התקנים לוגים מתוכנתים VHDL

פרויקט אמצע

מגישים:

מחמוד חגה 318396355

נאיל חסון 318386364

בהנחיית:

ד"ר פאדל טריף

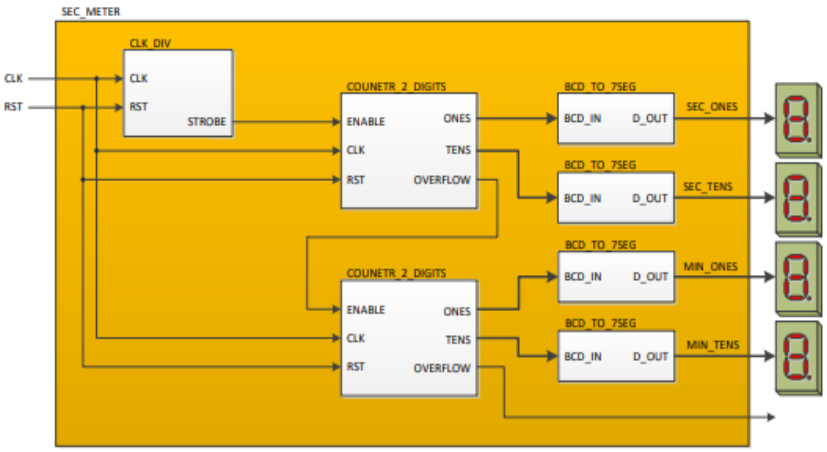
מר אורן זלץ

תוכן עניינים

דרישות המערכת

- המערכת תופעל משעון יחיד של 50 MHz

- למערכת כניסת איפוס (RESET) אסינכרונית שפועלת בנמוך



איור 1: סכמת בלוקים של ה-SEC_METER

סכמת המלבנים כוללת 7 בלוקים:

- הבלוק CLK_DIV שתפקידו לחלק את תדר המערכת (שבערכה שלנו הוא 50MHz) וליצר לנו בסיס זמן של 1 שניה.
- הבלוק COUNTER_2_DIGITS שמופיע פעמיים ותפקידו לספור מ-0 ועד הערך הגנרי COUNT_SIZE.
- הבלוק BCD_TO_7SEG שמופיע 4 פעמים ותפקידו להדליק תצוגת 7 מקטעים.

על מנת להדליק את תצוגת 7 מקטעים בעלייה בקצב 1 לשנייה, **צריך לחלק תדר השעון ב 50MHz**.

בפרויקט שלנו בחרנו את הערך המקסימלי למונה להיות 20. (**היה אפשרי לבחור כל ערך אחר(שהוא קטן מ100)**, בחרנו 20 בגלל שהוא קטן יחסית, וניתן לראות את השינוי בתצוגת המקטעים בזמן מהיר יותר בהשוואה לערך שגדול מ20).

קוד התוכנית+סימולציה

CLK_DIV

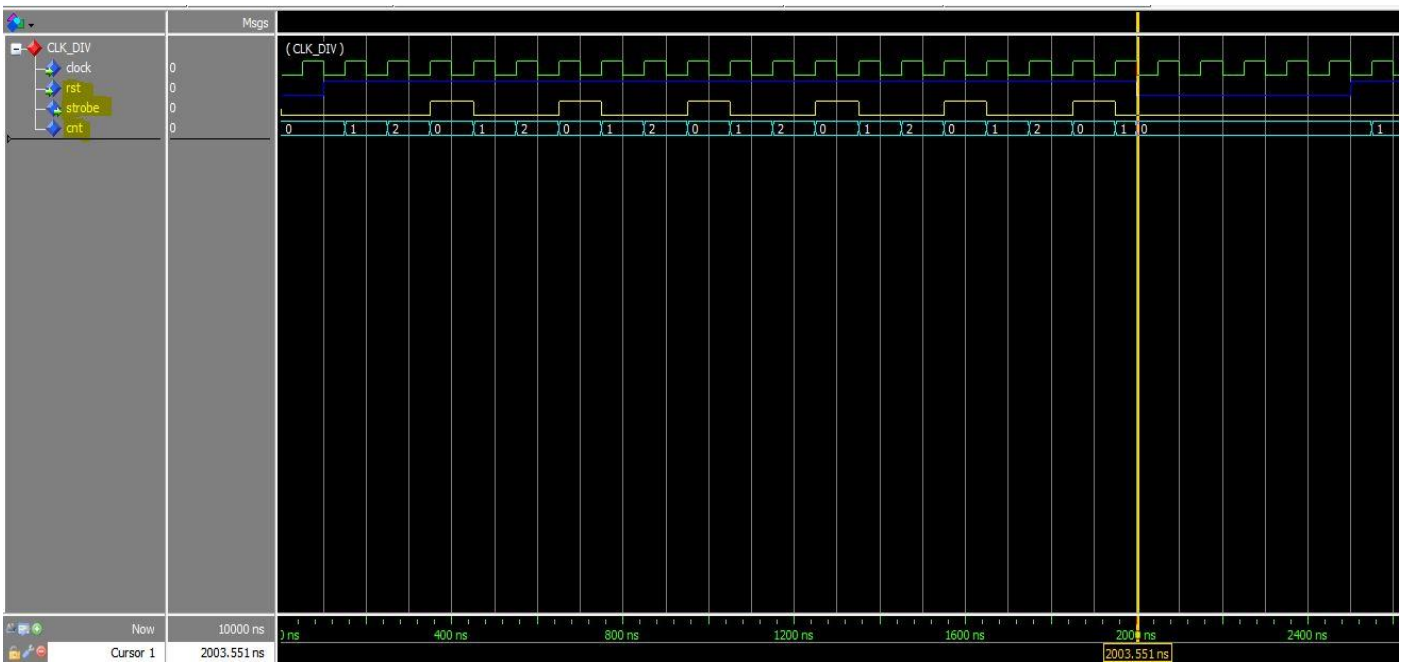
```
library ieee;
use ieee.std_logic_1164.all;

entity clk_div is
    generic (divisor: integer);
    port (
        clock  : in std_logic;
        rst    : in std_logic;
        strobe : out std_logic
    );
end clk_div;

architecture behave of clk_div is
    signal cnt: integer range 0 to divisor := 0;  -- Counter for tracking clock cycles
begin
    process (clock, rst)
    begin
        if rst = '0' then
            strobe <= '0';  -- Reset strobe signal
            cnt <= 0;       -- Reset counter
        elsif rising_edge(clock) then
            if cnt = divisor - 1 then
                strobe <= '1';  -- Activate strobe signal
                cnt <= 0;       -- Reset counter
            else
                cnt <= cnt + 1;  -- Increment counter
                strobe <= '0';  -- Deactivate strobe signal
            end if;
        end if;
    end process;
end behave;
```

TESTBANCH for CLK_DIV

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  entity clk_div_TB is
4  end;
5  architecture simple of clk_div_TB is
6    -- Components and Signals Declaration
7    Component clk_div is
8      generic (divisor: integer);
9      port (
10       clock, rst: in std_logic;
11       strobe: out std_logic
12     );
13   end component;
14   constant div: integer := 3; -- Desired divisor value
15   constant clk_period: time := 100 ns;
16   signal S_rst: std_logic := '0';
17   signal S_clk: std_logic := '0';
18   signal S_strobe: std_logic := '0';
19 begin
20   -- DUT instantiation
21   DUT: clk_div
22     generic map (divisor => div)
23     port map (
24       rst => S_rst,
25       clock => S_clk,
26       strobe => S_strobe
27     );
28   -- Signal's Waves Creation
29   S_rst <= '0', '1' after 100 ns, '0' after 2000 ns, '1' after 2500 ns;  --  reset signal (reset all when '0')
30   S_clk <= not S_clk after clk_period / 2; -- Generate a clock signal with half the specified period
31 end simple;
```

בחרנו בTESTBANCH ערך המחלק להיות 3 (שורה 14), אז בתוצאות חייבים לקבל ביציאה STROBE תדר השעות מחולק ב 3.

תוצאת הסימולציה



קבלנו תדר השעון מחולק ב 3 (ניתן לראות את זה תוך משתנה STROBE), רואים גם המונה (CNT) מתקדם בערך 1 כל עליית שעון, עד שיגיע ל 3.

COUNTER_2_DIGITS

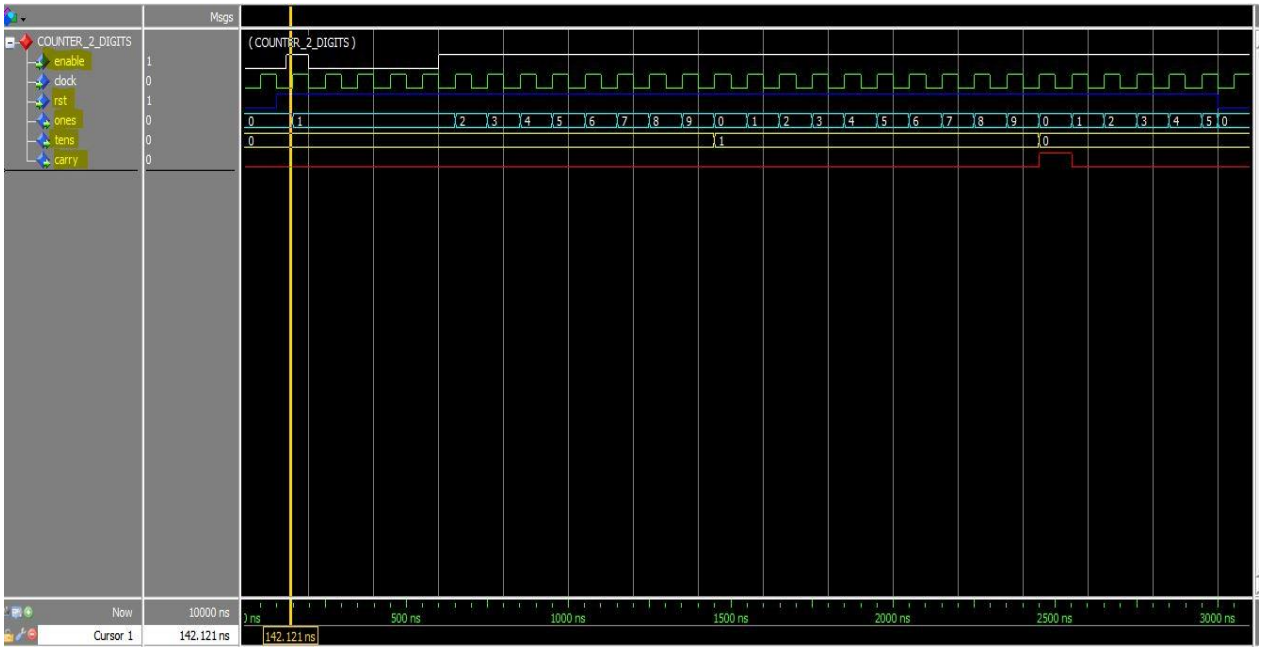
```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   entity COUNTER_2_DIGITS is
4       generic (count_size: integer);
5       port (
6           enable : in  std_logic;
7           clock  : in  std_logic;
8           rst    : in  std_logic;
9           ones   : out integer range 0 to 9;
10          tens   : out integer range 0 to 9;
11          carry  : out std_logic := '0'
12      );
13  end entity;
14
15  architecture behave of COUNTER_2_DIGITS is
16      signal cnt: integer range 0 to count_size;
17
18  begin
19
20      process (clock, rst)
21      begin
22          if rst = '0' then
23              carry <= '0';        -- Reset carry signal
24              cnt <= 1;            -- Reset counter
25              ones <= 0;           -- Reset ones digit
26              tens <= 0;           -- Reset tens digit
27          elsif rising_edge(clock) then
28              carry <= '0';        -- Clear carry signal
29
30            if enable = '1' then
31                if cnt = count_size - 1 then
32                    cnt <= 0;       -- Reset counter
33                    ones <= cnt rem 10;     -- Calculate ones digit
34                    tens <= cnt / 10;       -- Calculate tens digit
35                else
36                    cnt <= cnt + 1;         -- Increment counter
37                    ones <= cnt rem 10;     -- Calculate ones digit
38                    tens <= cnt / 10;       -- Calculate tens digit
39                end if;
40
41                if cnt = 0 then
42                    carry <= '1';   -- Set carry signal if counter wraps around
43                end if;
44            end if;
45        end if;
46    end process;
47
48  end behave;
```

TESTBANCH for COUNTER_2_DIGITS

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity COUNTER_2_DIGITS_tb is
5  end;
6
7  architecture simple of COUNTER_2_DIGITS_tb is
8
9    -- Components and Signals Declaration
10   Component COUNTER_2_DIGITS is
11     generic (count_size: integer);
12     port (
13       enable : in std_logic;
14       clock  : in std_logic;
15       rst    : in std_logic;
16       ones   : out integer range 0 to 9;
17       tens   : out integer range 0 to 9;
18       carry  : out std_logic
19     );
20   end component;
21
22   constant clk_period : time := 100 ns;
23   constant cnt_size   : integer := 20; -- Desired count size
24   signal S_rst        : std_logic := '0';
25   signal S_clk        : std_logic := '0';
26   signal S_enable     : std_logic := '0';
27   signal S_carry      : std_logic := '1';
28
29   signal S_ones       : integer range 0 to 9;
30   signal S_tens       : integer range 0 to 9;
31
32 begin
33
34   -- DUT instantiation
35   DUT: COUNTER_2_DIGITS
36     generic map (count_size => cnt_size)
37     port map (
38       rst    => S_rst,
39       clock  => S_clk,
40       enable => S_enable,
41       ones   => S_ones,
42       tens   => S_tens,
43       carry  => S_carry
44     );
45
46   -- Signal's Waves Creation
47   S_rst    <= '0', '1' after 100 ns, '0' after 3000 ns, '1' after 5000 ns; -- Toggle the reset signal at specific time
48   S_enable <= '0', '1' after 130 ns, '0' after 200 ns, '1' after 600 ns; -- Toggle the enable signal at specific time :
49   S_clk    <= not S_clk after clk_period / 2; -- Generate a clock signal with half the specified period
50
51 end simple;
```

בחרנו ב- TESTBANCH ערך המונה להיות 20 (שורה 23), אז המונה אמור להגיע מקסימום ל20-1, (מ 0 עד 19), וערך האחדות של המונה יכנס למשתנה ONES, וערך העשרות יכנס למשתנה TENS.

תוצאת הסימולציה



רואים כאשר הenable שווה ל1 המונה גודל ב1 עם כל עליית שעון (רואים שבין 200ns -500ns ערך ה enable שווה ל 0, לכן המונה לא התקדם ונשאר 1)

רואים ש ONES מקבל ערך האחדות של המונה, TENS מקבל ערך העשרות של המונה.

סביב 2500ns המונה מגיע ל ערך המקסימלי (20), אז רואים שערך הcarry עכשיו שווה ל1, גם ערכים הONES , TENS הפכו ל0.

BCDTO_7SEG

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

entity BcdTo_7SEG is
    -- Input: a single BCD digit in the range 0 to 9.
    Port (
        bcd_in : in INTEGER RANGE 0 to 9;
        -- Output: a 7-bit vector representing the digit in 7-segment display format.
        d_out : out STD_LOGIC_VECTOR (6 downto 0)
    );
end entity;

architecture behave of BcdTo_7SEG is
begin
    d_out <= "1000000" when bcd_in = 0 else
             "1111001" when bcd_in = 1 else
             "0100100" when bcd_in = 2 else
             "0110000" when bcd_in = 3 else
             "0011001" when bcd_in = 4 else
             "0010010" when bcd_in = 5 else
             "0000010" when bcd_in = 6 else
             "1111000" when bcd_in = 7 else
             "0000000" when bcd_in = 8 else
             "0010000" when bcd_in = 9 else
             "1111111"; -- If the input is invalid, display a blank character.
end architecture;
```

TESTBANCH for BCDTO_7SEG

```
library ieee;
use ieee.std_logic_1164.all;
entity BcdTo_7SEG_tb is end;
architecture simple of BcdTo_7SEG_tb is
------- Componrnts and Signals Declaration -----
Component BcdTo_7SEG is
PORT ( BCD_in  : IN   integer range 0 to 9;
       D_out    : OUT std_logic_vector (6 DOWNTO 0));
end component;
constant clk_period: time:=100 ns;
signal S_BCD_in: integer range 0 to 9;
signal S_D_out: std_logic_vector (6 DOWNTO 0);

begin
------------ DUT instantiation -----------------
DUT: BcdTo_7SEG
   port map (
                BCD_in      => S_BCD_in,
                D_out   => S_D_out);
------- Signal's Waves Creation ----------------
S_BCD_in <= 0, 1 after 1 us, 2 after 2 us, 3 after 3 us, 4 after 4 us, 5 after 5 us, 6 after 6 us, 7 after 7 us, 8 after 8 us,

end simple;
```

בחרנו בTESTBANCH להציג את כל הערכים של BCD (0-9) כדי להבטיח שנותן את כל הערכים בצורה נכונה.

תוצאת הסימולציה

רואים עבור כל כניסה BCD_IN מקבלים תוצאה ב D_OUT של ערך המספר ב- 7 SEG.

SEC_METER

(מחלקים תדר השעון ב 50M, סופרים עד20)

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

entity SEC_METER is
generic (div: integer:=50000000;count_limit: integer:=20);
    port(
        clk             : in std_logic;
        reset           : in std_logic;
        sec_ones_7seg   : out std_logic_vector(6 downto 0);
        sec_tens_7seg   : out std_logic_vector(6 downto 0);
        min_ones_7seg   : out std_logic_vector(6 downto 0);
        min_tens_7seg   : out std_logic_vector(6 downto 0);
        hour_led        : out std_logic
        );
end SEC_METER;
architecture behave of SEC_METER is

component clk_div
generic (divisor: integer);
port(
        clock   :in std_logic;
        rst     :in std_logic;
        strobe  : out std_logic
    );
end component;

component COUNTER_2_DIGITS
generic (count_size: integer);
port(
        enable  : in std_logic;
        clock   : in std_logic;
        rst     : in std_logic;
        ones    : out integer range 0 to 9;
        tens    : out integer range 0 to 9;
        carry   : out std_logic
    );
end component;

component BcdTo_7SEG
PORT (
        BCD_in  : IN   integer range 0 to 9;
        D_out    : OUT std_logic_vector (6 DOWNTO 0));
end component;

signal Strobe_1sec  : std_logic:='0';
signal Strobe_1min  : std_logic:='0';
signal sec_ones     :integer range 0 to 9;
signal sec_tens     :integer range 0 to 9;
signal min_ones     :integer range 0 to 9;
signal min_tens     :integer range 0 to 9;
begin

```

```
54   u1: clk_div
55   generic map (divisor=>div)
56   port map (
57               clock=>clk,
58               rst=>reset,
59               strobe=>strobe_1sec
60               );
61
62   u2:COUNTER_2_DIGITS
63   generic map (count_size=>count_limit)
64   port map (
65               clock=>clk,
66               rst=>reset,
67               enable=>strobe_1sec,
68               ones=>sec_ones,
69               tens=>sec_tens,
70               carry=>strobe_1min
71               );
72
73   u3:COUNTER_2_DIGITS
74   generic map (count_size=>count_limit)
75   port map (
76               clock=>clk,
77               rst=>reset,
78               enable=>strobe_1min,
79               ones=>min_ones,
80               tens=>min_tens,
81               carry=>hour_led
82               );
83
84   u4:BcdTo_7SEG
85   port map (
86               bcd_in=>sec_ones,
87               d_out=>sec_ones_7seg
88               );
89
90   u5:BcdTo_7SEG
91   port map (
92               bcd_in=>sec_tens,
93               d_out=>sec_tens_7seg
94               );
95
96   u6:BcdTo_7SEG
97   port map (
98               bcd_in=>min_ones,
99               d_out=>min_ones_7seg
100              );
101
102  u7:BcdTo_7SEG
103  port map (
104              bcd_in=>min_tens,
105              d_out=>min_tens_7seg
106              );
107
108  end behave;
```

בנינו את הקוד לפי דרישת המערכת, ודאגנו לכניסות ויציאות של כל בלוק.

CLK DIV *1

COUNTER 2 DIGITS *2

BCD TO 7 SEG *4

TESTBANCH

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

entity SEC_METER_tb is end;
architecture simple of SEC_METER_tb is
------- Componrnts and Signals Declaration -----
Component SEC_METER is
generic (div: integer;count_limit: integer);
    port(
        clk              : in std_logic;
        reset            : in std_logic;
        sec_ones_7seg    : out std_logic_vector(6 downto 0);
        sec_tens_7seg    : out std_logic_vector(6 downto 0);
        min_ones_7seg    : out std_logic_vector(6 downto 0);
        min_tens_7seg    : out std_logic_vector(6 downto 0);
        hour_led         : out std_logic);
end component;

constant clk_period : time:=100 ns;
constant divv    : integer:=5;
constant count_limitx    : integer:=20;


signal S_rst          : std_logic:='0';
signal S_clk          : std_logic:='0';

signal S_sec_ones_7seg  : std_logic_vector(6 downto 0);
signal S_sec_tens_7seg  : std_logic_vector(6 downto 0);
signal S_min_ones_7seg  : std_logic_vector(6 downto 0);
signal S_min_tens_7seg  : std_logic_vector(6 downto 0);

--signal S_strobe: std_logic:='1';

begin
------------ DUT instantiation -----------------
DUT: SEC_METER
generic map (div=>divv, count_limit=>count_limitx)
    port map (
                reset       => S_rst,
                clk         => S_clk,
                sec_ones_7seg=>S_sec_ones_7seg,
                sec_tens_7seg=>S_sec_tens_7seg,
                min_ones_7seg=>S_min_ones_7seg,
                min_tens_7seg=>S_min_tens_7seg
            );

S_rst   <='0','1' after 100 ns, '0' after 16000 ns, '1' after 20000 ns, '0' after 25000 ns, '1' after 26000 ns;
S_clk <= not S_clk after clk_period / 2;

end simple;
```

בחרנו בTESTBANCH שהמונה יספור עד 20, ותדר השעון יחולק ב5.(שורה 21,22)

תוצאת הסימולציה

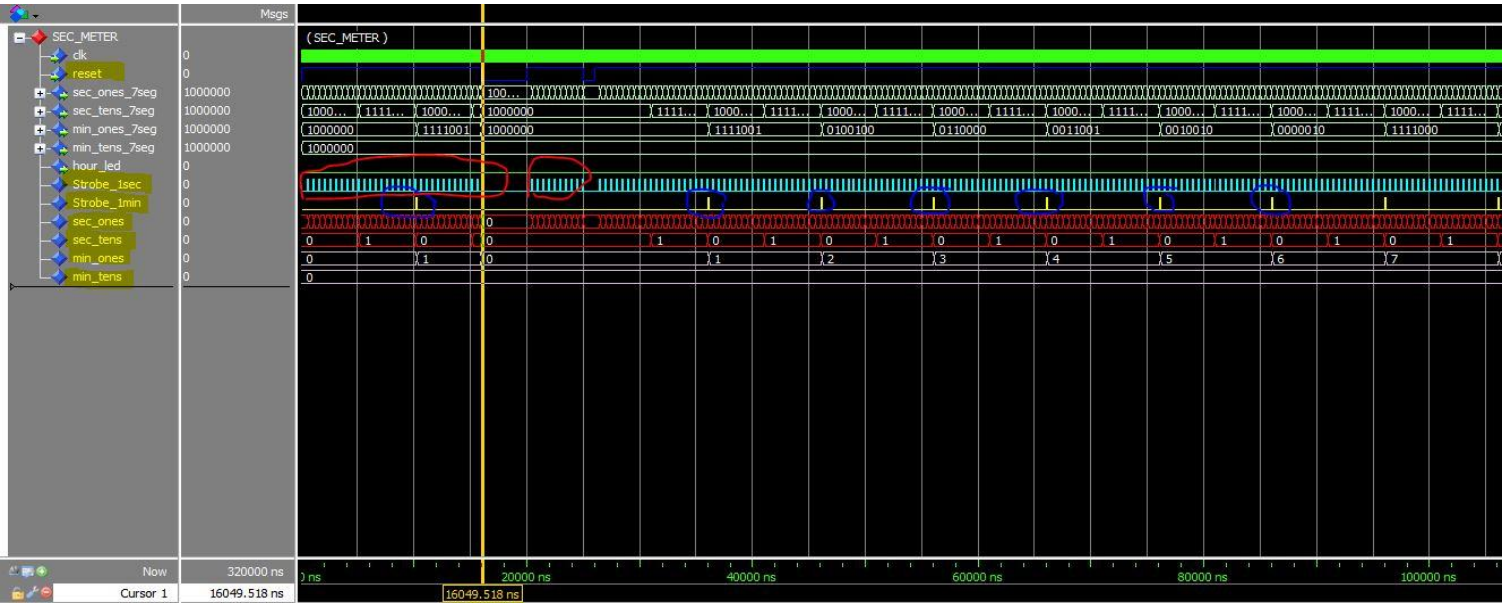
SEC ONES, SEC TENS מציגים את השניות באחדות ועשרות.

MIN ONES, MIN TENS מציגים את הדקות באחדות ועשרות.

STORBE 1SEC הוא enable למונה של השניות (כאשר הוא 1 אז מונה השניות יתקדם ב 1).

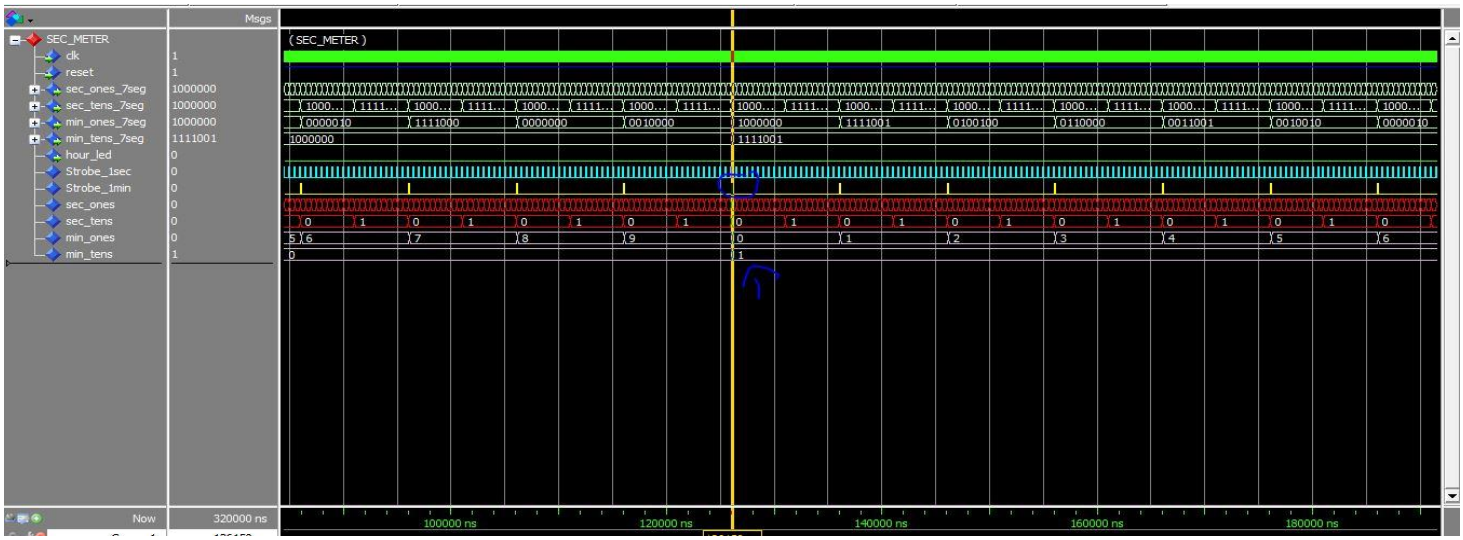
STORBE 1 MIN הוא enable למונה של הדקות (כאשר הוא 1 אז מונה הדקות יתקדם ב 1), הוא גם יציאת הCARRY של מונה השניות, כאשר יהיה לנו CARRY במונה השניות, אז יהיה enable למונה הדקות. (ניתן לראות את זה בצורה יותר ברורה בתמונה הבאה)

בתמונה רואים שבזמן 16050ns הreset יורד ל0, אז הערכים של MIN ONES, MIN TENS SEC ONES, SEC TENS, הופכים להיות 0.



בתמונה רואים העיגולים בצבע אדום הם ה enable של מונה השניות (STROBE 1 SEC)

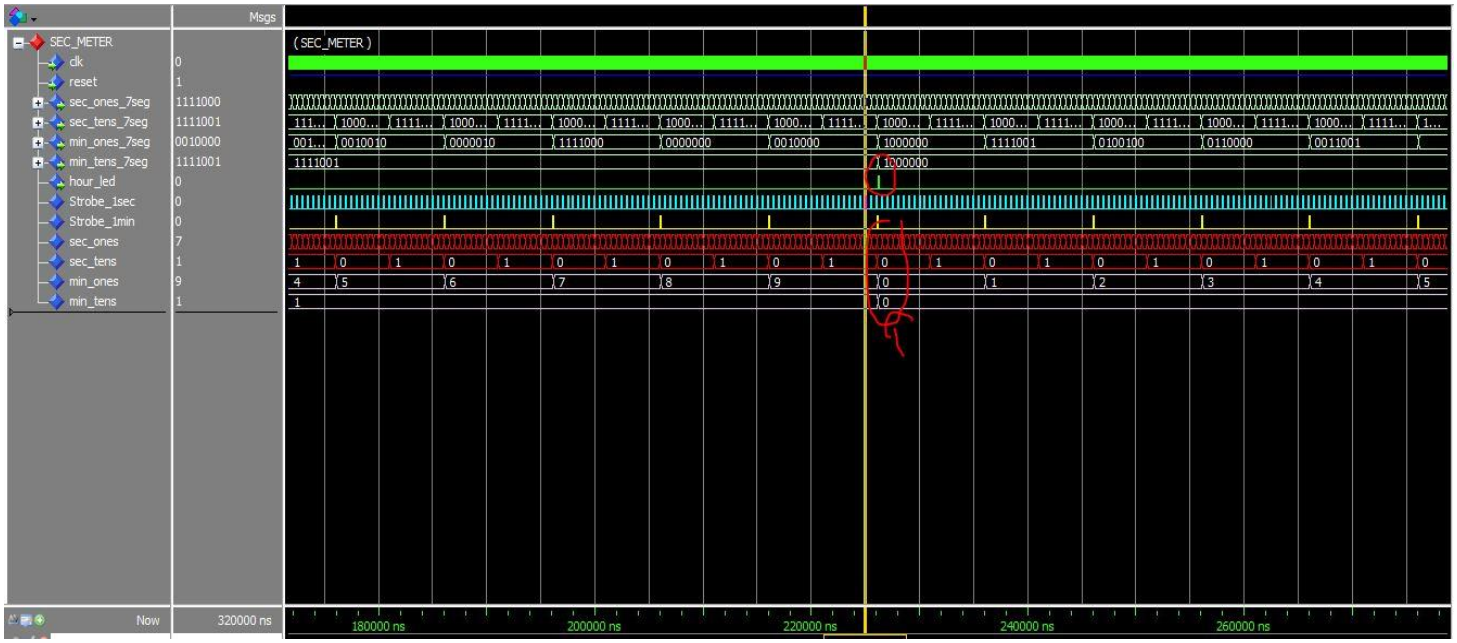
העיגולים בצבע כחול הם הCARRY של מונה השניות, וגם ה enable של מונה הדקות.



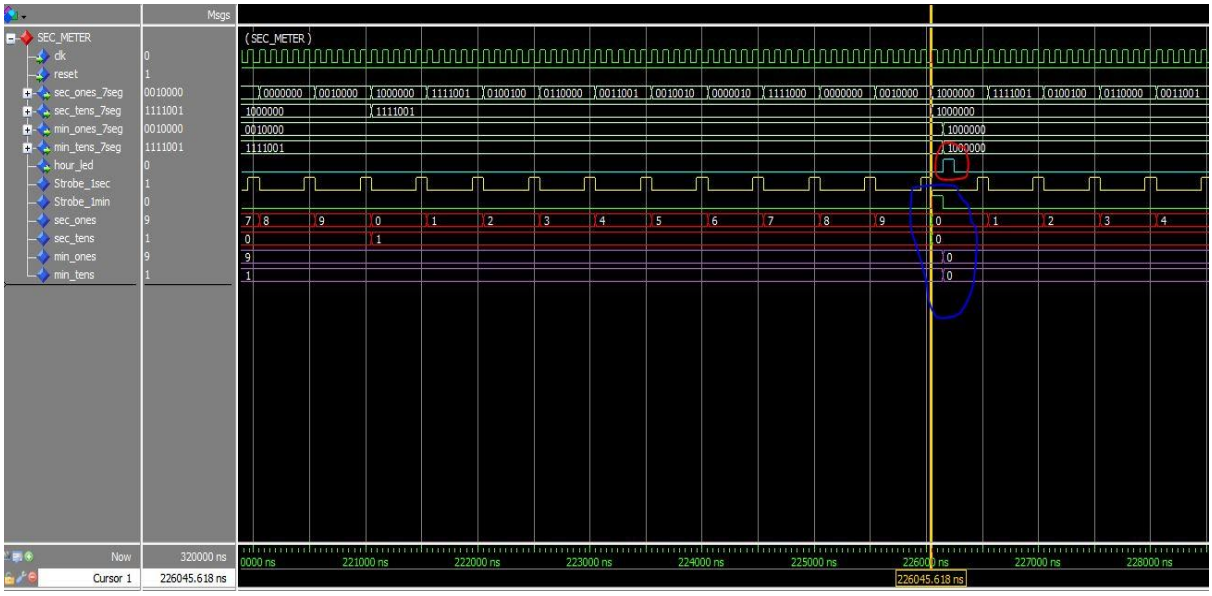
בתמונה הזאת בעיגול הכחול מופיע enable למונה הדקות, בזמן זה מונה השניות היה 19, ומונה הדקות היה 09, אז כאשר יהיה enable למונה הדקות, מונה השניות יהפוך ל 00, ומונה הדקות יהיה 10.

10 : 00

בתמונה רואים אותו הסבר של תמונה למעלה רק בקירוב, רואים שמונה השניות היה 19, ואחר כך הפך ל09, ומונה הדקות היה 00, ואחר כך הפך ל 10.

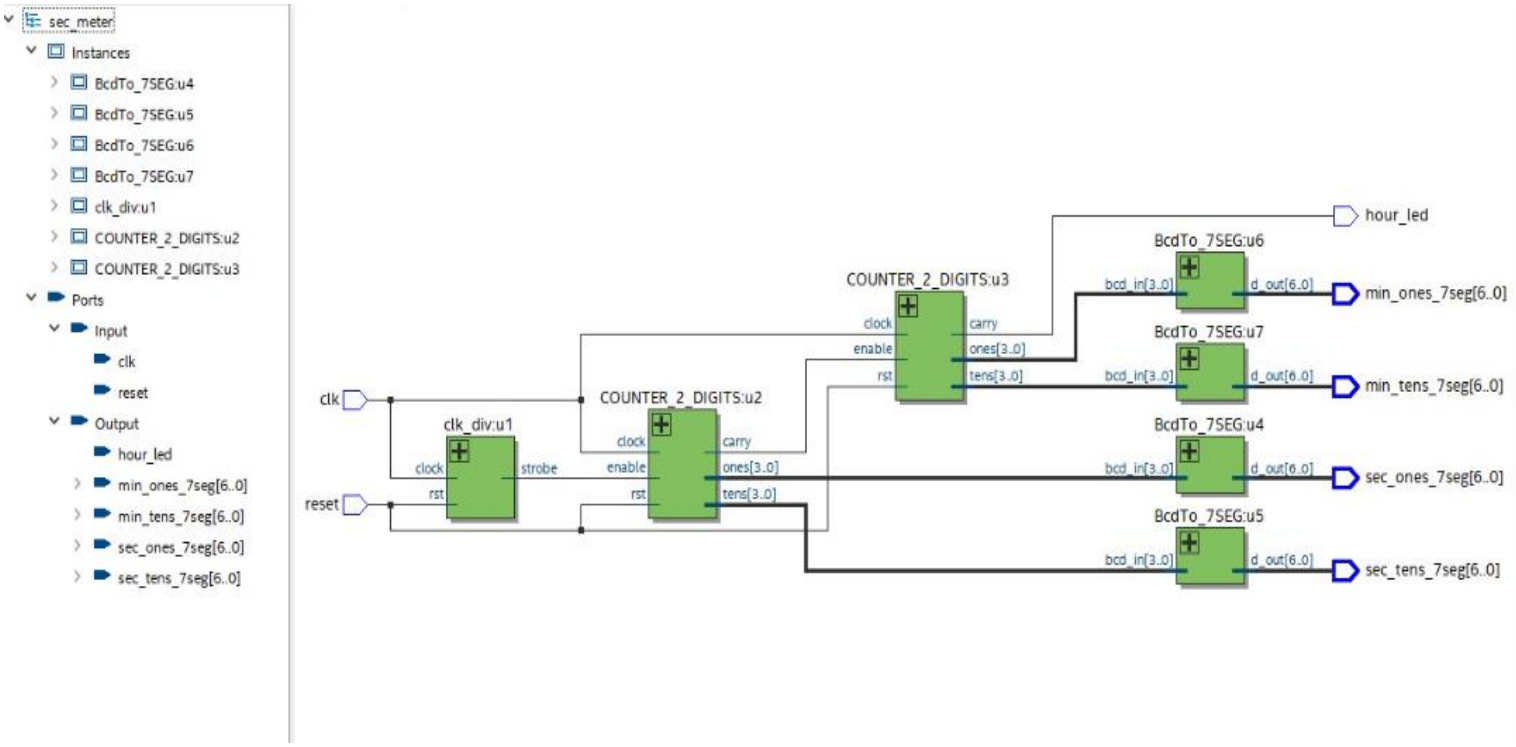


בתמונה הזאת בעיגול האדום מופיע enable למונה הדקות, בזמן זה מונה השניות היה 19, ומונה הדקות היה 19, אז כאשר יהיה enable למונה הדקות, מונה השניות יהפוך ל 00, ומונה הדקות יהיה 00, ויופיע carry במונה הדקות שהוא תוך המשתנה (hour led) שהוא מדליק LED בערכה שלנו.



בתמונה רואים אותו הסבר של תמונה למעלה רק בקירוב, רואים שמונה השניות היה 19, ואחר כך הפך ל00, ומונה הדקות היה 19, ואחר כך הפך ל 00, כמו כן רואים שהLED (hour_led) עכשיו נדלק (עיגול אדום).

מבנה לוגי של מבנית ושל הרמה העליונה



תיאור תקלות במהלך העבודה

1)

בקוד של counter 2 digits

היה לנו בעייה שהמונה היה גולש מהגבול המקסימלי,הבעייה הייתה בקוד שלנו, בגלל שהיינו מקדמים המונה בהתחלה אחר כך שואלים אם המונה קטן מהגבול המקסימלי, וזה היה נותן שגיאה.

```
architecture behave of Counter_2_digit is
begin
process(clock, rst)
variable cnt: integer range 0 to count_size;
begin
    if rst = '1' then
        carry <= '0';
        cnt := 0;
        ones <= 0;
        tens <=0;
    elsif clock'event and clock = '1' then
        carry <= '0';
        if enable = '1' then
            cnt := cnt +1;
            if cnt = count_size then
                carry <= '1';
                cnt := 0;
            end if;
            ones <= cnt rem 10;
            tens <= cnt/10;
        end if;
    end if;
end process;
```

2)

בעייה בקוד של counter 2 digits עם קוד sec meter.

כאשר במונה של השניות היה יוצא carry, והוא היה נכנס לenable של מונה הדקות, משך זמן הenable היה ארוך מדי, כך ש מונה הדקות היה סופר 5 מספרים לפני שה enable יהיה 0.

הבעייה הייתה בגלל הcarry היוצא ממונה השניות, שהוא היה שווה ל1 משך זמן ארוך, אז דאגנו שהcarry יהיה 1 רק למשך pulse אחד, על ידי איפוס הcarry בכל עליית שעון.

```
21      process (clock, rst)
22      begin
23          if rst = '0' then
24              carry <= '0';          -- Reset carry signal
25              cnt <= 1;              -- Reset counter
26              ones <= 0;             -- Reset ones digit
27              tens <= 0;             -- Reset tens digit
28          elsif rising_edge(clock) then
29
30              if enable = '1' then
31                  if cnt = count_size - 1 then
32                      cnt <= 0;          -- Reset counter
33                      ones <= cnt rem 10;      -- Calculate ones digit
34                      tens <= cnt / 10;        -- Calculate tens digit
35                      carry <= '1';          -- Clear carry signal
36                  else
37                      cnt <= cnt + 1;          -- Increment counter
38                      ones <= cnt rem 10;      -- Calculate ones digit
39                      tens <= cnt / 10;        -- Calculate tens digit
40                      carry <= '0';          -- Clear carry signal
41                  end if;
42              end if;
43          end if;
44      end process;
```

2)

בעייה בקוד counter 2 digits.

כאשר ה reset במערכת ירד ל0 אחר כך יעלה ל 1, אז המונה לא מתעדכן בעליית השעון הראשונה, אלא בעליית השעון השנייה.

זה יצר לנו בעייה בשני המונים, מונה השניות ומונה הדקות, במונה השניות היה כמו השהייה אחרי ה reset.

במונה של הדקות היה גם השהייה על הenable השני, לדוגמה היה מונה השניות מגיע ל 19 ומונה הדקות 00 אחר כן מונה השניות גודל ב 1, ויהיה שווה ל 00, אבל מונה הדקות גם ישאר 00 ולא יגדל ל 01. מונה הדקות יגדל ב enable השני.

פתרנו הבעייה על ידי תיקון הקוד שלנו.

```
21      process (clock, rst)
22      begin
23          if rst = '0' then
24              carry <= '0';        -- Reset carry signal
25              cnt <= 1;            -- Reset counter
26              ones <= 0;           -- Reset ones digit
27              tens <= 0;           -- Reset tens digit
28          elsif rising_edge(clock) then
29              carry <= '0';        -- Clear carry signal
30
31              if enable = '1' then
32                  if cnt = count_size - 1 then
33                      cnt <= 0;        -- Reset counter
34                      ones <= cnt rem 10;      -- Calculate ones digit
35                      tens <= cnt / 10;        -- Calculate tens digit
36                  else
37                      cnt <= cnt + 1;          -- Increment counter
38                      ones <= cnt rem 10;      -- Calculate ones digit
39                      tens <= cnt / 10;        -- Calculate tens digit
40                  end if;
41                  if cnt = 0 then
42                      carry <= '1';    -- Set carry signal
43                  end if;
44              end if;
45          end if;
46      end process;
```

סרטון לפרויקט שלנו ב YouTube

סרטון הפרויקט:

https://youtu.be/jH_z6gseHC0

סרטון כאשר המונה מגיע לגבול המקסמלי:

https://youtu.be/E6WdZ9wxXF0