הטכניון – מכון טכנולוגי לישראל
הפקולטה להנדסת חשמל

מעבדות 1, 1ח

ניפוי תקלות בחומרה (DEBUG)

דו"ח מכין - שאלות ותרגילי הכנה

HS DSL הניסוי פותח בחסות המעבדה למערכות ספרתיות מהירות

גרסה 1.1 (קיץ 2018)

עורכים: ארמנד שוקרון, ליאת שורץ
על פי החוברת המקורית של עמוס זסלבסקי

הנחיות

- קובץ זה הוא גם תבנית לדו"ח המכין, יש לשמור ב- PDF ולהגיש במודל.

תאריך הגשת דו"ח ההכנה	
שם המדריך	

סטודנט	שם פרטי	שם משפחה
1	ברק	זן
2	בועז	טייטלר

תוכן עניינים של דו"ח מכין DEBUG

1 מכונת RANDOM

בהתייחס למכונה ליצירת מספר אקראי RANDOM שתוארה בחומר הרקע ענה על השאלות הבאות:

א. הסבר מדוע היציאה RANDOM[7..0] היא מספר אקראי?

תשובה: ערך היציא המשתנה כאשר לוחצים על לחצן, ברגע הלחיצה נדגם ערך יציאה ממונה מעגלי שרץ **מהר מאוד** על הערכים 0-255, כיוון שהלחיצה היא איטית משמעותית מזמן המחזור של המונה המעגלי וכיוון שאין לנו דרך לדעת באיזה ערך המונה נמצא (או להגיב מספיק מהר גם אם איכשהו נדע) – אפשר להגיד שהיציאה רנדומלית.

ב . מהו תפקידה של היחידה vrise?

תשובה: תפקיד היחידה הוא ברגע הלחיצה על הלחצן להוציא פולס ברוחב מחזור שעון **אחד** ע"מ שערך המונה ידגם רק פעם אחת ליציאה. (כיוון שאורך לחיצה אנושית יהיה ארוך משמעותית מאורך זמן המחזור של השעון)

ג . למה משמש הקבוע MAXCOUNT?

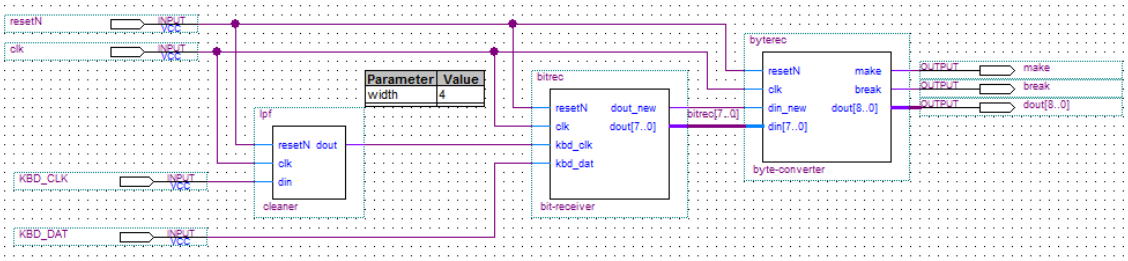
תשובה: ל"איפוס" המונה לערך מקסימלי ברגע שהוא מגיע ל0

ד. כיצד ניתן לשנות את המכונה כך שתגריל מספרים זוגיים בלבד?

תשובה: להחליף את המונה במונה שרץ רק על מספרים זוגיים **או** להוסיף ביט LSB 0 לכל מספר שנדגם

2 ממשק למקלדת

כפי שהוסבר בחומר הרקע לניסוי זה, התכן הסינכרוני הבא נבחר למימוש **ממשק חומרה למקלדת**.



כל אחת מהיחידות הנ"ל כתובה בשפת VHDL ותשמש לבנית הממשק שלך למקלדת במעבדה זו.
להלן הקבצים שבהם תעשה שימוש:

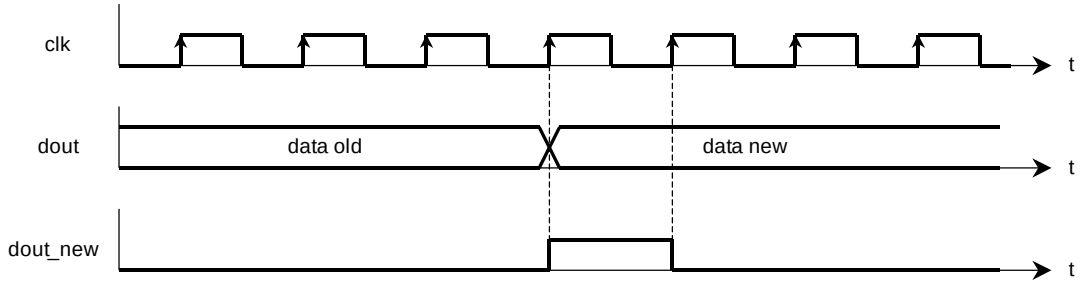
- ☐ יחידת מסנן מעביר נמוכים: lpf.vhd, lpf.bsf – נתונה לך במודל
- ☐ יחידת המקלט ברמת ה Bit: bitrec.vhd - נתון שלך שלה במודל
- ☐ יחידת המקלט ברמת ה – Byte: byterec.vhd, byterec.bsf – נתונה לך במודל

הערה חשובה: יש להביא למעבדה את כל הרכיבים אותם אתה כותב במסגרת עבודת ההכנה.

פתח את KBDINTF.QAR מהמודל.

2.1 תכן יחידת ה - BITREC

רקע למטלה: כמו שהוסבר בחומר הרקע תפקידה של היחידה שמטפלת בתשדורת הטורית BITREC הוא, להפיק מהמידע הטורי שמגיע לכניסות kbd_clk ו kbd_dat, מידע מקבילי ביציאה dout, יחד עם יציאת חיווי שפעילה למשך מחזור שעון אחד ושנקראת dout_new. דיאגרמת הזמנים הבאה מתארת אותות אלו אחד ביחס לשני וביחס לאות השעון:



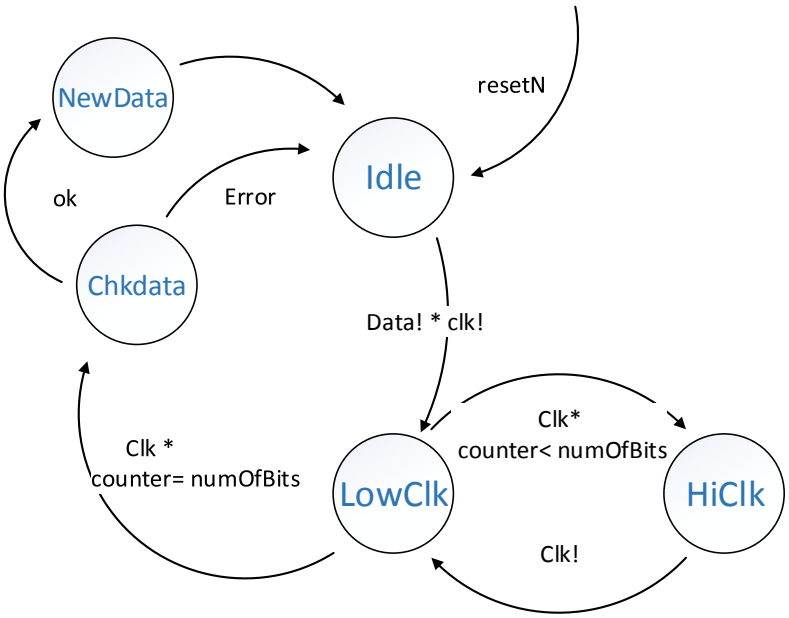
נתון לך הקובץ bitrec.vhd שהוא שלד המכיל את כל החלקים הדרושים כפי שהוסבר בחומר הרקע פרט למכונת המצבים.

שים לב! השתמש אך ורק בקובץ הנתון לך כעת במודל ולא בגרסאות אחרות מסמסטרים קודמים!

הוסף לקובץ זה את הקוד של מכונת המצבים, כפי שתתואר להלן, במקומות בקובץ שבהם כתובה ההערה:

```
-- WRITE YOUR CODE HERE --
```

מכונת מצבים (מסוג Moore) משמשת כבקר של היחידה. דיאגרמת המצבים הבאה מתארת את התנהגותה.



בדיאגרמה הנ"ל השתמשנו **בקיצורים** הבאים:

- clk מציין את האות Kbd_CLK בגבוה, ו- clk! בנמוך
- Data מציין את האות Kbd_DAT בגבוה, ו- Data! בנמוך
- ok מציין את הסיגנל parity_ok במצב true
- Error מציין את הסיגנל parity_ok במצב false
- counter מונה את מספר הביטים של קוד המקש שמגיעים בקו הסריאלי

הדרכה ודרישות:

כתוב קוד ב- VHDL המתאר את מכונת המצבים באמצעות תהליך סינכרוני בלבד. פתח את הקובץ bitrec.vhd מתוךהפרויקט הקיים KBDINTF והגדר אותו כהיררכיה עליונה. הוסף לקובץ bitrec.vhd את הקוד שלך בלבד בהתאם להנחיות להלן.

שים לב: אין צורך לשנות חלקים אחרים משלד הקוד הנתון ב- bitrec.vhd!

חשב מהו NUM_OF_BITS.
תשובה: 11 אחד לSTART אחד לSTOP ו9 לDATA

בטבלה הבאה מפורטים המצבים שבמכונה והפעולות לביצוע בכל מצב.
<u>מלא את העמודה האחרונה בטבלה</u> לפי הדוגמה שבשורה הראשונה:

שם המצב	פעילות עיקרית	לאיזה מצב עוברים מהמצב הנוכחי ובאילו תנאים – למלא את התאים הריקים
Idle	מאפסים את המונה count. ממתינים לתו חדש: אם יש ירידה באות השעון Kbd_CLK וגם באות הנתונים Kbd_DAT אז עוברים למצב הבא.	**עוברים ל-** LowClk **אם** ירידה בשעוון Kbd_CLK וגם ירידה ב- Kbd_DAT (סימן שמתחיל להגיע תו חדש)
LowClk	ממתינים לאות שעון גבוה כי זה אומר שהביט הבא כבר הגיע. אם Kbd_CLK גבוה: - משרשרים למקום האחרון ברגיסטר ההזזה shift_reg את הסיבית החדשה שהגיעה מה- Kbd_DAT. shift_reg <= kbd_dat & shift_reg(9 downto 1); - מקדמים את המונה count ב-1 - ובודקים אם הגיעו כל הביטים. אם כן עוברים למצב בדיקת הנתונים אם לא מחכים לירידת השעון הבא כדי להמשיך לקבל ביטים.	**עוברים ל-**HiClk **אם** ירידת שעון והמונה קטן ממספר הביטים **עוברים ל-**ChkData **אם** – ירידת שעון והמונה שווה למספר הביטים
HiClk	ממתינים לביט הבא. אם מגיע ביט, מסומן ע״י ירידה ב- Kbd_CLK עוברים למצב הבא, קבלת הביט.	**עוברים ל-**LowClk **אם** ירידת שעון
ChkData	בודקים את נכונות הנתונים ובהתאם לתוצאת הבדיקה עוברים למצב הבא. רק אם בדיקת הזוגיות (ה- parity) טובה מעדכנים את המוצא בתכולת הרגיסטר dout <= shift_reg(7 downto 0);	**עוברים ל-** NewData **אם** parity = 1 **אחרת עוברים ל-** Idle
NewData	מודיעים על מילה חדשה dout_new <= '1'; ועוברים מצב	**עוברים ל-**idle ללא תנאי

<u>בצע קומפילציה.</u>

צרף את הקוד של BITREC הכולל את מכונת המצבים המלאה:

```
entity bitrec is
   port ( resetN        : in  std_logic ;
          clk           : in  std_logic ;
          kbd_clk       : in  std_logic ;
          kbd_dat       : in  std_logic ;
          dout_new      : out std_logic ;
          dout          : out std_logic_vector(7 downto 0) ) ;
end bitrec ;

architecture arc_bitrec of bitrec is
   signal shift_reg : std_logic_vector(9 downto 0) ;
   signal parity_ok : std_logic                    ;
   type state is (idle , --initial state
          HighClk,
          LowClk,
          ChkData,
          NewData );
   constant numOfBits : integer  := 11 ;
begin
    parity_ok <=  shift_reg(8) -- same as kbd_dat
                 xor shift_reg(7) xor shift_reg(6)
                 xor shift_reg(5) xor shift_reg(4)
                 xor shift_reg(3) xor shift_reg(2)
                 xor shift_reg(2) xor shift_reg(0) ;

process ( resetN , clk )

  variable present_state : state;
  variable count : integer range 0 to 15;

begin

-- END OF DO NOT CHANGE PART ---------------------------

   ---- ASYNC PART ----
    if resetN = '0' then
      dout_new <= '0';
      count := 0 ;
      present_state := idle;
     ---- SYNCHRONOUS PART ----
    elsif rising_edge (clk) then

      ---- DEFAULT PART ----
        dout_new <= '0';
      ---- State Machine ----
        case present_state is

         when idle =>
            if kbd_dat = '0' and kbd_clk = '0' then
               present_state := LowClk;
            end if;
         when LowClk =>
         if kbd_clk = '1' then
            shift_reg <= kbd_dat & shift_reg(9 downto 1);
            count := count + 1;
            if count < numOfBits then
               present_state := HighClk;
            else
               present_state := ChkData;
            end if;
         end if;
         when HighClk =>
            if kbd_clk = '0' then
               present_state := LowClk;
            end if;
         when ChkData =>
            if parity_ok = '1' then
               present_state := NewData;
               dout_new <= '1';
               dout <= shift_reg(7 downto 0);
            else
               present_state := idle;
               dout_new <= '0';
               count := 0;
            end if;
         when NewData =>
            present_state := idle;
            dout_new <= '0';
            count := 0 ;
         end case;
   end if;
end process;
end architecture;
```

צרף לכאן צילום מסך של תוצאות קומפילציה מוצלחת של המעגל.

Flow Status	Successful - Wed Aug 15 12:03:12 2018
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DEBUG_STUDENTS
Top-level Entity Name	bitrec
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	14 / 56,480 (< 1 %)
Total registers	28
Total pins	13 / 268 (5 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
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 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

צור SYMBOL של קובץ זה אחרי קומפילציה מוצלחת.

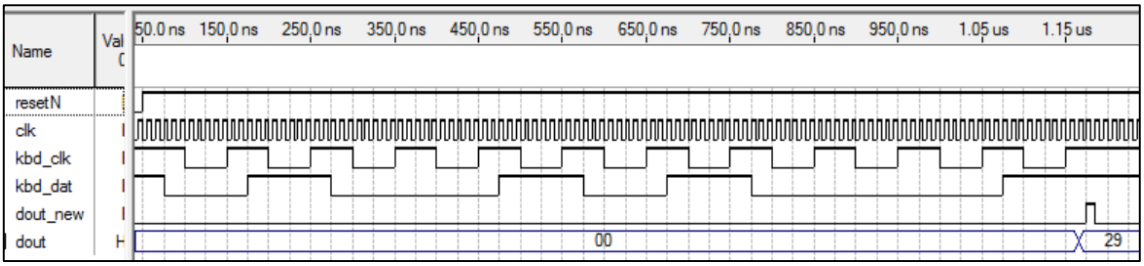
2.2 סימולציה

בצע סימולציה ב- Quartus כדי לדבג את מכונת המצבים שתכננת.

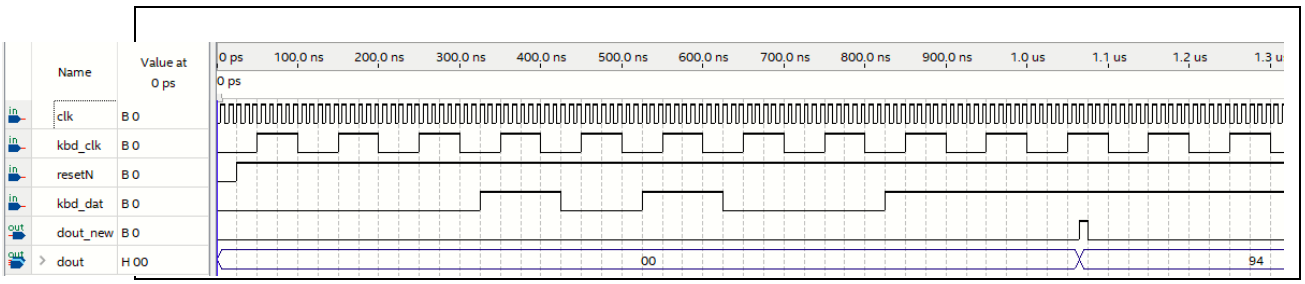
הדרכה לסימולציה: מומלץ להגדיר את **שעון** המערכת (clk) מהיר פי 10 משעון המקלדת (Kbd_CLK) : למשל, קבע בשעון המערכת period=10nsec ובשעון המקלדת period=100nsec. השתמש בגודל **grid** של 25 nsec ושים לב שהשינוי ב- Kbd_DAT מתרחש בזמנים ששעון המקלדת ב- '1' לוגי!

הראה שבסימולציה שלך התוצאות זהות לדוגמה הנתונה להלן. הראה שאם מכניסים רצף טורי של קוד מקש נתון ב- Kbd_DAT, למשל **29H הקוד של מקש הרווח**, מתקבל ב- dout 29H מקבילי (הצג אות זה ב- radix hexadecimal) ומתקבל '1' במשך מחזור שעון אחד שמודיע על מקש חדש ב- dout_new אחרי שה- clk האחרון הסתיים (אחרי ה- Stop bit).

חשוב מאד: לביצוע הסימולציה יש להזין אך ורק את אות המבוא KBD_DAT כפי שנתון בדוגמה להלן ובקובץ הנתון לכם במודל!



צרף לכאן צילום מסך של תוצאות סימולציה מוצלחת.



3 חישוב עומק הזכרון עבור הנתח הלוגי

רקע למטלה: על מנת לדבג את המערכת רוצים לדגום באמצעות הנתח הלוגי את אות המבוא Kbd_DAT של יחידת ה- BITREC בזמן הקשה על מקש כלשהו.

ברוב המקשים קוד המקש מכיל 11 סיביות, אך במקשים מהסוג החדש, הקוד מכיל 11 סיביות נוספות ומחזור שעון הפסקה (למשל הקוד של מקש Down Arrow מהסוג החדש הוא (72 E0). כמו כן, שעון המקלדת Kbd_CLK, שמשמש לסנכרון סיביות הנתונים של Kbd_DAT, עובד בתדר של 12.5 KHz. **לביצוע החישוב היעזר בהסבר המפורט מחומר הרקע**.

חשב מה צריך להיות עומק הזכרון המינימלי בנתח הלוגי הדרוש לקליטת כל הקוד במקרה זה.
חישוב ותשובה: (8X10^-5)*(50X10^6)*(23)=92k
ולכן נצטרך 128K

4 מטלת תכן עם מקלדת (זיהוי מקש SHIFT שמאלי)

רקע למטלה: יישומים המשתמשים במקלדות בדרך כלל מקצים למקשים תפקידים מיוחדים. כמו מקש SHIFT שבוחר את האותיות הגדולות (UpperCase) באנגלית.

בחומר העזר **נתונה דוגמא** ב-VHDL למימוש ישום המשתמש במקש ה- CapsLock להחלפה אחת של מצבו של לד בלוח DE10, בין הדלקה לכיבוי.
ממש מערכת שמבצעת פעולה דומה למקש SHIFT שמאלי, (קוד מקש 12H) **בעזרת מכונת מצבים**.
כל לחיצה על מקש זה (גם אם מדובר בלחיצה ארוכה) תגרום להחלפה אחת בלבד של מצב הנורית בלוח DE10.

הערה חשובה: הקבוע ("001011000") שהוא קוד המקש מופיע בגוף הקוד ולא כCONSTANT, עליכם לתקן שגיאה זו גם כן.

שים לב שיישום זה ישתמש במערכת הממשק למקלדת שיצרת קודם, ז"א אותות המבוא שלו הם אותות המוצא של הממשק למקלדת (מהמוצא של BYTEREC).

בהמשך, המערכת תידרש לתמוך במקש כלשהוא לא רק בבמקש SHIFT שמאלי ולכן עליך לאפשר הזנה של קוד המקש המבוקש ממודול חיצוני, כגון רכיב LPM_CONSTANT (אותו תחבר למערכת במעבדה).

הדרכה ודרישות:

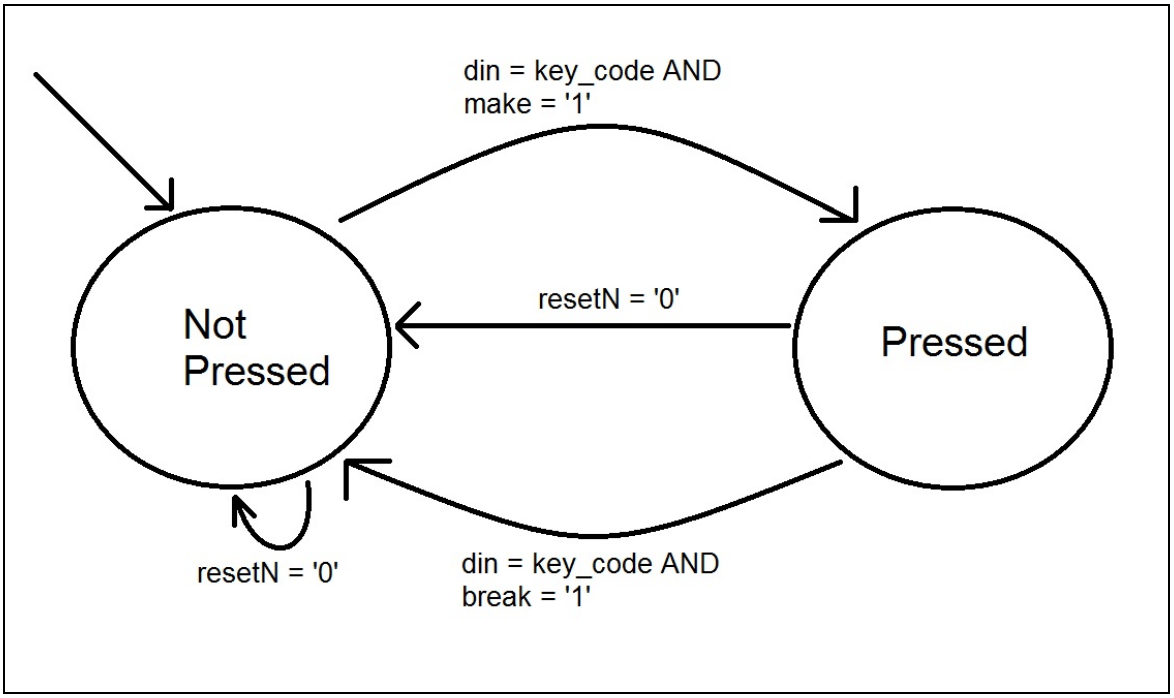
פתח קובץ VHDL חדש ב – Quartus, באותו פרויקט KBDINTF. קרא לקובץ בשם LEFT_SHIFT.VHD. וודא שהוא שייך לפרויקט והוא מוגדר **כהיררכיה עליונה**.

כתוב את הקוד של הישום בשפת VHDL בקובץ שפתחת. תוכל להיעזר בדוגמת הקוד לישום כזה הנתונה בנספח של חומר הרקע לניסוי.

בשונה מהדוגמה הנתונה, בתוכנה שלך הגדר **וקטור כניסה in** בשם key_code באורך 8 סיביות, שיקבל את קוד המקש הרצוי.

בשונה מהדוגמה הנתונה, יש לתכנן את היישות בעזרת **מכונת מצבים**.

צרף לכאן דיאגרמת מצבים עליה התבססת למימוש היישות.



<u>שים לב ששם ה- ENTITY צריך להיות זהה לשם הקובץ.</u>

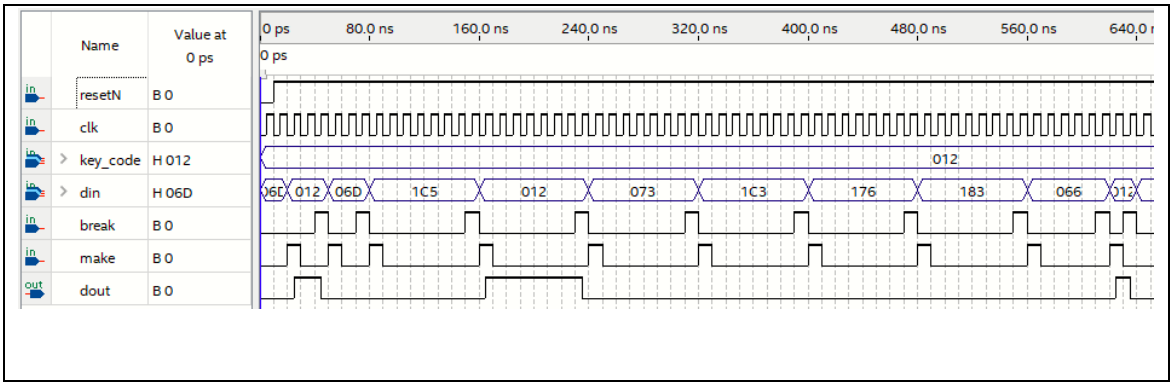
<u>קמפל</u> את הקוד. אחרי שהקומפילציה עברה בהצלחה, **<u>צור</u>** SYMBOL של קובץ זה.

<u>בצע סימולציה</u> והראה שהיישום עובד כמתוכנן. הראה שבכל לחיצה על מקש SHIFT שמאלי (קוד 12H) יש שינוי אחד במצב הנורית led_out (מ- '0' ל- '1' ואחר כך מ- '1' ל- '0'). לשם כך הזן ל- din ול- key_code את אותו מספר, 12H.

צרף את קטע הקוד הרלוונטי להדלקה/כיבוי לד בכל הקשה על מקש ה- SHIFT שמאלי **:**

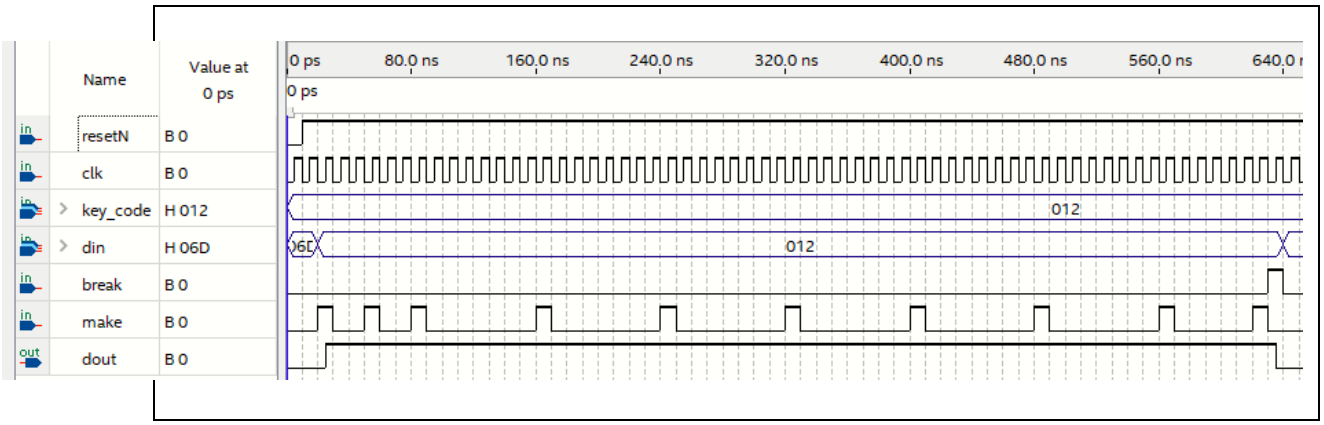
```
architecture arc_LEFT_SHIFT of LEFT_SHIFT is
 type press_state is(pressed, not_pressed);
     signal state: press_state;
     signal out_led: std_logic;

 begin

  dout <= '1' when state = pressed else '0';

    process ( resetN , clk)
     begin
     if resetN = '0' then
        state <= not_pressed ;
     elsif rising_edge(clk) then
       if (din = key_code) and (make = '1' )  then
          state <= pressed;
       elsif (din = key_code) and (break = '1' ) then
          state <= not_pressed;
       end if;
    end if;
 end process;
 end architecture arc_LEFT_SHIFT;
```

צרף לכאן צילום מסך של תוצאות סימולציה מוצלחת.



בצע סימולציה נוספת והראה שבלחיצה ארוכה על מקש ה- SHIFT שמאלי, יש רק שינוי אחד במצב הנורית. לחיצה ארוכה על מקש מדמים על ידי מספר פולסי make רצופים (הראה לפחות שנים) ללא פולס break ביניהם.

צרף לכאן צילום מסך של תוצאות סימולציה מוצלחת.



על הקוד שכתבתם יש לבצע פעולת ARCHIVE ב QUARTUS (כמתואר בפרק 16 של quartus 17 cook book במודל).
את הקובץ המכווץ שתקבלו מפעולה זו יש להעלות במודל למקום המתאים

QAR דחוס של DEBUG- דוח הכנה

5 פרויקט

(לא חלק מהציון של דו"ח זה)

נושא הפרוייקט	

5.1 סכמת מלבנים

הוסף סכמת מלבנים עיקרית של הפרוייקט – 5-10 מלבנים משמעותיים

5.2 רשימת תהליכים (מלבנים) עיקרית

רשום את כל הרכיבים (תהליכים) העיקריים בפרוייקט, לכל רכיב רשום את תפקידיו ואת הכניסות והיציאות העיקריות

שם	תאור	כניסות	יציאות
.1			
.2			
.3			
.4			

5.3 סיפתח (אִסְתִּפְתָּאח = إستفتاح)

הגדר מהו החלק שתממש כסיפתח לפרוייקט,

רשום את כל התליכים (מלבנים) העיקריים בסיפתח , לכל רכיב רשום את תפקידיו ואת הכניסות והיציאות העיקריות

שם	תאור	כניסות	יציאות
.1			
.2			
.3			
.4			

שרטט את ההיראכיה העליונה של הסיפתח – אין צורך לממש

לאחר שסיימת - לחץ על ה LINK ומלא בבקשה את השאלון המצורף

מלא את הטופס