 (default)
HEX2[4]	Output	PIN_V20	5A	B5A_N0	PIN_V20	3.3-...VTTL		16mA...ult)	1 (default)
HEX2[3]	Output	PIN_V22	5A	B5A_N0	PIN_V22	3.3-...VTTL		16mA...ult)	1 (default)
HEX2[2]	Output	PIN_U20	5A	B5A_N0	PIN_U20	3.3-...VTTL		16mA...ult)	1 (default)
HEX2[1]	Output	PIN_AD6	3A	B3A_N0	PIN_AD6	3.3-...VTTL		16mA...ult)	1 (default)
HEX2[0]	Output	PIN_AD7	3A	B3A_N0	PIN_AD7	3.3-...VTTL		16mA...ult)	1 (default)
HEX3[6]	Output	PIN_AC22	5A	B5A_N0	PIN_AC22	3.3-...VTTL		16mA...ult)	1 (default)
HEX3[5]	Output	PIN_AC23	5A	B5A_N0	PIN_AC23	3.3-...VTTL		16mA...ult)	1 (default)
HEX3[4]	Output	PIN_AC24	5A	B5A_N0	PIN_AC24	3.3-...VTTL		16mA...ult)	1 (default)
HEX3[3]	Output	PIN_AA22	5A	B5A_N0	PIN_AA22	3.3-...VTTL		16mA...ult)	1 (default)
HEX3[2]	Output	PIN_AA23	5A	B5A_N0	PIN_AA23	3.3-...VTTL		16mA...ult)	1 (default)
HEX3[1]	Output	PIN_Y23	5A	B5A_N0	PIN_Y23	3.3-...VTTL		16mA...ult)	1 (default)
HEX3[0]	Output	PIN_Y24	5A	B5A_N0	PIN_Y24	3.3-...VTTL		16mA...ult)	1 (default)
LED_SIGN	Output	PIN_H9	8A	B8A_N0	PIN_H9	2.5 V		12mA...ult)	1 (default)
LEDG[3]	Output	PIN_E9	8A	B8A_N0	PIN_E9	2.5 V		12mA...ult)	1 (default)
LEDG[2]	Output	PIN_D8	8A	B8A_N0	PIN_D8	2.5 V		12mA...ult)	1 (default)
LEDG[1]	Output	PIN_K6	8A	B8A_N0	PIN_K6	2.5 V		12mA...ult)	1 (default)
LEDS_1E4[3]	Output	PIN_J10	8A	B8A_N0	PIN_J10	2.5 V		12mA...ult)	1 (default)
LEDS_1E4[2]	Output	PIN_H7	8A	B8A_N0	PIN_H7	2.5 V		12mA...ult)	1 (default)
LEDS_1E4[1]	Output	PIN_K8	8A	B8A_N0	PIN_K8	2.5 V		12mA...ult)	1 (default)
LEDS_1E4[0]	Output	PIN_K10	8A	B8A_N0	PIN_K10	2.5 V		12mA...ult)	1 (default)
RSTn	Input	PIN_P11	3B	B3B_N0	PIN_P11	1.2 V		8mA ...ult)	
STARTn	Input	PIN_P12	3B	B3B_N0	PIN_P12	1.2 V		8mA ...ult)	

4. מימוש הבלוק main_controller:



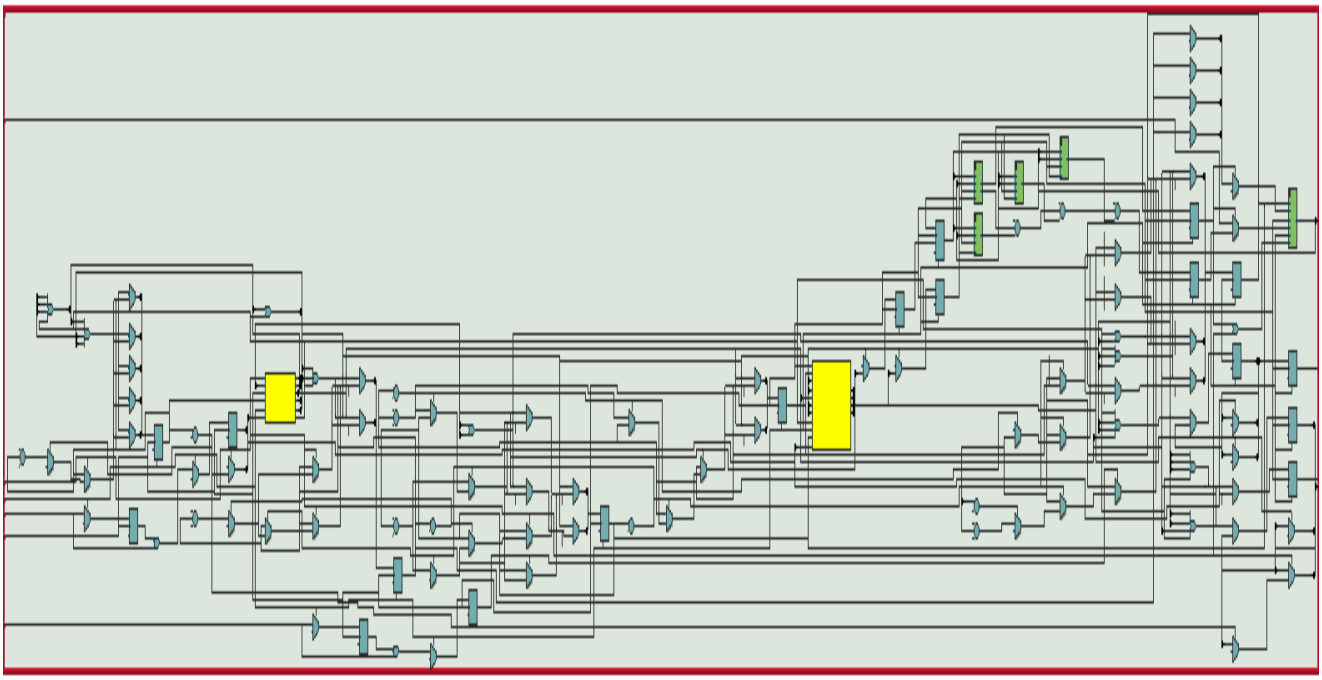
הבלוק main_controller הוא רכיב הבקרה המרכזי של המערכת, והוא אחראי על ניהול כל תהליך כפל המטריצות והצגת התוצאות על תצוגות 7-segment ו-LEDs. הבלוק מקבל את הפקודות מהמשתמש באמצעות לחצנים (START, DISPLAY) לאחר שעוברים תהליך של סנכרון וזיהוי עליות אותות, ופועל בהתאם לפקודות שניתנו,הבלוק שולח פולס של DATA_REQUEST לבלוק data_generator כדי לבקש את נתוני המטריצות. לאחר מכן, הבלוק מתחיל לקלוט את המידע ולהעבירו לזיכרון פנימי לשימוש בחישובים.טוען שתי מטריצות בזיכרון, אחת אחרי השנייה. במהלך שלב זה, הבלוק מעדכן מונים ואינדקסים כדי לעקוב אחרי מיקומי המידע בזיכרון ולוודא שכל הנתונים נטענים בצורה נכונה. עם סיום טעינת המטריצות, הבלוק מאותת באמצעות GOT_ALL_MATRICES שהמטריצות נטענו בהצלחה.

הבלוק מנהל את פעולת רכיבי הכפל (my_multiplier) ומוודא שכל תוצאת כפל נשמרת בצורה נכונה בזיכרון

על תצוגות 7-segment והפעלת נורות LED חיווי בהתאם למצב הנוכחי של התוצאה המחושבת. במידה ותוצאה שנבדקת נמצאה כלא תקינה, תידלק נורת חיווי אדומה (LED_SIGN).

כאשר כל המידע מוצג והתהליך מסתיים, לחיצה נוספת על לחצן ההתחלה (START) מאפסת את המערכת ומכבה את כל נורות החיווי (לדים ירוקים ואדום), ומחזירה את המערכת למצב המתנה (IDLE).

4.1 מבנה לוגי :



4.2 קובץ main_controller.vhd :

```vhdl
-- Import necessary IEEE libraries
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- Entity declaration for the main_controller module
entity main_controller is
    port (
        RST              : in    std_logic; -- Reset signal
        CLK              : in    std_logic; -- System clock signal
        START            : in    std_logic; -- Start signal
        DISPLAY          : in    std_logic; -- Display signal
        DIN              : in    std_logic_vector(7 downto 0); -- Data input
        DIN_VALID        : in    std_logic;  -- Data input valid signal
        RESULTS_READY    : out   std_logic; -- Signal indicating that the results are ready
        DATA_REQUEST     : out   std_logic;  -- Signal requesting more data
        RESULT           : out   std_logic_vector(16 downto 0); -- Final result output
        GOT_ALL_MATRICES : out   std_logic -- Signal indicating that all matrices have been loaded
    );
end entity main_controller;

-- Architecture definition for the main_controller module
architecture behave of main_controller is

   -- Constants for internal use
    constant N              : integer := 8; -- Width of the data for multipliers
    constant NUM_ELEMENTS   : integer := 16; -- Number of elements in each matrix
    constant LATENCY        : integer := 2; -- Latency for the multipliers
    constant SIGNED_MODE    : boolean := true; -- Indicates whether the multipliers should use signed data

    -- State Definitions for the finite state machines (FSM)
    type main_state_type is (IDLE, LOAD_FIRST_MATRIX, LOAD_SECOND_MATRIX, WAIT_CALC, CALCULATE, DISPLAY_RESULTS);
    type calc_state_type is (INIT, FETCH_ROW, FETCH_COL, SAVE_RESULT);

    -- Component definition for the my_multiplier used for matrix multiplication
    component my_multiplier is
        generic (
            N          : integer := 8; -- Data width
            LATENCY    : integer range 1 to 8 := 1;  -- Latency of the multiplier
            IS_SIGNED  : boolean := false  -- Whether the multiplier is signed (+/-)
        );
        port (
            CLK        : in  std_logic; -- Clock signal
            DIN_VALID  : in  std_logic; -- Data valid signal
            A          : in  std_logic_vector(N-1 downto 0); -- Matrix A & Matrix B
            B          : in  std_logic_vector(N-1 downto 0);
            Q          : out std_logic_vector(N*2-1 downto 0); -- Multiplication result
            DOUT_VALID : out std_logic
        );
    end component;

    -- Component definition for the matrix_ram used for storing matrix data
    component matrix_ram is
        generic (
            DATA_WIDTH    : integer := 8; -- Data width for memory
            ADDRESS_BITS  : integer := 6 -- Number of address bits for memory
        );
        port (
            RST       : in  std_logic; -- Reset signal
```

```vhdl
            CLK       : in  std_logic; -- Clock signal
            WREN      : in  std_logic; -- Write enable signal
            DATA      : in  std_logic_vector(DATA_WIDTH-1 downto 0);  -- Data input
            BYTEENA   : in  std_logic_vector((DATA_WIDTH/8)-1 downto 0); -- Byte enable input
            ADDRESS   : in  std_logic_vector(ADDRESS_BITS-1 downto 0); -- Memory address input
            Q         : out std_logic_vector(DATA_WIDTH-1 downto 0)     -- Data output
        );
    end component;
    -- Internal signals
    signal main_state       : main_state_type := IDLE;  -- Main FSM state
    signal calc_state       : calc_state_type := INIT; -- Calculation FSM state
    signal data_counter     : integer range 0 to NUM_ELEMENTS; -- Element counter for data
    signal row_index        : integer range 0 to 3 ; -- Row index for matrix operations
    signal col_index        : integer range 0 to 3; -- Column index for matrix operations
    signal byte_enables     : std_logic_vector(3 downto 0)  ;
    signal memory_enable    : std_logic_vector(3 downto 0)  ;
    signal memory_address   : std_logic_vector(4 downto 0)  ;
    signal memory_data_in   : std_logic_vector(31 downto 0);
    signal memory_write     : std_logic  ; -- Memory write signal
    signal high_address     : std_logic_vector(2 downto 0) := "000"; -- High address signal for matrix memory
    signal iteration_count  : integer range 0 to 3 := 0; -- Iteration counter for matrix calculations
    signal save_mat1_row    : std_logic := '0'; -- Signal for controlling the saving of the first matrix row
    signal save_mat2_col    : std_logic := '0'; -- Signal for controlling the saving of the second matrix column
    signal mat1_row_content : std_logic_vector(31 downto 0)  ; -- Content of a row from the first matrix
    signal memory_output    : std_logic_vector(31 downto 0)  ;
    signal mult1_result     : std_logic_vector(15 downto 0);
    signal mult2_result     : std_logic_vector(15 downto 0)  ;
    signal mult3_result     : std_logic_vector(15 downto 0)  ;
    signal mult4_result     : std_logic_vector(15 downto 0)  ; -- Fourth multiplier result
    signal mult1_valid      : std_logic; -- Valid signal for the first multiplier
    signal mult2_valid      : std_logic ; -- Valid signal for the second multiplier
    signal mult3_valid      : std_logic ; -- Valid signal for the third multiplier
    signal mult4_valid      : std_logic ; -- Valid signal for the fourth multiplier
    signal matrix_result_valid : std_logic := '0'; -- Indicates that the matrix result is valid
    signal matrix_result       : std_logic_vector(31 downto 0) := (others => '0'); -- Final matrix result
    signal result_available    : std_logic := '0'; -- Indicates that the result is available
    signal start_edge         : std_logic; -- Rising edge detection for the START signal
    signal start_prev         : std_logic := '0'; -- Previous state of the START signal
    signal display_edge       : std_logic; -- Rising edge detection for the DISPLAY signal
    signal display_prev       : std_logic := '0'; -- Previous state of the DISPLAY signal
begin
-- Process to detect rising edges of START and DISPLAY signals (מחליף לזיהוי עליות באותות START ו-DISPLAY)
process(CLK)
    variable display_latched : std_logic := '0'; -- Variable to track the previous state of DISPLAY
    variable start_latched   : std_logic := '0'; -- Variable to track the previous state of START
begin
    if rising_edge(CLK) then
        if RST = '1' then  -- Active high asynchronous Reset
            -- Reset both edge detection signals and previous state signals
            display_edge <= '0';
            display_prev <= '0';
            display_latched := '0';

            start_edge <= '0';
            start_prev <= '0';
            start_latched := '0';
        else
            -- Edge detection for DISPLAY signal
            if DISPLAY = '1' and display_latched = '0' then
```

```
                display_edge <= '1'; -- Rising edge detected for DISPLAY
            else
                display_edge <= '0';
            end if;
            display_latched := DISPLAY;  -- Update the latched value
            display_prev <= display_latched;  -- Maintain previous state tracking

            -- Edge detection for START signal
            if START = '1' and start_latched = '0' then
                start_edge <= '1'; -- Rising edge detected for START
            else
                start_edge <= '0';
            end if;
            start_latched := START;  -- Update the latched value
            start_prev <= start_latched;  -- Maintain previous state tracking
        end if;
    end if;
end process;

    -- Main FSM Process to control the state of the main_controller
    process(CLK, RST)
    begin
        if RST = '1' then
            main_state <= IDLE;
            calc_state <= INIT;
            DATA_REQUEST <= '0';
            data_counter <= 0;
            row_index <= 0;
            col_index <= 0;
            high_address <= "000";
            iteration_count <= 0;
            save_mat1_row <= '0';
            save_mat2_col <= '0';
            result_available <= '0';
            GOT_ALL_MATRICES <= '0';
            RESULTS_READY <= '0';
        elsif rising_edge(CLK) then
            -- Default signal assignments
            DATA_REQUEST <= '0';
            save_mat1_row <= '0';
            save_mat2_col <= '0';

            -- Main FSM state transitions
            case main_state is
                when IDLE =>
                    if start_edge = '1' then
                        main_state <= LOAD_FIRST_MATRIX;
                        DATA_REQUEST <= '1';
                    end if;

                    -- Reset indices and counters
                    row_index <= 0;
                    col_index <= 0;
                    iteration_count <= 0;

                when LOAD_FIRST_MATRIX =>
                    if DIN_VALID = '1' then
                        if data_counter = NUM_ELEMENTS - 1 then
                            main_state <= LOAD_SECOND_MATRIX;
```

```
                            main_state <= LOAD_SECOND_MATRIX;
                            data_counter <= 0;
                            DATA_REQUEST <= '1';
                        else
                            data_counter <= data_counter + 1;
                        end if;


                        if col_index = 3 then
                            col_index <= 0;
                            if row_index = 3 then
                                row_index <= 0;
                            else
                                row_index <= row_index + 1;
                            end if;
                        else
                            col_index <= col_index + 1;
                        end if;
                    end if;
                    high_address <= "000";

                when LOAD_SECOND_MATRIX =>
                    if DIN_VALID = '1' then
                        if data_counter = NUM_ELEMENTS - 1 then
                            main_state <= WAIT_CALC;
                            GOT_ALL_MATRICES <= '1';
                            data_counter <= 0;
                        else
                            data_counter <= data_counter + 1;
                        end if;

                        -- Update row and column indices
                        if row_index = 3 then
                            row_index <= 0;
                            if col_index = 3 then
                                col_index <= 0;
                            else
                                col_index <= col_index + 1;
                            end if;
                        else
                            row_index <= row_index + 1;
                        end if;
                    end if;
                    high_address <= "001";

                when WAIT_CALC =>
                    -- Prepare for calculation
                    row_index <= 0;
                    high_address <= "000";
                    if start_edge = '1' then
                        main_state <= CALCULATE;
                        calc_state <= FETCH_ROW;
                    end if;

                when CALCULATE =>
                    -- Calculation FSM state transitions
                    case calc_state is
                        when FETCH_ROW =>
                            calc_state <= FETCH_COL;
```

```
                    calc_state <= FETCH_COL;
                    high_address <= "001";
                    row_index <= 0;
                    save_mat1_row <= '1';

                when FETCH_COL =>
                    if row_index = 3 then
                        calc_state <= SAVE_RESULT;
                        row_index <= 0;
                        high_address <= '1' & std_logic_vector(to_unsigned(iteration_count, 2));
                    else
                        row_index <= row_index + 1;
                    end if;
                    save_mat2_col <= '1';

                when SAVE_RESULT =>
                    if row_index = 3 then
                        if iteration_count = 3 then
                            calc_state <= INIT;
                        else
                            iteration_count <= iteration_count + 1;
                            calc_state <= FETCH_ROW;
                            high_address <= "000";
                            row_index <= iteration_count + 1;
                        end if;
                    else
                        row_index <= row_index + 1;
                    end if;

                when INIT =>
                    main_state <= DISPLAY_RESULTS;
                    result_available <= '1';
                    GOT_ALL_MATRICES <= '0';
                    row_index <= 0;
                    high_address <= "100";
            end case;

        when DISPLAY_RESULTS =>
            -- Display the results
            if display_edge = '1' then
                if row_index = 3 then
                    if unsigned(high_address(1 downto 0)) = 3 then
                        high_address <= "100";
                    else
                        high_address <= std_logic_vector(unsigned(high_address) + 1);
                    end if;
                    row_index <= 0;
                else
                    row_index <= row_index + 1;
                end if;
            end if;
            if start_edge = '1' then
                main_state <= IDLE;
                result_available <= '0';
            end if;

    end case;

    -- Update the RESULTS_READY signal based on the result availability
```

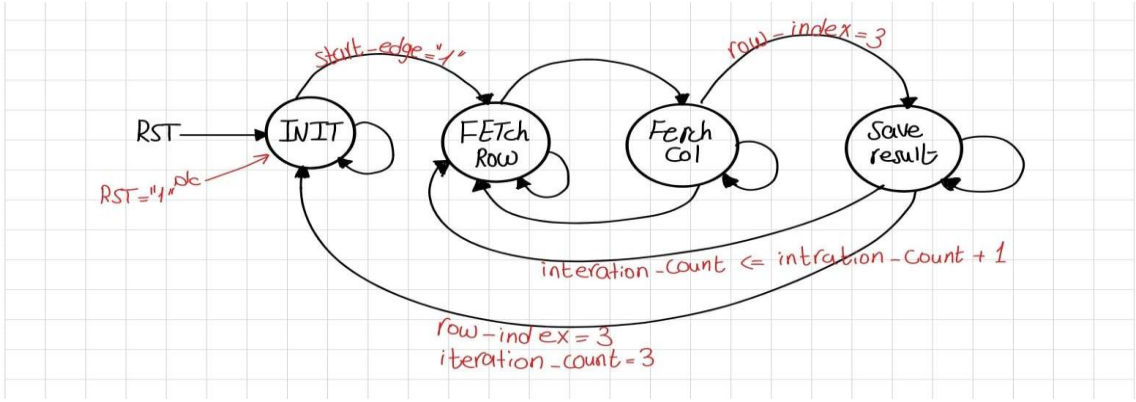
```
    -- Update the RESULTS_READY signal based on the result availability
    RESULTS_READY <= result_available;

end if;
end process;

-- Update the RESULTS_READY signal based on the result availability
process(CLK, RST)
begin
    if RST = '1' then
        mat1_row_content <= (others => '0');
    elsif rising_edge(CLK) then
        if save_mat1_row = '1' then
            mat1_row_content <= memory_output;
        end if;
    end if;
end process;

-- Address and Memory Control
memory_address <= high_address & std_logic_vector(to_unsigned(row_index, 2));
memory_data_in <= matrix_result when matrix_result_valid = '1' else DIN & DIN & DIN & DIN;
memory_write <= DIN_VALID or matrix_result_valid;
memory_enable <= "1111" when matrix_result_valid = '1' else byte_enables;

-- Byte Enable Logic
byte_enable_process : process(col_index)
begin
    case col_index is
        when 0 => byte_enables <= "0001";
        when 1 => byte_enables <= "0010";
        when 2 => byte_enables <= "0100";
        when 3 => byte_enables <= "1000";
        when others => byte_enables <= (others => '0');
    end case;
end process;

-- Instantiate Matrix RAM
matrix_ram_inst : matrix_ram
    generic map (
        DATA_WIDTH => 32,
        ADDRESS_BITS => 5
    )
    port map (
        RST       => RST,
        CLK       => CLK,
        DATA      => memory_data_in,
        BYTEENA   => memory_enable,
        WREN      => memory_write,
        ADDRESS   => memory_address,
        Q         => memory_output
    );

-- Instantiate Multipliers
mult1 : my_multiplier
    generic map (
        N          => N,
        LATENCY    => LATENCY,
        IS_SIGNED  => SIGNED_MODE
    )
```

```
352             port map (
353                 CLK        => CLK,
354                 DIN_VALID  => save_mat2_col,
355                 A          => mat1_row_content(7 downto 0),
356                 B          => memory_output(7 downto 0),
357                 Q          => mult1_result,
358                 DOUT_VALID => mult1_valid
359             );
360
361         mult2 : my_multiplier
362             generic map (
363                 N          => N,
364                 LATENCY    => LATENCY,
365                 IS_SIGNED  => SIGNED_MODE
366             )
367             port map (
368                 CLK        => CLK,
369                 DIN_VALID  => save_mat2_col,
370                 A          => mat1_row_content(15 downto 8),
371                 B          => memory_output(15 downto 8),
372                 Q          => mult2_result,
373                 DOUT_VALID => mult2_valid
374             );
375
376         mult3 : my_multiplier
377             generic map (
378                 N          => N,
379                 LATENCY    => LATENCY,
380                 IS_SIGNED  => SIGNED_MODE
381             )
382             port map (
383                 CLK        => CLK,
384                 DIN_VALID  => save_mat2_col,
385                 A          => mat1_row_content(23 downto 16),
386                 B          => memory_output(23 downto 16),
387                 Q          => mult3_result,
388                 DOUT_VALID => mult3_valid
389             );
390
391         mult4 : my_multiplier
392             generic map (
393                 N          => N,
394                 LATENCY    => LATENCY,
395                 IS_SIGNED  => SIGNED_MODE
396             )
397             port map (
398                 CLK        => CLK,
399                 DIN_VALID  => save_mat2_col,
400                 A          => mat1_row_content(31 downto 24),
401                 B          => memory_output(31 downto 24),
402                 Q          => mult4_result,
403                 DOUT_VALID => mult4_valid
404             );
405
406         -- Summation of Products Process
407         product_summation : process(CLK, RST)
408         begin
409             if RST = '1' then
410                 matrix result <= (others => '0');
```

```
410                 matrix_result <= (others => '0');
411                 matrix_result_valid <= '0';
412             elsif rising_edge(CLK) then
413                 matrix_result <= std_logic_vector(
414                     resize(signed(mult1_result), 32) +
415                     resize(signed(mult2_result), 32) +
416                     resize(signed(mult3_result), 32) +
417                     resize(signed(mult4_result), 32)
418                 );
419                 matrix_result_valid <= mult1_valid;
420             end if;
421         end process;
422
423         -- Assign the final RESULT output
424         RESULT <= memory_output(16 downto 0);
425
426     end architecture behave;
```

4.3 מכונות מצבים :

מבנה לוגי 1 :



המערכת מתחילה במצב INIT לאחר שהיא מקבלת פקודת reset .

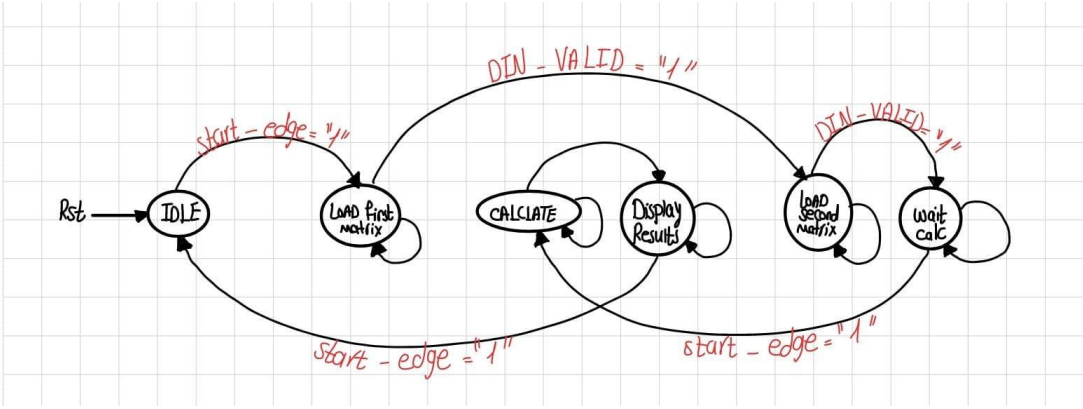
אם start_edge=`1` המכונה מעבירה ל FETCH_ROW-כדי לאסוף שורה מהמטריצה הראשונה

לאחר איסוף שורה, המכונה מעבירה ל FETCH_COL-כדי לאסוף עמודה מהמטריצה השנייה

לאחר איסוף העמודה, אם כל השורות הוקראו המכונה מעבירה ל SAVE_RESULT- שבו שומרים את התוצאות שהתקבלו מהכפל.אם לא כל השורות הוקראו, המכונה ממשיכה לאסוף את שאר השורות.

אם row_index=3 ו intration_count=3 וכל האיטרציות הושלמו המכונה חוזרת ל INIT לשלב הבא של חישוב.

מבנה לוגי 2 :

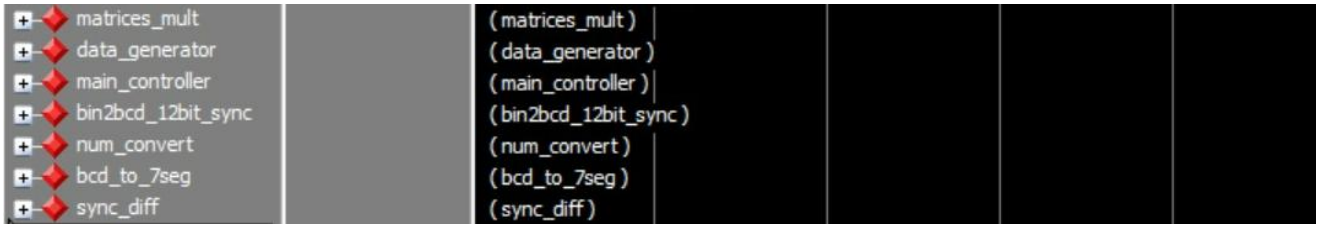


המערכת מתחילה במצב IDLE לאחר שהיא מקבלת פקודת RST.

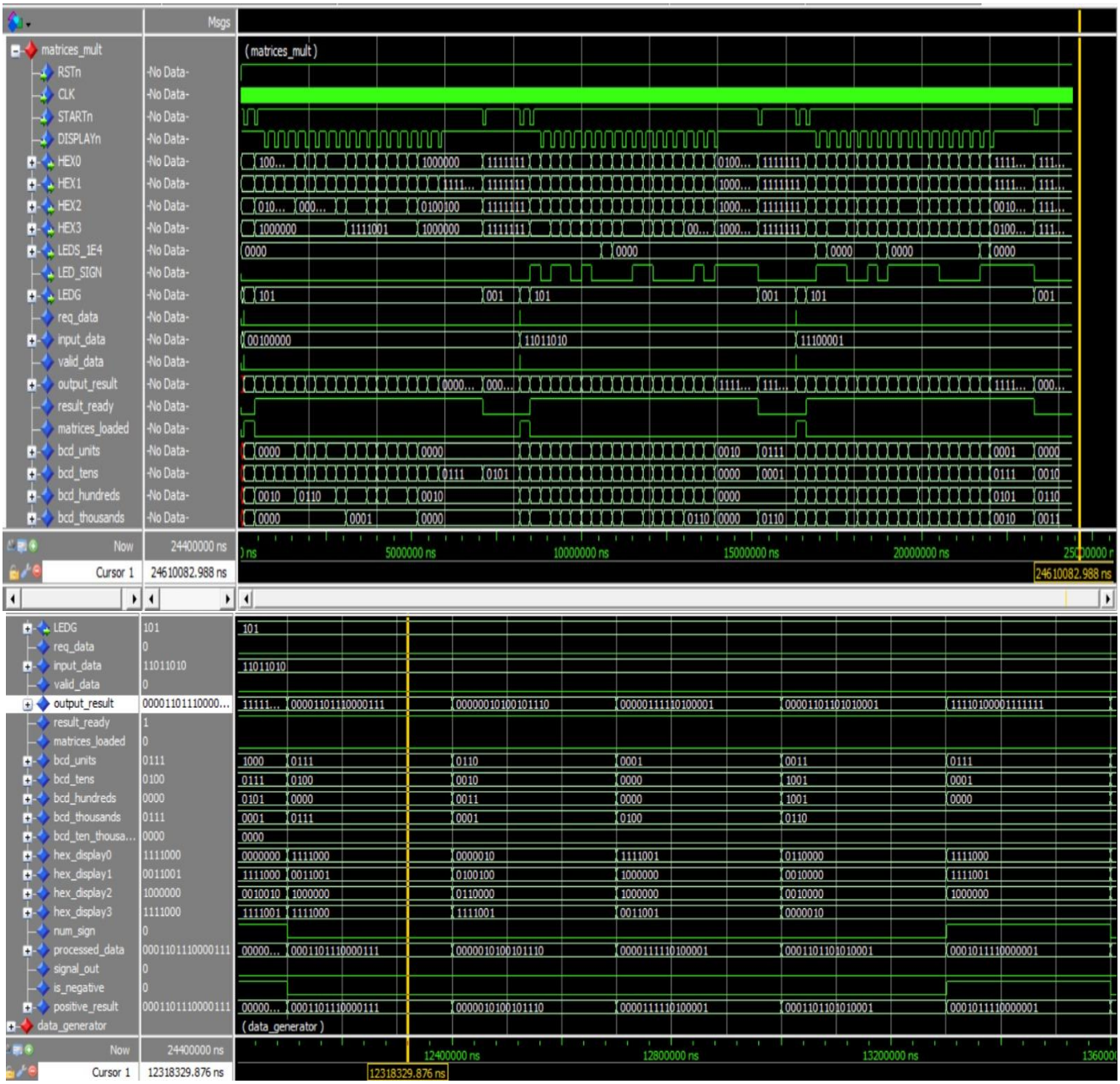
אם start_edge=`1` המכונה עוברת ל LOAD_FIRST_MATRIX- במצב זה, המערכת טוענת את המטריצה הראשונה כאשר המטריצה הראשונה נטענה בהצלחה אם DIN_VALED=`1` המכונה מעבירה ל.LOAD_SECOND_MATRIX- לטעון את המטריצה השנייה לאחר טעינת המטריצה השנייה, המערכת יכולה להמתין להמשך חישוב במצב AWAIT_CALC. אם start_edge=`1` המערכת עוברת למצב CALCULATE כדי לבצע את החישובים הנדרשים. אם החישוב הושלם ואין צורך במטריצה נוספת, המערכת עוברת למצב DISPLAY_RESULTS להצגת התוצאות.

לאחר הצגת התוצאות, המערכת יכולה לחזור למצב IDLE ולהמתין לתהליך חדש או לקבל פקודה נוספת.

5. תוצאות viewer netlist RTL עבור כל בלוק והרמה העליונה :



עבור הרמה העליונה matrices_mult:



ניתן לראות כי בתחילת הסימולציה, ניתן לראות שהאות RSTn שווה לאפס מה שאומר שהמערכת נמצאת במצב איפוס.

לאחר זמן, RSTn הופך להיות שווה ל 1 המערכת יוצאת ממצב איפוס ומתחילה לפעול.

ברגע שכל הנתונים נטענים (matrices_loaded = 1), המערכת מתחילה את חישוב כפל המטריצות.

ניתן לראות שעם הזמן, output_result מתחיל להשתנות ומציג את התוצאה של כפל המטריצות.

כאשר result_ready הופך ל-1, התוצאה מוכנה להצגה.

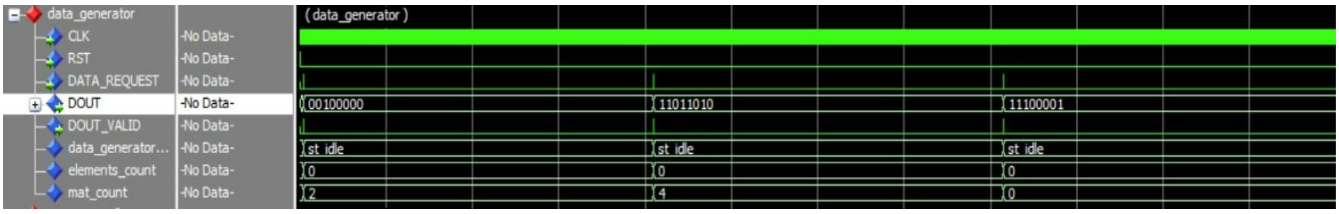
התוצאות מופיעות על תצוגות 7-segment (HEX0 - HEX3) וה-LEDs בהתאם לערכים המחושבים (bcd_units, bcd_tens, וכו').

LED_SIGN מציין את הסימן של התוצאה.

LEDS_1E4 מציג את הערך של ספרת עשרת אלפים אם נדרש.

LEDG מדליקים את נורות ה-LED המתאימות המציינות שהמערכת פועלת והתוצאה מוכנה.

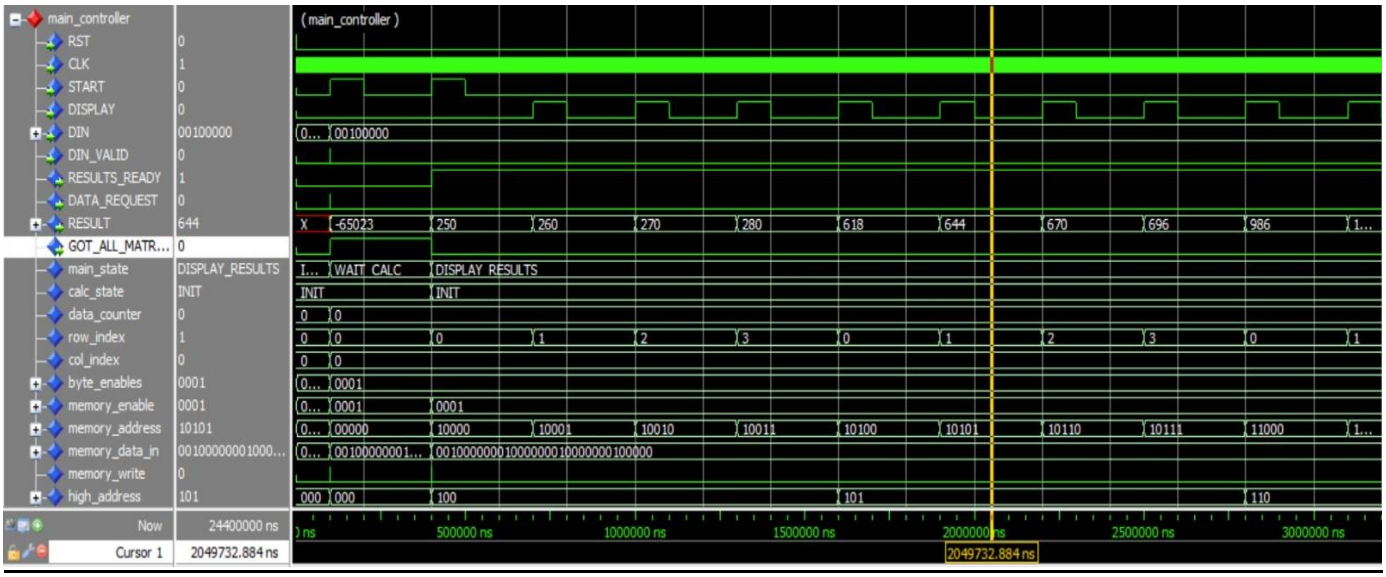
עבור data_generator :

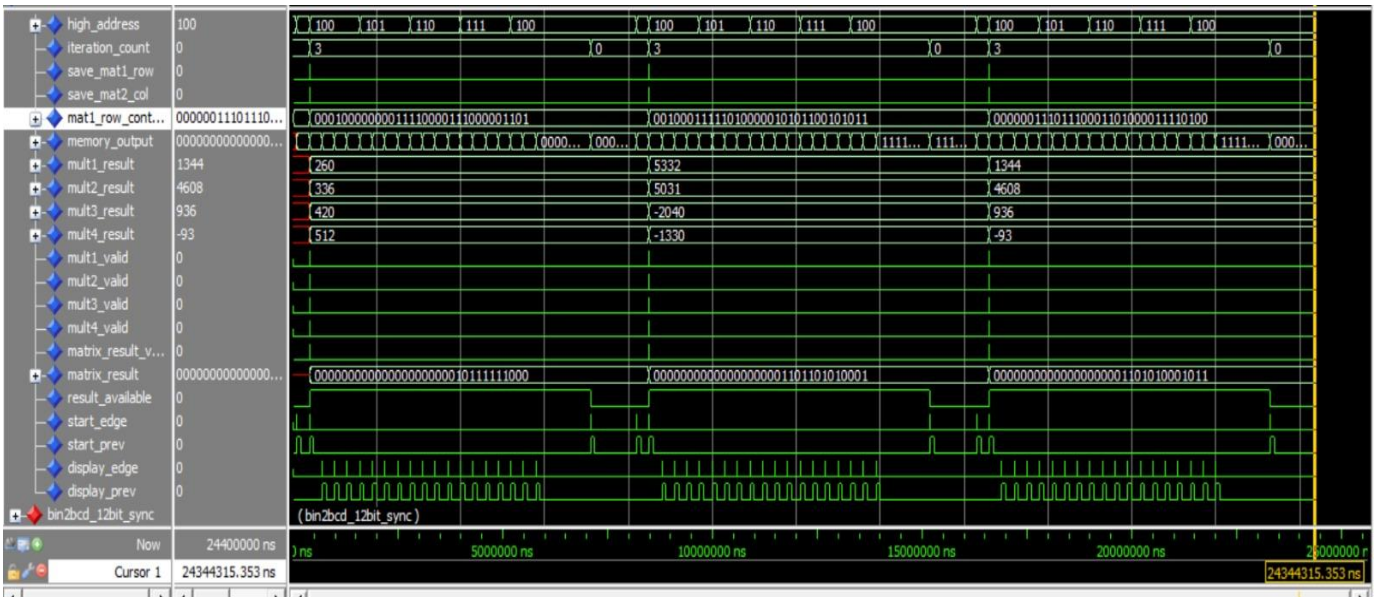


עבור synic_diff:

ניתן לראות מסימולציה הפלט מציג את השינויים באות SIG_IN לאחר הסנכרון והגזירה. וניתן לראות שהוא מפיק פלט מסונכרן SIG_OUT רק כשהוא מזהה שינוי מתאים ב-SIG_IN.

עבור main_controller:





אפשר לראות RST מאופס לאחר פרק זמן קצר, והמודול ממשיך לפעול לפי אות השעון CLK ,

START ו-DISPLAY משפיעות על מצבי מכונת המצבים.

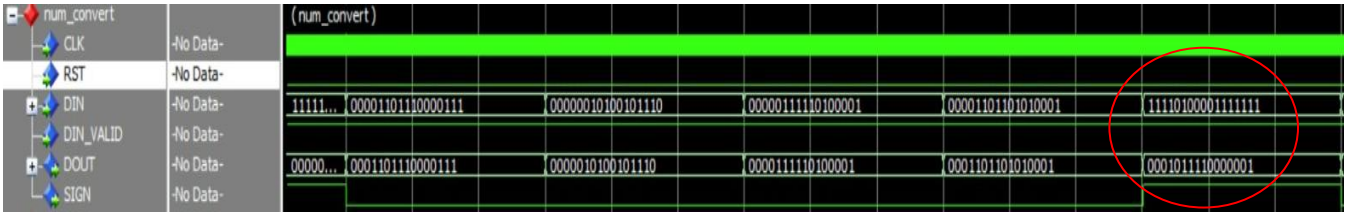
תוצאת החישוב מוצגת באמצעות אות ה-RESULT

מצביע על כך שכל הנתונים הנדרשים לחישוב נטענו והמערכת מוכנה להמשיך לחישוב התוצאה הסופית.

עבור bin2bcd_12bit:

ניתן לראות מהסימולציה שהוא מממיר המספר הבינארי 0000000011111010 שזה שווה ל 250 בבסיס עשרוני ומפריד אותו לתוך הספרות אחדות עשרות מאות אלפים עשרות האלפים כך ש באחדות מקבלים 0000 (0 בבסיס עשרוני), עשרות מקבלים 0101 (5 בבסיס עשרוני) מאות 0010 (2 בבסיס עשרוני) והשאר ממלא אפס .

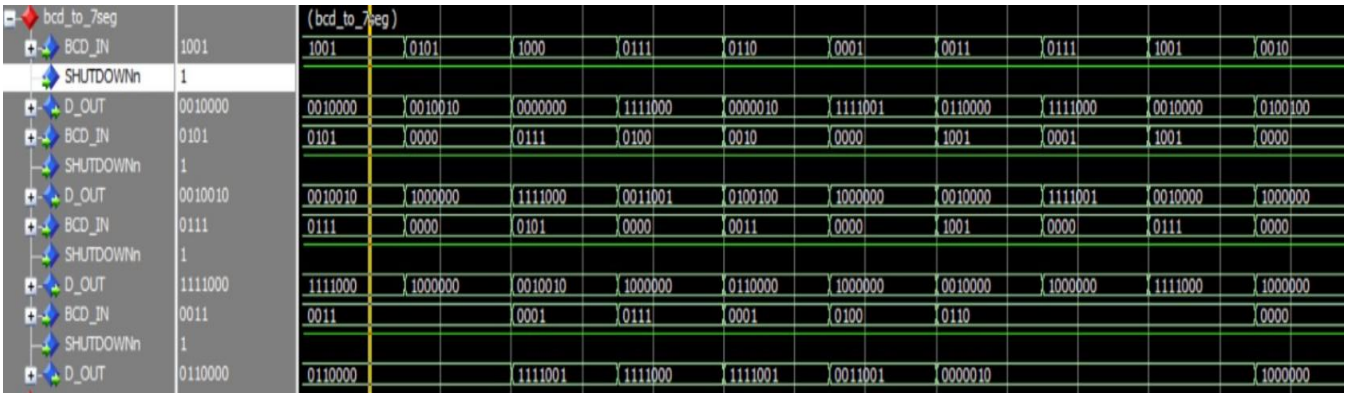
עבור num_convert:



ניתן לראות מהסימולציה שהמספר החיובי נשאר אותו דבר ללא שינוי. מתוך העיגול האדום נזהה שהמספר 11110100001111111 זהו מספר שלילי בגלל ביט הסימן (1)

נהפוך את כל הביטים ונקבל 0001011110000000 נוסיף 1 למספר המהופך 0001011110000000, התוצאה היא 0001011110000001 מאחר וזה היה מספר שלילי מקבלים DOUT התוצאה המוחלטת היא 000101111000001 שמייצג את הערך המוחלט של 6017 בעשרוני.

עבור bcd_to_7seg:



ניתן לראות בסימולציה כל ערך שנכנס למערכת מתורגם לערך שמסמל איזה לידים צריכים להידלק בצג שעל הערכה (הלדים נדלקים בנמוך).המוצא תלוי ב- SHUTDOWN, כלומר כאשר הוא אפס המצוא יהיה '1111111'זאת אומרת ה- 7 segment של הערכה יהיה כבוי, וכאשר '1' = SHUTDOWN המערכת תעבוד בדיוק כמו בסימולציה.

6. ניצול משאבים :

ניצול משאבים עבור כל התוכנית :

- Flow Summary
- Flow Settings
- Flow Non-Default C
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthes
- Fitter
- Assembler
- Timing Analyzer
- EDA Netlist Writer
- Flow Messages
- Flow Suppressed M

<<Filter>>

Flow Status	Successful - Wed Aug 14 15:37:31 2024
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	matrices_mult
Top-level Entity Name	matrices_mult
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	1,128 / 29,080 (4 %)
Total registers	2344
Total pins	40 / 364 (11 %)
Total virtual pins	0
Total block memory bits	5,888 / 4,567,040 (< 1 %)
Total DSP Blocks	4 / 150 (3 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 12 (0 %)
Total DLLs	0 / 4 (0 %)

ניתן לראות ש- LOGIC UTILIZATION = 4% זה בגלל קוץ של LOGIC ANALYZER INTERFACE היה מוגדר, אם מסרים אותו יהיה 2%.

ניצול משאבים לכל בלוק :

Entity:Instance	ALMs needed [=A-B+C]	[A] ALMs used in final placement	[B] Estimate of ALMs recoverable by dense packing	[C] Estimate of ALMs unavailable	ALMs used f
my_multiplier:mult3	4.0 (4.0)	4.5 (4.5)	0.5 (0.5)	0.0 (0.0)	0.0 (0.0)
my_multiplier:mult4	3.3 (3.3)	6.9 (6.9)	3.6 (3.6)	0.0 (0.0)	0.0 (0.0)
num_convert:converter_inst					
data_generator:data_generator_inst	20.7 (20.7)	24.3 (24.3)	3.6 (3.6)	0.0 (0.0)	0.0 (0.0)
sync_diff:diff_inst					
bcd_to_7seg:seg0	4.7 (4.7)	4.7 (4.7)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
bcd_to_7seg:seg1	4.7 (4.7)	4.7 (4.7)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
bcd_to_7seg:seg2	4.7 (4.7)	4.7 (4.7)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
bcd_to_7seg:seg3	4.7 (4.7)	4.7 (4.7)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)

[C] Estimate of ALMs unavailable	ALMs used for memory	Combinational ALUTs	Dedicated Logic Registers	I/O Registers	Block Memory Bits	M10Ks	DSP Blocks	Pins	Virtual Pins	Full Hierarchy Name
1.0 (0.0)	0.0 (0.0)	731 (22)	1287 (33)	0 (0)	0	0	4	40	0	|matrices_mult
0.0 (0.0)	0.0 (0.0)	66 (66)	39 (39)	0 (0)	0	0	0	0	0	|matrices_mult|bin2...
1.0 (0.0)	0.0 (0.0)	582 (95)	1193 (61)	0 (0)	0	0	4	0	0	|matrices_mult|mai...
1.0 (1.0)	0.0 (0.0)	487 (487)	1066 (1066)	0 (0)	0	0	0	0	0	|matrices_mult|mai...
0.0 (0.0)	0.0 (0.0)	0 (0)	18 (18)	0 (0)	0	0	1	0	0	|matrices_mult|mai...
0.0 (0.0)	0.0 (0.0)	0 (0)	16 (16)	0 (0)	0	0	1	0	0	|matrices_mult|mai...
0.0 (0.0)	0.0 (0.0)	0 (0)	16 (16)	0 (0)	0	0	1	0	0	|matrices_mult|mai...
0.0 (0.0)	0.0 (0.0)	0 (0)	16 (16)	0 (0)	0	0	1	0	0	|matrices_mult|mai...

Cyclone V: 5CGXFC5C6F27C7			
matrices_mult	1135.5 (33.7)	1518.5 (34.0)	383.5 (0.3)
sld_hub:auto_hub	63.5 (0.5)	74.5 (0.5)	11.0 (0.0)
sld_signaltap:auto_signaltap_0	346.5 (17.2)	509.0 (45.4)	162.5 (28.2)
bin2bcd_12bit_sync:bcd_converter	52.8 (52.8)	56.5 (56.5)	3.6 (3.6)
main_controller:controller_inst	599.2 (49.6)	800.7 (49.6)	202.0 (0.0)
num_convert:converter_inst			
data_generator:data_generator_inst	21.2 (21.2)	25.2 (25.2)	4.0 (4.0)
sync_diff:diff_inst			
bcd_to_7seg:seg0	4.7 (4.7)	4.7 (4.7)	0.0 (0.0)

[C] Estimate of ALMs unavailable	ALMs used for memory	Combinational ALUTs	Dedicated Logic Registers	I/O Registers	Block Memory Bits	M10Ks
0.5 (0.0)	0.0 (0.0)	1132 (51)	2378 (33)	0 (0)	7296	2
0.0 (0.0)	0.0 (0.0)	90 (1)	98 (0)	0 (0)	0	0
0.0 (0.0)	0.0 (0.0)	280 (2)	1001 (114)	0 (0)	7296	2
0.0 (0.0)	0.0 (0.0)	66 (66)	39 (39)	0 (0)	0	0
0.5 (0.0)	0.0 (0.0)	584 (97)	1186 (60)	0 (0)	0	0
0.0 (0.0)	0.0 (0.0)	33 (33)	21 (21)	0 (0)	0	0
0.0 (0.0)	0.0 (0.0)	7 (7)	0 (0)	0 (0)	0	0

mory	Combinational ALUTs	Dedicated Logic Registers	I/O Registers	Block Memory Bits	M10Ks	DSP Blocks	Pins	Virtual Pins	Full Hierarchy Name
	1132 (51)	2378 (33)	0 (0)	7296	2	4	40	0	|matrices_mult
	90 (1)	98 (0)	0 (0)	0	0	0	0	0	|matrices_mult|sld_...
	280 (2)	1001 (114)	0 (0)	7296	2	0	0	0	|matrices_mult|sld_...
	66 (66)	39 (39)	0 (0)	0	0	0	0	0	|matrices_mult|bin2...
	584 (97)	1186 (60)	0 (0)	0	0	4	0	0	|matrices_mult|mai...
	33 (33)	21 (21)	0 (0)	0	0	0	0	0	|matrices_mult|data...
	7 (7)	0 (0)	0 (0)	0	0	0	0	0	|matrices_mult|bcd...

7. תוצאות מעניינות של SignalTAB

בדיקה עבור לחיצת DISPLAYn:

Type	Alias	Name	57	57	1 Basic Al
		RSTn			
		STARTn			
		DISPLAYn			
		main_controller:controller_inst|RESULT[16..0]			XXXXXXXXXX
		HEX0[6..0]			XXXXXXXb
		HEX1[6..0]			XXXXXXXb
		HEX2[6..0]			XXXXXXXb
		HEX3[6..0]			XXXXXXXb
		LED_SIGN			
		LEDS_1E4[3..0]			XXXXb
		LEDG[3..1]			XXXb
		...r:controller_inst|main_state.DISPLAY_RESULTS			

7.1 עבור מטריצה A :

Alias	Name	12	13
	RSTn		
	STARTn		
	DISPLAYn		
	main_controller:controller_inst|RESULT[16..0]	260	
	HEX0[6..0]	1000000b	
	HEX1[6..0]	0000010b	
	HEX2[6..0]	0100100b	
	HEX3[6..0]	1000000b	
	LED_SIGN		
	LEDS_1E4[3..0]	0000b	
	LEDG[3..1]	101b	
	...r:controller_inst|main_state.DISPLAY_RESULTS		

Alias	Name	12	13
	RSTn		
	STARTn		
	DISPLAYn		
	main_controller:controller_inst|RESULT[16..0]	270	
	HEX0[6..0]	1000000b	
	HEX1[6..0]	1111000b	
	HEX2[6..0]	0100100b	
	HEX3[6..0]	1000000b	
	LED_SIGN		
	LEDS_1E4[3..0]	0000b	
	LEDG[3..1]	101b	
	...r:controller_inst|main_state.DISPLAY_RESULTS		

ias	Name	12	13
	RSTn		
	STARTn		
	DISPLAYn		
	main_controller:controller_inst|RESULT[16..0]	280	
	HEX0[6..0]	1000000b	
	HEX1[6..0]	0000000b	
	HEX2[6..0]	0100100b	
	HEX3[6..0]	1000000b	
	LED_SIGN		
	LEDS_1E4[3..0]	0000b	
	LEDG[3..1]	101b	
	...r:controller_inst|main_state.DISPLAY_RESULTS		

7.2 עבור מטריצה C :

ias	Name	12	13
	RSTn		
	STARTn		
	DISPLAYn		
	main_controller:controller_inst|RESULT[16..0]	-14069	
	HEX0[6..0]	0010000b	
	HEX1[6..0]	0000010b	
	HEX2[6..0]	1000000b	
	HEX3[6..0]	0011001b	
	LED_SIGN		
	LEDS_1E4[3..0]	0001b	
	LEDG[3..1]	101b	
	...r:controller_inst|main_state.DISPLAY_RESULTS		

ias	Name	12	13
	RSTn		
	STARTn		
	DISPLAYn		
	main_controller:controller_inst|RESULT[16..0]	-2571	
	HEX0[6..0]	1111001b	
	HEX1[6..0]	1111000b	
	HEX2[6..0]	0010010b	
	HEX3[6..0]	0100100b	
	LED_SIGN		
	LEDS_1E4[3..0]	0000b	
	LEDG[3..1]	101b	
	...r:controller_inst|main_state.DISPLAY_RESULTS		

7.3 **<u>בדיקה עבור לחיצת RSTn</u>**:

Type	Alias	Name			Basic Al
*		RSTn	✓	✓	⎍
*		STARTn	✓	✓	▩
*		DISPLAYn	✓	✓	▩
out		⊞ HEX0[6..0]	✓	✓	XXXXXXXb
out		⊞ HEX1[6..0]	✓	✓	XXXXXXXb
out		⊞ HEX2[6..0]	✓	✓	XXXXXXXb
out		⊞ HEX3[6..0]	✓	✓	XXXXXXXb
*		LED_SIGN	✓	✓	▩
out		⊞ LEDS_1E4[3..0]	✓	✓	XXXXb
out		⊞ LEDG[3..1]	✓	✓	XXXb

		Name	
*		STARTn	
*		DISPLAYn	
out		⊞ HEX0[6..0]	1111111b
out		⊞ HEX1[6..0]	1111111b
out		⊞ HEX2[6..0]	1111111b
out		⊞ HEX3[6..0]	1111111b
*		LED_SIGN	
out		⊞ LEDS_1E4[3..0]	0000b
out		⊞ LEDG[3..1]	001b
*		...r:controller_inst|main_state.DISPLAY_RESULTS	

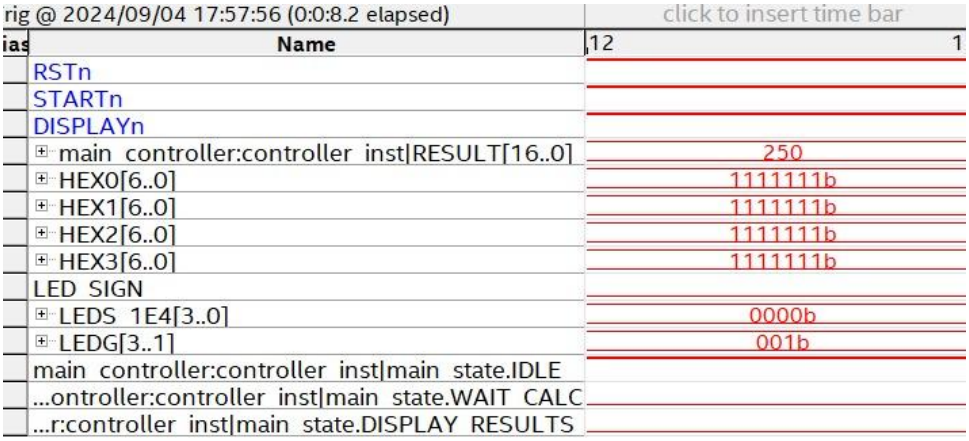
7.3 בדיקה עבור לחיצת STARTn:

Type	Alias	Name	59	59	Basic Al
*		RSTn	✓	✓	▩
*		STARTn	✓	✓	⎍
*		DISPLAYn	✓	✓	▩
out		⊞ main_controller:controller_inst|RESULT[16..0]	✓	✓	XXXXXXXXXX
out		⊞ HEX0[6..0]	✓	✓	XXXXXXXb
out		⊞ HEX1[6..0]	✓	✓	XXXXXXXb
out		⊞ HEX2[6..0]	✓	✓	XXXXXXXb
out		⊞ HEX3[6..0]	✓	✓	XXXXXXXb
*		LED_SIGN	✓	✓	▩
out		⊞ LEDS_1E4[3..0]	✓	✓	XXXXb
out		⊞ LEDG[3..1]	✓	✓	XXXb
*		main_controller:controller_inst|main_state.IDLE	✓	✓	▩
*		...ontroller:controller_inst|main_state.WAIT_CALC	✓	✓	▩
*		...r:controller_inst|main_state.DISPLAY_RESULTS	✓	✓	▩

לחיצה ראשונה:

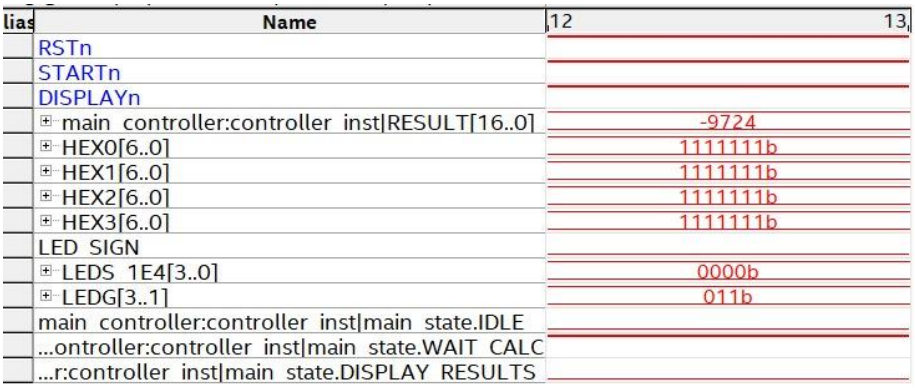
Alias	Name	12	13
	RSTn		
	STARTn		
	DISPLAYn		
	⊞ main_controller:controller_inst|RESULT[16..0]	250	
	⊞ HEX0[6..0]	1000000b	
	⊞ HEX1[6..0]	0010010b	
	⊞ HEX2[6..0]	0100100b	
	⊞ HEX3[6..0]	1000000b	
	LED_SIGN		
	⊞ LEDS_1E4[3..0]	0000b	
	⊞ LEDG[3..1]	101b	
	main_controller:controller_inst|main_state.IDLE		
	...ontroller:controller_inst|main_state.WAIT_CALC		
	...r:controller_inst|main_state.DISPLAY_RESULTS		

לחיצה שניה:



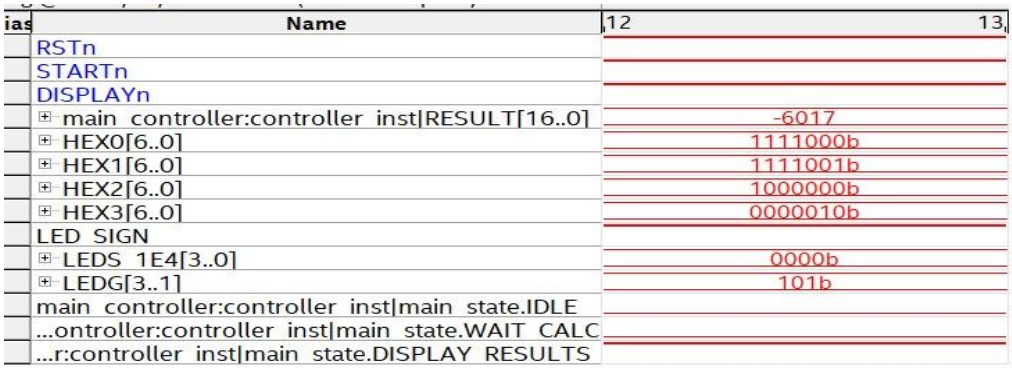
rig @ 2024/09/04 17:57:56 (0:0:8.2 elapsed)	click to insert time bar
Name	12 … 1
RSTn	
STARTn	
DISPLAYn	
main_controller:controller_inst|RESULT[16..0]	250
HEX0[6..0]	1111111b
HEX1[6..0]	1111111b
HEX2[6..0]	1111111b
HEX3[6..0]	1111111b
LED_SIGN	
LEDS_1E4[3..0]	0000b
LEDG[3..1]	001b
main_controller:controller_inst|main_state.IDLE	
...ontroller:controller_inst|main_state.WAIT_CALC	
...r:controller_inst|main_state.DISPLAY_RESULTS	

לחיצה שלישית:



Name	12 … 13
RSTn	
STARTn	
DISPLAYn	
main_controller:controller_inst|RESULT[16..0]	-9724
HEX0[6..0]	1111111b
HEX1[6..0]	1111111b
HEX2[6..0]	1111111b
HEX3[6..0]	1111111b
LED_SIGN	
LEDS_1E4[3..0]	0000b
LEDG[3..1]	011b
main_controller:controller_inst|main_state.IDLE	
...ontroller:controller_inst|main_state.WAIT_CALC	
...r:controller_inst|main_state.DISPLAY_RESULTS	

לחיצה רבעית:



Name	12 … 13
RSTn	
STARTn	
DISPLAYn	
main_controller:controller_inst|RESULT[16..0]	-6017
HEX0[6..0]	1111000b
HEX1[6..0]	1111001b
HEX2[6..0]	1000000b
HEX3[6..0]	0000010b
LED_SIGN	
LEDS_1E4[3..0]	0000b
LEDG[3..1]	101b
main_controller:controller_inst|main_state.IDLE	
...ontroller:controller_inst|main_state.WAIT_CALC	
...r:controller_inst|main_state.DISPLAY_RESULTS	

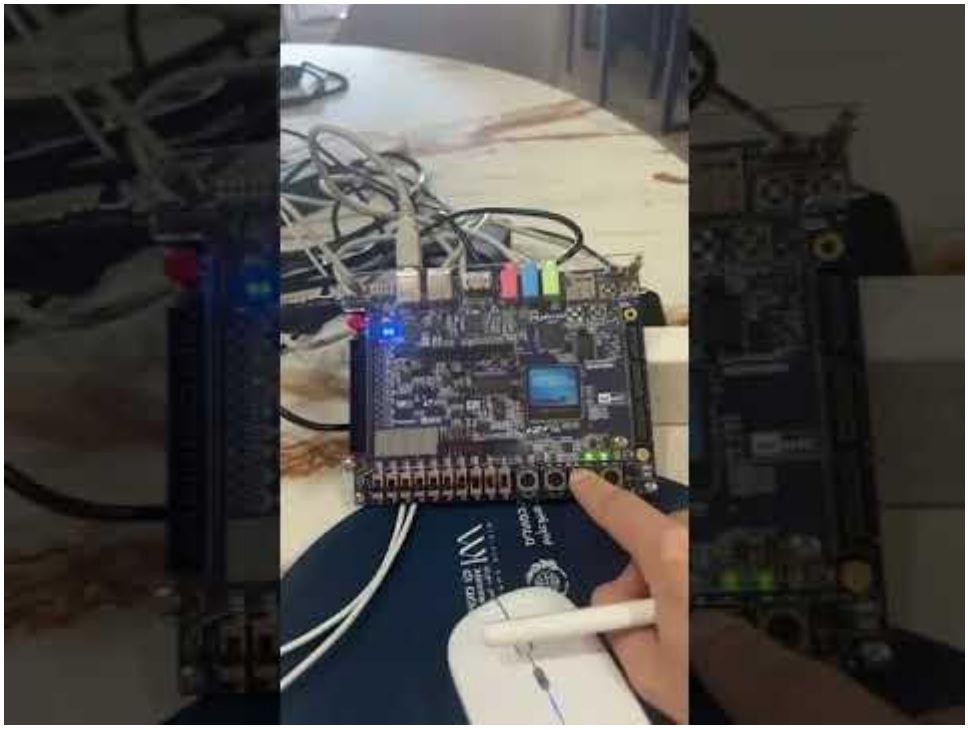
המערכת נמצאת במטריצה A ומציגה תוצאה של 250 על תצוגות ה-segment.7-

תצוגות HEX0 עד HEX3 מציגות את הספרות 0, 2, 5 ו-0 לפי הקודים הבינאריים המתאימים.

נורות LEDG [3..1]: LED מציגות 101b ה LED-הראשון והשלישי דולקים, השני כבוי .
LED_SIGN מראה ערך 0, שמציין שהתוצאה חיובית.

מכונת המצבים עוברת בין st_idle, st_wait_for_calculate ו st_display: המערכת מתחילה במצב המתנה, מחכה לחישוב ואז מציגה את התוצאה.

8. סרטון:



סיכום:

אפשר להגיד שהצלחנו לאמת את כל הסימולציות עשינו את המשימות כמו שצריך .