הפקולטה להנדסת חשמל
ע"ש אנדרו וארנה ויטרבי

הטכניון
מכון טכנולוגי לישראל

מעבדה בהנדסת חשמל
044157 'א1

ניסוי DEBUG - ניפוי תקלות בחומרה

שאלות ודוח הכנה

גרסה 3.44
קיץ תשפ"ב 2022

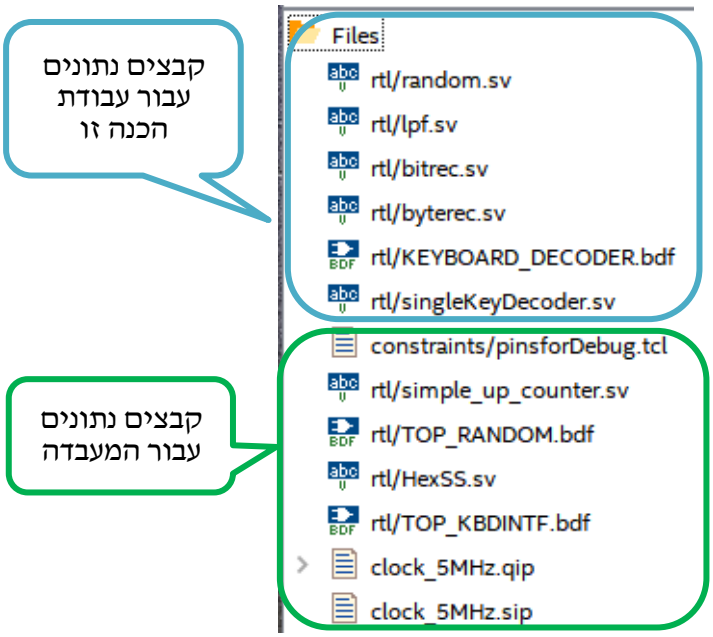
שם המדריך	תאריך ההגשה של דוח ההכנה

סטודנט	שם פרטי	שם משפחה
1	עמיחי	שטרנליכט
2	יקיר	לוגסי

תוכן עניינים של דו"ח הכנה DEBUG

1 פתיחת הקבצים לעבודה

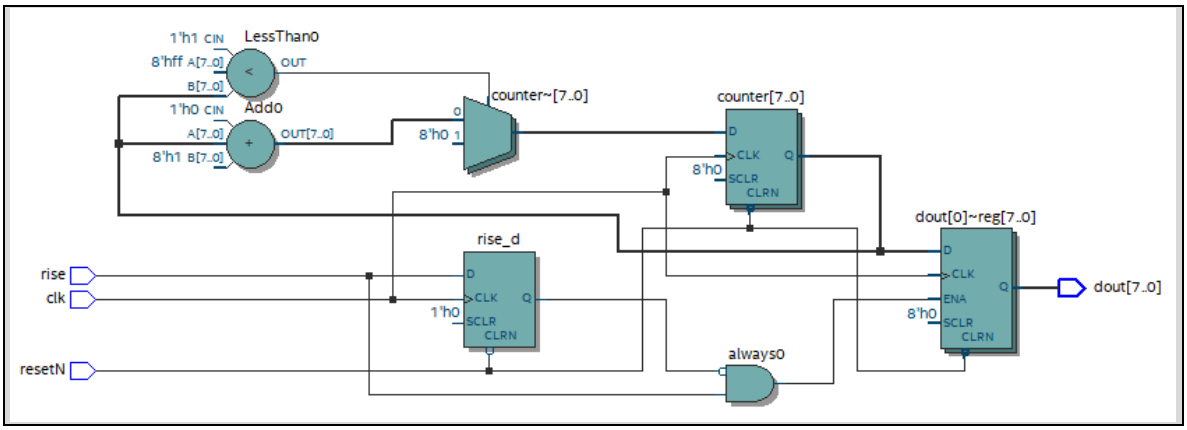
צור תיקיה למעבדה זו. הורד מהמודל קובץ ארכיב של המעבדה ופתח אותו לפרויקט בתיקייה שיצרת.
שים לב לשנות את ה PATH שמציע הקוורטוס ל- PATH קצר, שאינו מכיל עברית, רווחים ו/או את הסימן '–' .
ודא תכולת קבצים דומה לזו :



הקבצים המסומנים בכחול הם עבור עבודת הכנה זו.
הקבצים המסומנים בירוק הם עבור המעבדה. הם נתונים לך עכשיו כחלק מהפרויקט שתתחיל אותו כעת בעבודת ההכנה ותמשיך אותו במעבדה.

2 מכונת RANDOM

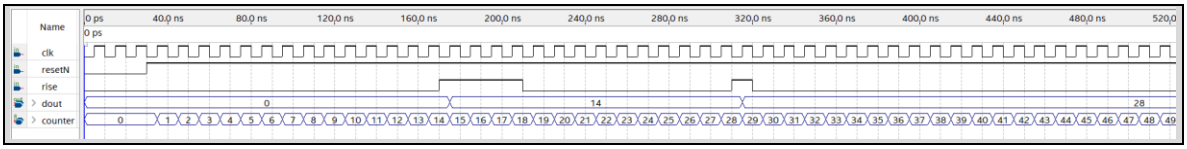
נתון לך קובץ random.sv, המממש מערכת שמייצרת מספרים בצורה אקראית.
<u>פתח</u> אותו ונסה להבין את פעולתו.
<u>הפוך</u> אותו ל-TOP והרץ אנליזה שלו.
<u>הצג את היצוג הגרפי</u> שלו כ- RTL VIEW (Tools -> Netlist Viewers -> RTL Viewer).



הסבר מדוע היציאה dout[7..0] היא מספר אקראי?
תשובה :
במערכת ישנו מונה מהיר ברוחב 7 סיביות (סופר מ-0 ועד 255 באופן ציקלי). כאשר הסיגנל rise (בשליטת המשתמש) עולה ל-1 לפחות למשך מחזור שעון, מוצא ה- always יהיה 1, מה שיאפשר את מעבר ערך המונה הנוכחי למוצא המערכת. מספר זה הוא אקראי כיוון שהמשתמש מעלה את rise בזמן שרירותי, ובגלל ששעון הכרטיס מהיר מאוד, המשתמש לא יכול לתזמן את הלחיצה כדי להשיג מספר שאינו אקראי.

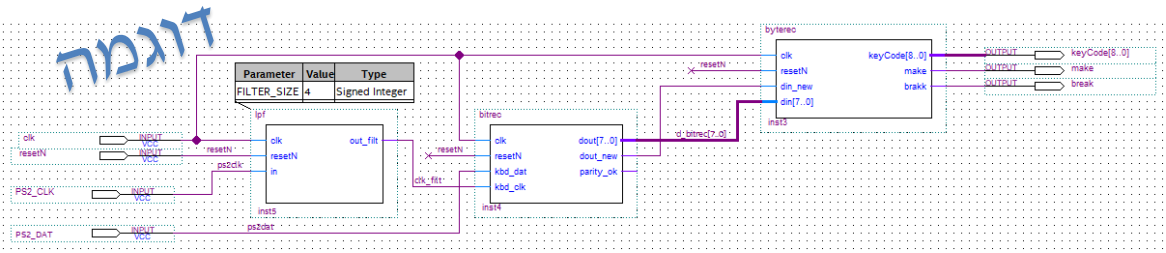
<u>כיצד ניתן לשנות</u> את המכונה כך שתגריל מספרים שהם כפולות של 2 בלבד?
תשובה:
ניתן לשנות את השורה שמגדילה את המונה באחד(29), להגדיל כל פעם את המונה ב2.
לחילופין נבחין כי עבור כל הפולות של 2 הLSB=0, לכן ניתן גם להגריל רק כפולות של 2 באמצעות קביעת הביט התחתון ב0 וספירה רק של שאר הביטים.

<u>הרץ סימולציה</u> לבדיקת הפעולה של המודול. **<u>הראה</u>** תוצאה אקראית בשני מקרים לפחות.
<u>הקפד</u> להציג בסימולציה גם את המונה הפנימי.



3 ממשק למקלדת

המטרה בתרגיל זה היא לבנות ממשק למקלדת שיאפשר חיבור מקלדת לכרטיס DE10. המקלדת תשמש בהמשך להפעלת הפרויקט.
כפי שהוסבר בחומר הרקע לניסוי זה, התכן הסינכרוני הבא נבחר למימוש **ממשק חומרה למקלדת**.

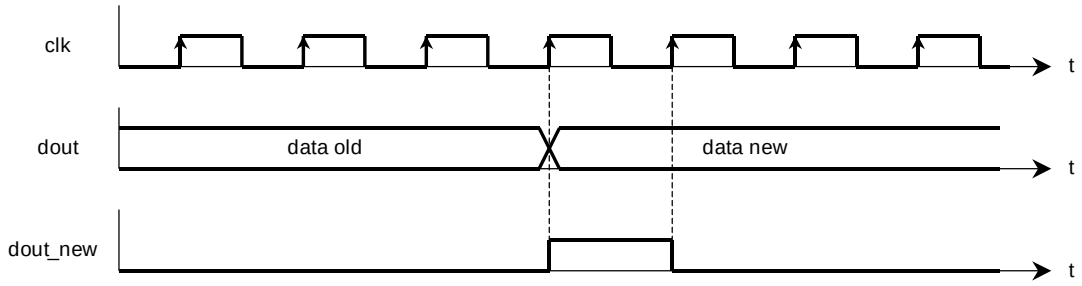


יחידות אלה כתובות בשפת SYS-VERILOG ותשמשנה לבנית הממשק למקלדת במעבדה.
להלן הקבצים הנתונים לך שמרכיבים את הממשק למקלדת:

- יחידת מסנן מעביר נמוכים: lpf.sv – נתונה
- יחידת המקלט ברמת ה – Bit: bitrec.sv - נתון שלד שלה
- יחידת המקלט ברמת ה – Byte: byterec.sv – נתונה

3.1 תכן יחידת ה - BITREC

רקע למטלה: כמו שהוסבר בחומר הרקע תפקידה של היחידה שמטפלת בתשדורת הטורית, ה-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 ונקראת dout_new. דיאגרמת הזמנים הבאה מתארת אותות אלו אחד ביחס לשני וביחס לאות השעון:

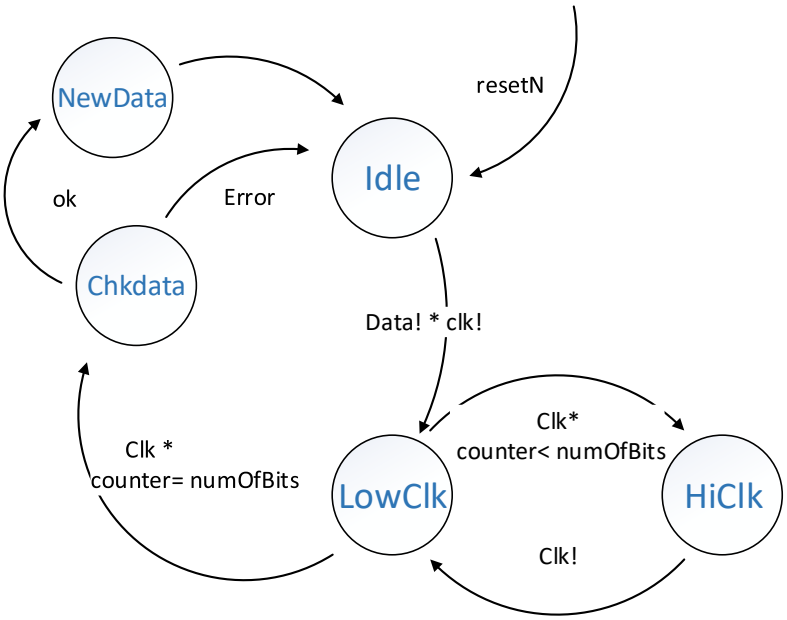


נתון לך הקובץ bitrec.sv שהוא שלד המכיל את כל החלקים הדרושים כפי שהוסבר בחומר הרקע פרט למכונת המצבים.

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 `fill your code please` `&&&&&&&&&&`

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 (ממופה לפין PS2_CLK) בגבוה, ו- clk! בנמוך
- Data מציין את האות kbd_dat (ממופה לפין PS2_DAT) בגבוה, ו- Data! בנמוך
- ok מציין את הסיגנל parity_ok במצב true (זה מה שצריך להיות)
- Error מציין את הסיגנל parity_ok במצב false
- counter מונה את מספר הביטים של קוד המקש שמגיעים בקו הטורי

הדרכה ודרישות:

כתוב קוד המתאר את מכונת המצבים באמצעות תהליך סינכרוני בלבד. פתח את הקובץ **bitrec.sv** מתוך הפרויקט הנוכחי והגדר אותו כהיררכיה עליונה. הוסף לקובץ את הקוד שלך בלבד בהתאם להנחיות להלן.

שימו לב: במכונה הוכנסה תקלה במכוון
אין צורך לשנות חלקים אחרים משלד הקוד הנתון ב- bitrec.sv!
אם מצאתם את התקלה- אנא אל תדווחו עליה בפורום שאלות ותשובות
וגם אל תספרו לחבריכם, השאירו להם את חוויית הגילוי העצמי !

מהו NUM_OF_BITS? (לשים לב לגודל הרגיסטר שמקבל את המידע הטורי) הוסף אותו לקוד במקום המתאים.
תשובה:
9 : 8 ביטי מידע + ביט PARITY.

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

הערות:
- פעולה של המתנה לאות מסויים או ספירה עד ערך מסויים מתורגמת לקוד באמצעות משפטי IF
- אם אין לצאת ממצב מסויים לא צריך לעשות פעולת סרק
- היעזרו בדיאגרמת המצבים, בה מופיעים תנאי המעברים, להשלמה של עמודת המעברים בטבלה הבאה

שם המצב	פעילות עיקרית	לאיזה מצב עוברים מהמצב הנוכחי ובאילו תנאים – למלא את התאים הריקים
Idle	זהו מצב ההמתנה למילה חדשה. כאן - מאפסים את המונה count. - ממתינים למילה חדשה (סימן למילה חדשה – ירידה באותות השעון והנתונים)	**עוברים ל-** LowClk **עם** ירידה באות השעון kbd_clk וגם ירידה באות kbd_dat
LowClk	זהו מצב קבלת ביט. במצב זה ממתינים לאות שעון גבוה המסמן שהביט הבא מגיע. אם kbd_clk גבוה: - משרשרים למקום האחרון ברגיסטר ההזזה shift_reg את הסיבית החדשה שהגיעה מה- kbd_dat. `shift_reg_NS = {kbd_dat, shift_reg [9:1]};` - בודקים אם מונה הביטים cntr קטן ממספר הביטים. אם כן מקדמים את המונה.	בודקים אם מונה הביטים cntr קטן ממספר הביטים: - אם כן (טרם הגיעו כל הביטים של המילה) עוברים ל-HI_CLK_ST - אם לא (המילה השלמה התקבלה) עוברים ל-CHK_DATA_ST
HiClk	במצב זה השעון גבוה וממתינים לביט הבא (לשעון נמוך).	אם מגיע ביט חדש עוברים ל-LOW_CLK_ST
ChkData	מצב בו בודקים את נכונות הנתונים על ידי בדיקת הזוגיות parity_ok. - אם בדיקת הזוגיות טובה (1 לוגי) מעדכנים את המוצא בתכולת הרגיסטר `Dout_NS = shift_reg[7:0];`	אם בדיקת הזוגיות טובה עוברים ל-NEW_DATA_ST - אם הבדיקה לא טובה עוברים ל-IDLE_ST
NewData	במצב זה תמיד מודיעים על התו החדש `dout_new = 1'b1 ;`	**עוברים ל-IDLE_ST**

<u>בצע סינתזה.</u>

```
module bitrec
 (
      input logic  clk,
      input logic  resetN,
      input logic  kbd_dat,
      input logic  kbd_clk,
      output logic [7:0] dout,
      output logic dout_new,
      output logic parity_ok
  ) ;


  enum  logic [2:0] {IDLE_ST, // initial state
                             LOW_CLK_ST, // after clock low
                             HI_CLK_ST, // after clock hi
                             CHK_DATA_ST, // after all bits recieved
                             NEW_DATA_ST // valid parity laod new data
                             }  pres_st /* for simulation --> synthesis
keep = 1 */,
                                   next_st /* synthesis keep = 1 */;
```

```
  logic [3:0] cntr_PS,  cntr_NS /* for simulation -->synthesis keep = 1
*/ ;
  logic [9:0] shift_reg_PS, shift_reg_NS  /* for simulation -->synthesis
keep = 1 */ ;
  logic [7:0] Dout_NS  /* for simulation -->synthesis keep = 1 */ ;


  localparam NUM_OF_BITS = 9 ;  // &&&&&& fill please the right number

always_ff @(posedge clk or negedge resetN)
begin: fsm_sync_proc
      if (resetN == 1'b0) begin
            pres_st <= IDLE_ST ;
            cntr_PS <= 4'h0  ;
            shift_reg_PS <= 10'h000 ;
            dout <= 8'h00 ;

      end
      else begin
            pres_st <= next_st;
            cntr_PS <= cntr_NS ;
            shift_reg_PS <= shift_reg_NS ;
            dout <= Dout_NS  ;
      end ;
end // end fsm_sync_proc

always_comb
begin
      // default values
            next_st = pres_st ;
            cntr_NS = cntr_PS  ;
            shift_reg_NS = shift_reg_PS  ;
            Dout_NS = dout ;
            dout_new = 1'b0 ;



      case(pres_st)
            IDLE_ST:
            begin
//---------------
                  cntr_NS = 4'h0  ;

                  if( (kbd_clk == 1'b0) && (kbd_dat == 1'b0) )  //check
start bit
                        next_st = LOW_CLK_ST;
            end

            LOW_CLK_ST:
            begin
//---------------
                  if (kbd_clk == 1'b1)
                        begin
                              shift_reg_NS = {kbd_dat, shift_reg_PS
[9:1]};
                              if (cntr_PS < NUM_OF_BITS)
                                    begin
                                          cntr_NS <= cntr_PS + 1;
                                          next_st = HI_CLK_ST;
                                    end
```

```
                                else if (cntr_PS == NUM_OF_BITS)
                                      begin
                                            next_st = CHK_DATA_ST;
                                      end
                            end

            end

            HI_CLK_ST:
            begin
//---------------
                  if (kbd_clk == 1'b0)
                        begin
                              next_st = LOW_CLK_ST;
                        end
            end

            CHK_DATA_ST:
            begin
//---------------
                  if ((kbd_clk == 1'b1) && (kbd_dat == 1'b1)) //check
stop bit
                  begin
                        if (parity_ok == shift_reg_PS[9])
                              begin
                                    Dout_NS = shift_reg_PS[8:1];
                                    next_st = NEW_DATA_ST;
                              end
                        else
                              begin
                                    next_st = IDLE_ST;
                              end
                  end
            end

            NEW_DATA_ST:
            begin
                        dout_new = 1'b1;
                        next_st = IDLE_ST;
//---------------
            end
      endcase
end

// parity Calc [1'b1 to apply the NOT]
assign parity_ok = 1'b1 ^ shift_reg_PS[8] ^ shift_reg_PS[7] ^
shift_reg_PS[6] ^ shift_reg_PS[5] ^ shift_reg_PS[4]
       ^ shift_reg_PS[3] ^ shift_reg_PS[2] ^ shift_reg_PS[1];

endmodule
```

Flow Summary

<<Filter>>

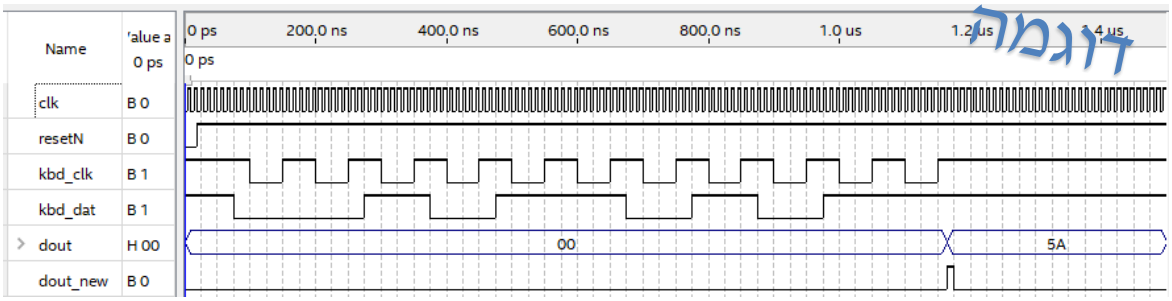
Flow Status	Successful - Sun Aug 14 13:03:57 2022
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	KBDINTF
Top-level Entity Name	bitrec
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	27
Total pins	14
Total virtual 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 סינתזה מוצלחת.

3.2 סימולציה

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

פתח קובץ סימולציה חדש ושרטט את אות הכניסה הבא (עבור כניסת מקש ה- Enter 5Ah או המילה של 11 סיביות בבינארי 00101101011.



הדרכה לסימולציה: מומלץ להגדיר:

- **שעון** מערכת (clk) מהיר פי 10 משעון המקלדת (kbd_clk): למשל, קבע בשעון המערכת period=10nsec ובשעון המקלדת period=100nsec.
- **grid** של 25 nsec ושים לב שהשינוי ב- kbd_dat מתרחש בזמנים ששעון המקלדת ב- '1' לוגי!
- **משך הסימולציה** כ- End time = 1.4 usec

הראה בסימולציה שלך תוצאות זהות לדוגמה הנתונה לעיל. הראה שאם מכניסים רצף טורי של קוד מקש נתון ב- kbd_dat, מתקבל:

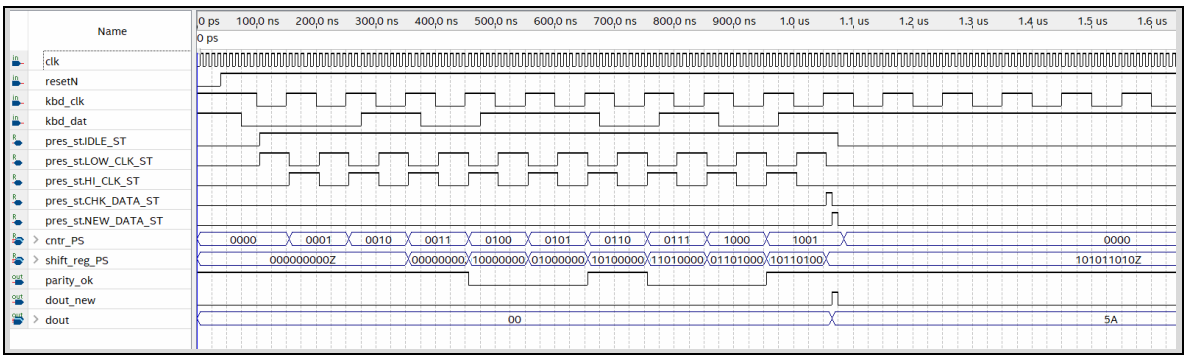
- ב- dout: 5AH מקבילי (הצג אות זה ב- radix hexadecimal)

- ב- dout_new '1' במשך מחזור שעון אחד שמודיע על מקש חדש אחרי שה- kbd_clk האחרון הסתיים (אחרי ה- Stop bit).

חשוב מאד: לביצוע הסימולציה יש להזין אך ורק את אות המבוא kbd_dat כפי שנתון בדוגמה לעיל!

חשוב להראות בסימולציה גם סיגנלים פנימיים כגון המונה, הSHIFT REGISTER ומכונת המצבים (שורה לכל מצב) , שים לב שעל מנת שהסימולציה לא תצמצם את המשתנים יש להוסיף להגדרת המשתנים את הפקודה הבאה, כפי שכבר קיים בקוד:

```
logic  [3:0] cntr_PS,  cntr_NS /* synthesis keep = 1 */ ;
```



4 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 Enter מהסוג החדש הוא (E0 5A). כמו כן, שעון המקלדת PS2_CLK, שמשמש לסנכרון סיביות הנתונים של PS2_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:

נעזר בנוסחה:

bits*time*frequency = memory

Bits: 11+1+11 מכיוון שלפי הנתון ישנם מקשים אשר דורשים קידוד באורך 11 סיביות פעמיים עם מחזור שעון הפסקה ביניהם

Time:

PS2_CLK=1/12.5KHZ=80usec

Frequency:

תדר שעון הדגימה הוא 50MHZ

לכן קיבלנו: 23*80usec*50MHZ=92K

ונצטרך לבחור 128K (כאשר נעגל כלפי מעלה)

5 מטלת תכן עם מקלדת

רקע למטלה: בישומים רבים אפשר להשתמש במקלדת לביצוע פעולות שונות, בדומה למפסקים ולחצנים שעל הכרטיס. במטלה זו תלמד איך להשתמש בממשק למקלדת לביצוע פעולות באמצעות מקשים מסוימים. בדוגמה כאן משתמשים במקלדת מספרים של 18 מקשים בלבד (ראה הסבר בחומר הרקע).

פתח את הקובץ הגרפי הנתון KEYBOARD_DECODER.bdf .

הסבר למודול KEYBOARD_DECODER.bdf:

בישום זה קוד המקש המופק מממשק המקלדת (KBDINTF) מוזן לשני מודולים מאותו סוג בשם **singleKeyDecoder.sv**, במטרה לעשות פעולות מסויימות באמצעות מקשים מוגדרים של המקלדת.

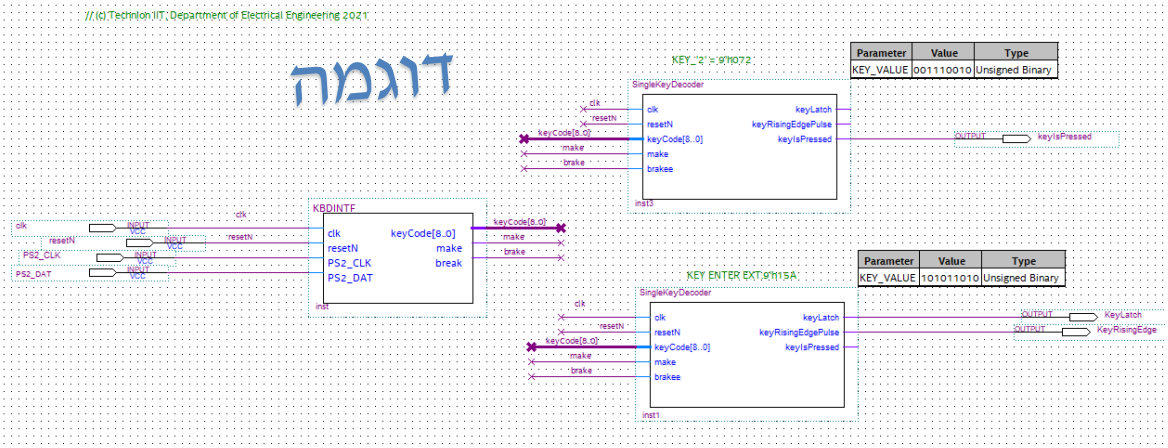
במטלה זו נתמקד במודול singleKeyDecoder.sv ונשלים בו קוד חסר.

הסבר למודול singleKeyDecoder.sv

מודול זה מקבל כניסות מממשק המקלדת – קוד מקש, ואותות make ו- break.
הוא מזהה מקש ספציפי לפי קוד מקש נתון כפרמטר ומפיק 3 אותות שונים:

- keyLatch - מחליף מצב כל לחיצה על המקש בין 0 לוגי ל-1 לוגי
- keyRisingEdgePulse - גוזר, מוציא פולס צר בתחילת הלחיצה על המקש
- keyIsPressed - מוציא 1 לוגי בכל משך הזמן שהמקש לחוץ

בדוגמה הנתונה משתמשים פעמיים במודול זה, פעם עבור **מקש " 2 חץ למטה"** (עם הפרמטר 9'h072) ופעם עם מקש **ENTER** (עם הפרמטר 9'h15A).



פתח את **הקובץ singleKeyDecoder.sv וסיים** את כתיבת המימוש שלו **במקום בו כתובה ההערה** (העזר בחומר רקע)

```
&&&&&&&&&&    fill your code please
```

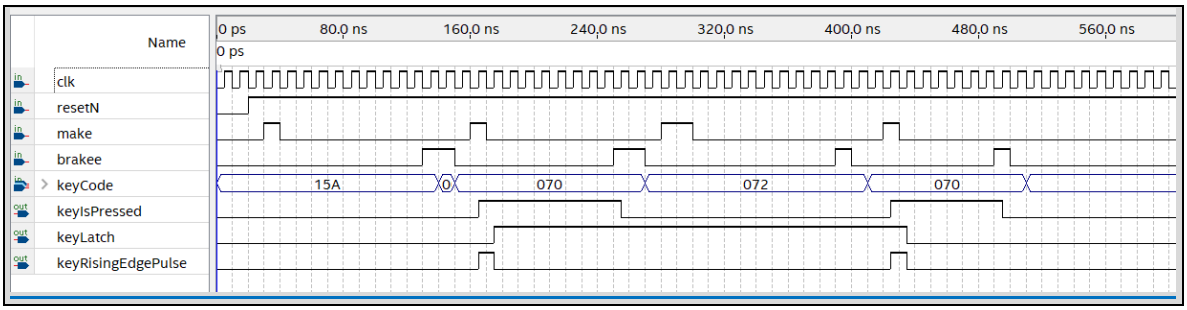
בדוגמה הנתונה מה עושה המקש "2 חץ מטה" ומה עושה המקש ENTER?
תשובה:
"2 חץ מטה" מזהה את כל זמן לחיצת המקש ולכן מתפקד ככפתור, לעומתו מקש ENTER מזהה את תחילת הלחיצה בלבד ומחליף מצב בכל פעם שלוחצים על המקש ולכן מתפקד כמתג

```
always_ff @(posedge clk or negedge resetN)
   begin: fsm_sync_proc
      if (resetN == 1'b0) begin
         keyIsPressed_d <= 0  ;
         keyIsPressed  <= 0 ;
         keyLatch <= 0 ;

      end
      else begin
         if (keyCode  == KEY_VALUE )
         begin
            if (make == 1'b1)
               keyIsPressed  <= 1'b1 ;
            if (brakee == 1'b1)
               keyIsPressed  <= 1'b0 ;

         end
```

בצע סימולציה למודול זה (**singleKeyDecoder.sv**) והראה ששלוש היציאות עובדות נכון עבור לפחות שני מקשים שונים. פעם או פעמיים עם מקש שעובד איתו ופעם עם מקש אחר כלשהוא, וזאת כדי לוודא שהמודול לא מגיב אליו.



לא הגיב למקש 15A וכן לא הגיב למקש 072, לעומת זאת הגיב למקש אותו אמור לגלות (070) כמצופה.

6 הכנה למעבדה וגיבוי העבודה

כהכנה נוספת למעבדה מומלץ לראות את הסרטונים הנתונים במודל, המסבירים על השימוש בנתח הלוגי, ה- SignalTap.

סרטונים להבהרת השימוש בנתח הלוגי, ה- Signal Tap

שמור דוח זה רגיל וכ- PDF והעלה את קובץ ה- PDF למודל.

שמור את הפרויקט רגיל וגם כארכיב (באמצעות Project -> Archive Project).
שים לב לשנות את השם שמציע הקוורטוס לשם קצר, שאינו מכיל: עברית, רווחים ו/או את הסימן '–' ומכיל את התאריך והשעה של יצירת הארכיב.

העלה את קובץ הארכיב למודל, כי תצטרך אותו בניסוי, בסיס לעבודתך במעבדה.

גבה את הדוח והפרויקט גם באמצעים אחרים.

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

מלא את הטופס