



מעבדות 1, ח1

## ניפוי תקלות בחומרה (DEBUG)

דו"ח מכין - שאלות ותרגילי הכנה

הניסוי פותח בחסות המעבדה למערכות ספרתיות מהירות 

גרסה 1.0 (אביב 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 | 12/5/18 |
| שם המדריך             | דודי    |

| סטודנט | שם פרטי | שם משפחה |
|--------|---------|----------|
| 1      | חגי     | קריטי    |
| 2      | יהונתן  | סרלואי   |

# תוכן עניינים של דו"ח מכין ניפוי תקלות בחומרה (DEBUG)

|    |       |  |     |
|----|-------|--|-----|
| 2  | ..... | מכונת RANDOM   | 1   |
| 3  | ..... | ממשק למקלדת  | 2   |
| 3  | ..... | תכן יחידת ה- BITREC  | 2.1 |
| 8  | ..... | סימולציה   | 2.2 |
| 8  | ..... | חישוב עומק הזכרון עבור הנתח הלוגי  | 3   |
| 9  | ..... | מטלת תכן עם מקלדת (זיהוי NUMLock)  | 4   |
| 13 | ..... | פרויקט   | 5   |
| 13 | ..... | סכמת מלבנים  | 5.1 |
| 13 | ..... | רשימת תהליכים (מלבנים) עיקרית  | 5.2 |
| 14 | ..... | סיפתח (אסתפתאח =  ) ( | 5.3 |

## 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: מכיוון שה- count8 סופר מהר מאוד ביחס (50MHz) לקצב הלחיצה על מקש key0 כל לחיצה עליו תוציא מספר אקראי כלשהו בין 255 ל-0

ב. מהו תפקידה של היחידה vrise?

תשובה: להוציא '1' בכל פעם שיש עליה ב-pulse ולאפשר שמירה ב-regout

ג. למה משמש הקבוע MAXCOUNT?

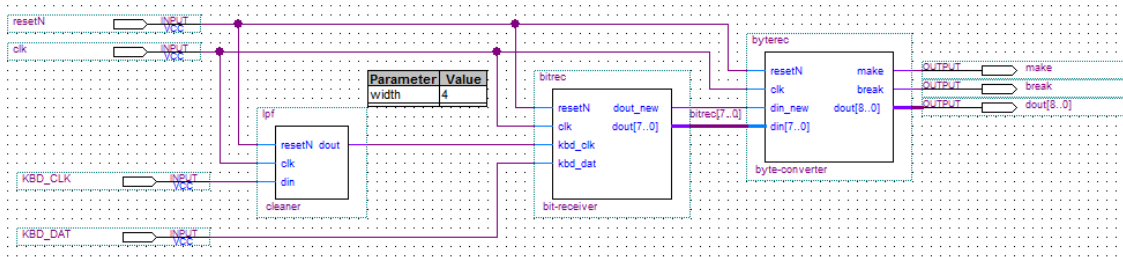
תשובה: לטעינה חדשה של ה-counter

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

תשובה: ניתן לאפס את ה-LSB וכך נקבל מספרים זוגיים בלבד

## 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 לבניית הממשק שלך למקלדת במעבדה זו. להלן הקבצים שבהם תעשה שימוש:

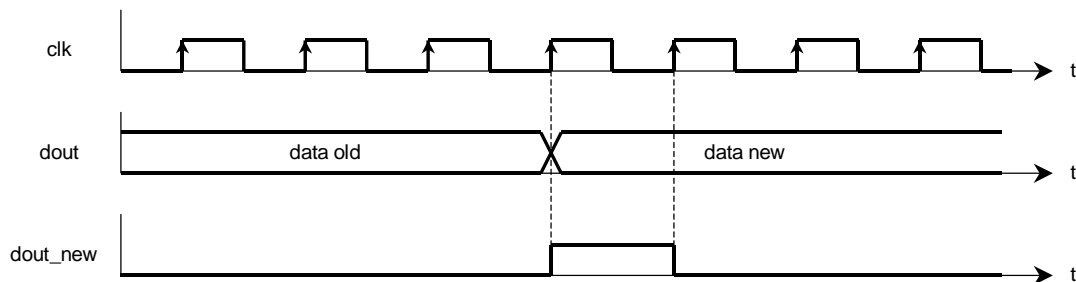
- ☐ יחידת מסנן מעביר נמוכים : lpf.vhd, lpf.bsf – נתונה לך במודל
- ☐ יחידת המקלט ברמת ה Bit : bitrec.vhd – נתון שלד שלה במודל
- ☐ יחידת המקלט ברמת ה Byte : byterec.vhd, byterec.bsf – נתונה לך במודל

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

צור פרויקט חדש בשם KBDINTF, העתק אליו את הקבצים הנ"ל 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:



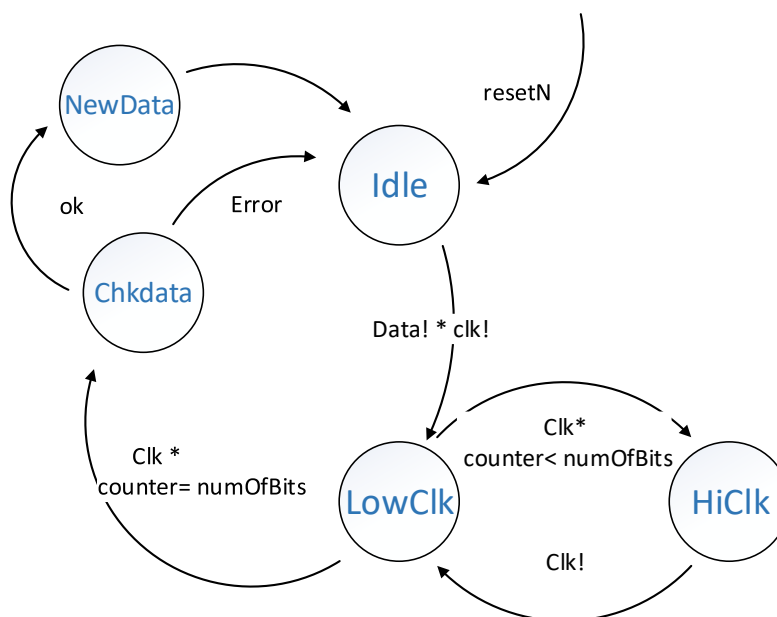
נתון לך הקובץ bitrec.vhd שהוא שלד המכיל את כל החלקים הדרושים כפי שהוסבר בחומר הרקע פרט למכונת המצבים.

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

הוסף לקובץ זה את הקוד של מכונת המצבים, כפי שתואר 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 תוך הוספה שלו לפרויקט 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:** 8databits + 1paritybit + 1startbit + 1stopbit = 11bits

בטבלה הבאה מפורטים המצבים שבמכונה והפעולות לביצוע בכל מצב.  
**מלא את העמודה האחרונה בטבלה 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 עם ירידה בשעון Kbd_CLK וגם ירידה ב- Kbd_DAT (סימן שמתחיל להגיע תו חדש) |
| LowClk  | ממתינים לאות שעון גבוה כי זה אומר שהביט הבא כבר הגיע.   | עוברים ל- HighClk אם Kbd_Clk גבוה וגם   |

|  |   |         |
|--|---|---------|
| <b>count&lt;numOfBits</b><br>עוברים ל-ChkData אם Kbd_Clk גבוה וגם<br>count = numOfBits | אם Kbd_CLK גבוה :<br>- משרשרים למקום האחרון ברגיסטר<br>ההזזה shift_reg את הסיבית החדשה<br>שהגיעה מה-Kbd_DAT.<br>shift_reg <= kbd_dat &<br>shift_reg(9 downto 1);<br>- מקדמים את המונה count ב-1<br>- ובודקים אם הגיעו כל הביטים. אם כן<br>עוברים למצב בדיקת הנתונים אם לא<br>מחכים לירידת השעון הבא כדי<br>להמשיך לקבל ביטים. |         |
| <b>עוברים ל-LowClk</b><br><b>אם kbd_clk נמוך</b>                                       | ממתינים לביט הבא. אם מגיע ביט,<br>מסומן ע"י ירידה ב-Kbd_CLK<br>עוברים למצב הבא, קבלת הביט.  | HiClk   |
| <b>עוברים ל-NewData</b><br><b>אם parity גבוה</b><br>עוברים ל-idle<br>אם parity נמוך    | בודקים את נכונות הנתונים ובהתאם<br>לתוצאת הבדיקה עוברים למצב הבא.<br>רק אם בדיקת הזוגיות (ה-parity)<br>טובה מעדכנים את המוצא בתכולת<br>הרגיסטר<br>dout <= shift_reg(7 downto 0);  | ChkData |
| <b>עוברים ל-idle</b><br><b>עם עליית שעון</b>   | מודיעים על מילה חדשה<br>dout_new <= '1';<br>ועוברים מצב   | NewData |

#### בצע קומפילציה.

|  |
|--|
| <p align="center"><b>צרף את הקוד של BITREC הכולל את מכונת המצבים המלאה:</b></p> <pre> library ieee ; use ieee.std_logic_1164.all ; -- simplified 5 states bitrec -- Dudy Bar On, lab 1 Dept of EE -- Copyright technion IIT 2016  -- DO NOT CHANGE IN THIS PART OF CODE ----- 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 </pre> |
|--|

```

        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 =>
                ---- WRITE YOUR CODE HERE ----
                count := 0;
                if(kbd_dat = '0' and kbd_clk = '0') then
                    present_state := LowClk;
                end if;
            -----

            when LowClk =>
                ---- WRITE YOUR CODE HERE ----
                if(kbd_clk = '1') then
                    count:= count + 1;
                    shift_reg <= kbd_dat & shift_reg(9 downto 1);
                    if(count < numOfBits) then
                        present_state := HighClk;
                    elsif(count = numOfBits) then
                        present_state := ChkData;
                    end if;
                end if;
            -----

            when HighClk =>
                ---- WRITE YOUR CODE HERE ----
                if(kbd_clk = '0') then
                    present_state := LowClk;
                end if;
            -----
        end case;
    end if;
end process;

```

```

when ChkData =>
    ---- WRITE YOUR CODE HERE ----
    if (parity_ok = '0') then
        present_state := idle;
    else
        dout <= shift_reg(7 downto 0);
        present_state := NewData;
    end if;
    -----
when NewData =>
    ---- WRITE YOUR CODE HERE ----
    dout_new <= '1';
    present_state := idle;
    -----
end case;
end if;
end process;
end architecture;

```

צריך לכאן צילום מסך של תוצאות קומפילציה מוצלחת של המעגל.

|                                 |   |
|---------------------------------|---|
| Flow Status                     | Successful - Sat May 12 16:49:13 2018       |
| Quartus Prime Version           | 17.1.0 Build 590 10/25/2017 SJ Lite Edition |
| Revision Name                   | KBDINTF                                     |
| Top-level Entity Name           | bitrec                                      |
| Family                          | Cyclone V                                   |
| Device                          | 5CSXFC6D6F31C6                              |
| Timing Models                   | Final                                       |
| Logic utilization (in ALMs)     | N/A   |
| Total registers                 | 28  |
| Total pins                      | 13  |
| Total virtual pins              | 0   |
| Total block memory bits         | 0   |
| Total DSP Blocks                | 0   |
| Total HSSI RX PCSs              | 0   |
| Total HSSI PMA RX Deserializers | 0   |
| Total HSSI TX PCSs              | 0   |
| Total HSSI PMA TX Serializers   | 0   |
| Total PLLs                      | 0   |
| Total DLLs                      | 0   |

**צור SYMBOL** של קובץ זה אחרי קומפילציה מוצלחת.

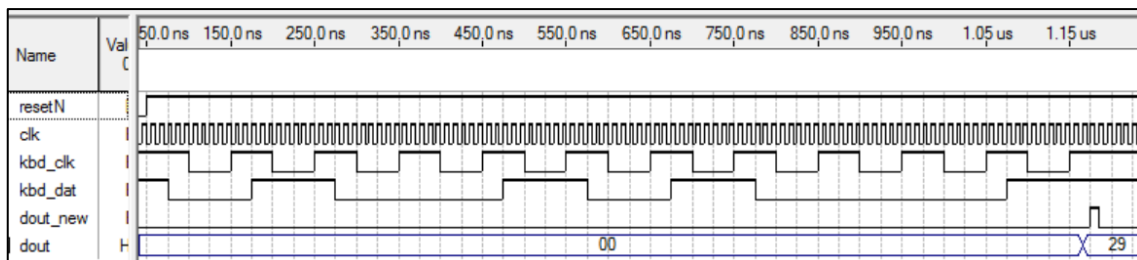
## 2.2 סימולציה

**בצע סימולציה** ב- Quartus כדי לדבג את מכונת המצבים שתכנתת.

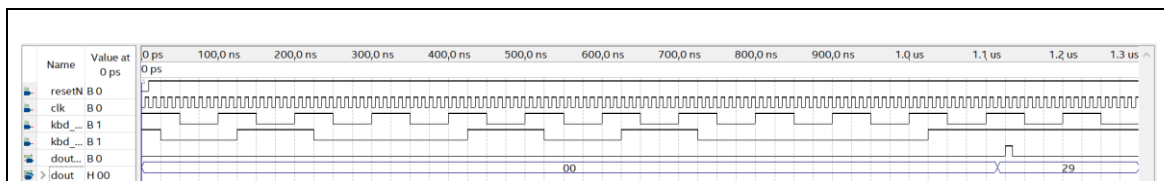
**הדרכה לסימולציה:** מומלץ להגדיר את **שעון** המערכת (clk) מהיר פי 10 משעון המקלדת (Kbd\_CLK): למשל, קבע בשעון המערכת period=10nsec ובשעון המקלדת period=100nsec. השתמש בגודל **grid** של 25 nsec ושים לב שהשינוי ב- Kbd\_DAT מתרחש בזמנים ששעון המקלדת ב- '1' לוגי!

**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 מאוד: לביצוע הסימולציה יש להזין אך ורק את אות המבוא KBD\_DAT כפי שנתון בדוגמה להלן ובקובץ הנתון לכם במודל!**



צריך 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:

$$\text{NumOfBits} * \text{Time} * \text{Frequency} = \text{Memory}$$
$$(11+1+11) * (1/12.5K) * 50M = 92K$$

לכן נבחר 128K ע"פ האפשרויות מהחומר רקע.

## 4 מטלת תכן עם מקלדת (זיהוי NUMLock)

**רקע למטלה:** יישומים המשתמשים במקלדות בדרך כלל מקצים למקשים תפקידים מיוחדים של החלפת תפקידיהם של מקשים אחרים במקלדת, כמו מקש ה-NumLock. מקש זה גורם בדרך כלל להחלפה של מספרים וחיצים. כל הקשה עליו, גורמת להחלפה אחת בלבד של המצב, גם כאשר מדובר בלחיצה ארוכה.

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 NumLock, **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") מופיע בגוף הקוד ולא כ-CONSTANT, 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 ב- NUMLOCK ולכן עליך לאפשר הזנה של קוד המקש המבוקש ממודול חיצוני, כגון רכיב LPM\_CONSTANT (אותו תחבר למערכת במעבדה).

### הדרכה ודרישות:

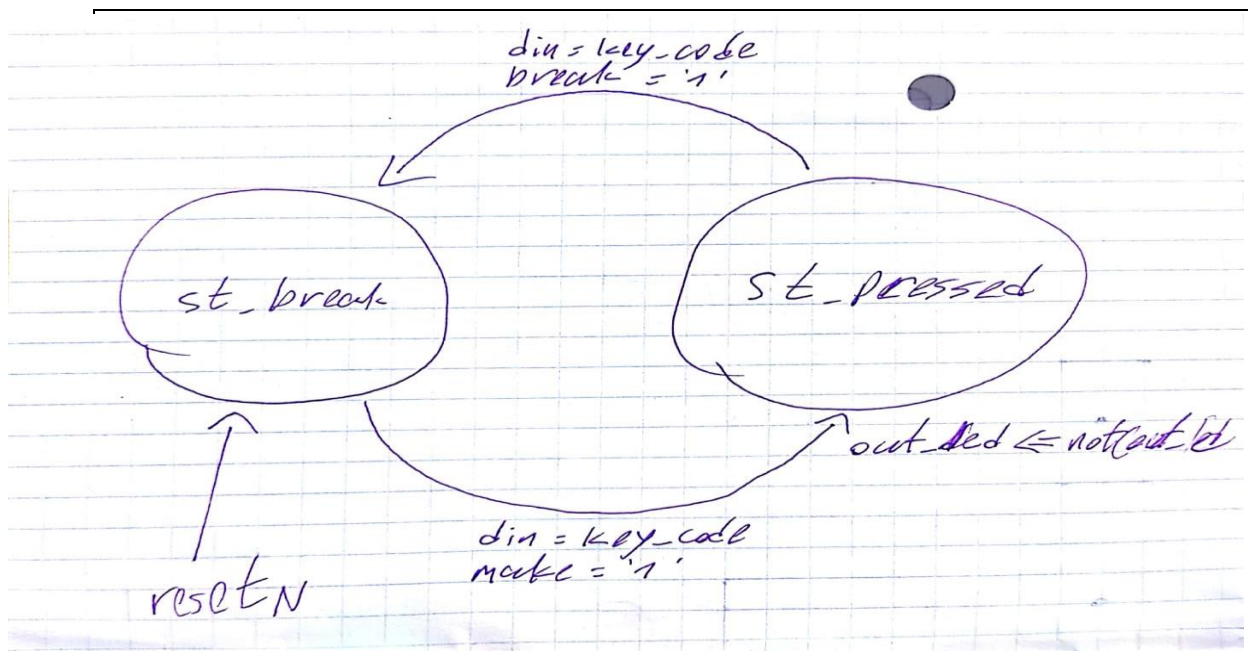
**פתח** קובץ VHDL חדש ב- Quartus, באותו פרויקט KBDINTF. קרא לקובץ בשם NUM\_LOCK.VHD וודא שהוא שייך לפרויקט והוא מוגדר **בהיררכיה עליונה**.

**כתוב** את הקוד של ישום ה-NumLock בשפת VHDL בקובץ שפתחת. תוכל להיעזר בדוגמת הקוד ליישום 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**.

צריך לכאן דיאגרמת מצבים עליה התבססת למימוש היישות.



**שם לב ששם ה- ENTITY צריך להיות זהה לשם הקובץ.**

**קמפל** את הקוד. אחרי שהקומפילציה עברה בהצלחה, **צור** SYMBOL של קובץ זה.

**בצע סימולציה** והראה שהיישום עובד כמתוכנן. הראה שבכל לחיצה על מקש ה- NumLock (קוד 77H) 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, 77H.

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

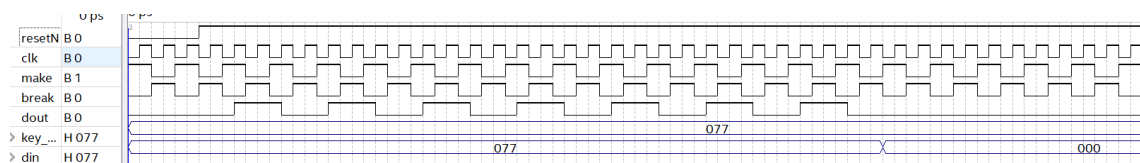
```

entity NUM_LOCK is
port( resetN : in std_logic;
      clk : in std_logic;
      din : in std_logic_vector(8 downto 0);
      key_code : in std_logic_vector(8 downto 0);
      make : in std_logic;
      break : in std_logic;
      dout : out std_logic);
end NUM_LOCK;

architecture behavior of NUM_LOCK is
    signal out_led : std_logic;
    type state is (st_pressed, st_break);
begin
    dout <= out_led;
    process(resetN, clk)
        variable pr_state : state;
    begin
        if resetN = '0' then
            out_led <= '0';
            pr_state := st_break;
        elsif rising_edge(clk) then
            case pr_state is
                when st_break =>
                    if(din = key_code) and (make = '1') then
                        out_led <= not(out_led);
                        pr_state := st_pressed;
                    end if;
                when st_pressed =>
                    if(din = key_code) and (break = '1') then
                        pr_state := st_break;
                    end if;
            end case;
        end if;
    end process;
end architecture;

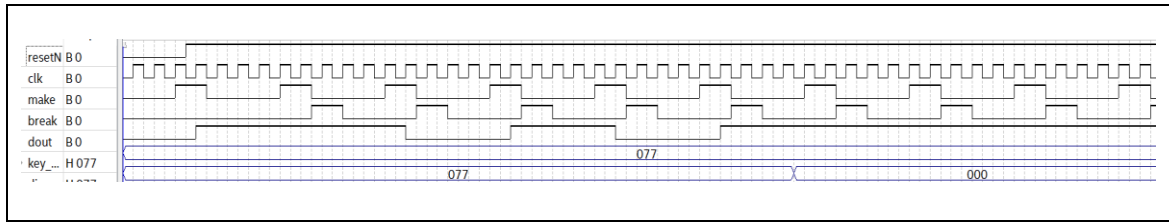
```

צורף לכאן צילום מסך של תוצאות סימולציה מוצלחת.



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

צורף לכאן צילום מסך של תוצאות סימולציה מוצלחת.



## 5 פרויקט

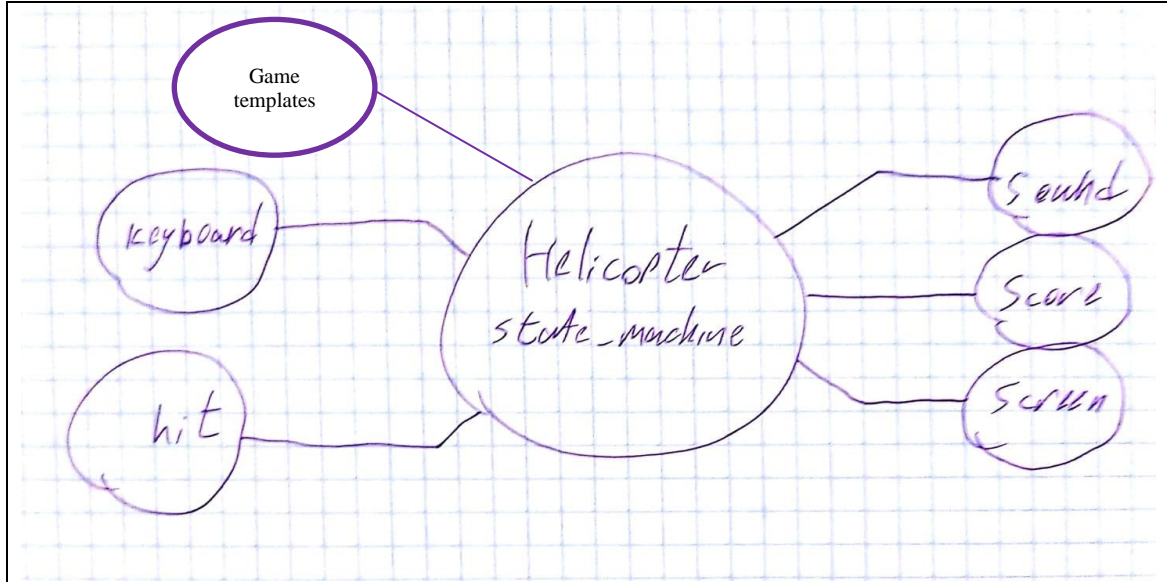
(לא חלק מהציון של דו"ח זה)

נושא הפרויקט

Helicopter

### 5.1 סכמת מלבנים

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



### 5.2 רשימת תהליכים (מלבנים) עיקרית

רשום את כל הרכיבים (תהליכים) העיקריים בפרויקט, לכל רכיב רשום את תפקידו ואת הכניסות והיציאות העיקריות

| שם                          | תאור  | כניסות   | יציאות                      |
|-----------------------------|---|--|-----------------------------|
| 1. Keyboard                 | ממשק של המקלדת למשחק והגדרת תפעול כללי של המקשים                | ממשק למקלדת מחשב   | Make, brake, key_data_out   |
| 2. Game_templates           | הגדרת מספר טמפליטים של מסך המשחק בזכרון                         | State_machine  |                             |
| 3. hit                      | תצורת מסך נוכחית דינאמית הכוללת את תזוזת חלקי המשחק             | State_machine<br>Game_templates                                    | State_machine               |
| 4. Helicopter_state_machine | הגדרת מעבר בין מצבי המשחק השונים (התחלה וסיום משחק, פסילה וכו') | חיווי מהמקלדת ותצורת המסך הנוכחית (או מטמפלייט בתחילת ובסוף המשחק) | Sound, score, present_state |
| 5. sound                    | מימוש התנהגות הצלילים במשחק                                     | State_machine  | רמקול                       |
| 6. score                    | תצוגת הניקוד במשחק  | State_machine  | 7seg                        |
| 7. screen                   | תצוגת המשחק   | State_machine<br>Game_templates                                    | VGA                         |

### 5.3 סיפתח (אסתפתח = استفتاح )

הגדר מהו החלק שתממש כסיפתח לפרויקט,

תצוגה המכילה מסך פשוט והוספת מסוק במיקום קבוע בלחיצה של לחצן ספציפי

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

| שם                | תאור | כניסות | יציאות |
|-------------------|------|--------|--------|
| 1. מסך            |      |        |        |
| 2. מקלדת          |      |        |        |
| 3. Game_templates |      |        |        |
| 4. State_machine  |      |        |        |

שרטט את ההירארכיה העליונה של הסיפתח – אין צורך לממש

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

**מלא את הטופס**