**הטכניון - מכון טכנולוגי לישראל**
**הפקולטה להנדסת חשמל**

# מעבדה בהנדסת חשמל
# 044157 'א1

# ממשק מסך VGA וצלילים

# תדריך מעבדה

# גרסה 1.21

# ללא מקלדת

# קיץ 2020


נכתב על ידי: דודי בר-און, אלכס גרינשפון, ליאת שורץ,
נעם ליבוביץ עציון


| מועד | ביצוע עד סעיף | שם המדריך בפועל | תאריך |
|---|---|---|---|
| ביצוע הניסוי | 6 | אברהם קפלן | 25.08.2020 |
| השלמת חלקים חסרים | | | |

| סטודנט | שם פרטי | שם משפחה |
|---|---|---|
| 1 | יפתח | אדלשטיין |
| 2 | חן | קציר |

# תוכן עניינים

**מטרות הניסוי**:

- הכרת ממשק ה- VGA
- הכרת ממשק השמע
- הרחבת היישום
- אינטגרציה עם מקלדת והשמעת צליל

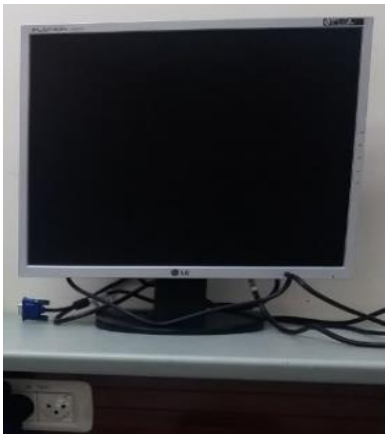
# 1 כללי

הורד מהמודל את קובץ הארכיב של המעבדה ופתח אותו לפרויקט בתיקיה יעודית על הדיסק שלך (לא לפתוח פרויקט ב- DOWNLOADS, אלא להעביר לתיקייה שלך כדי שתהיה לך גישה אליו מכל עמדת מחשב אחרת).

# 2 הכרת פלטפורמת ה- VGA

## *2.1 חיבור המערכת*

- הפעל את כרטיס הניסוי. בדוק שהכרטיס והמסך הנוסף שנמצא בכל עמדת עבודה על המדף (מעל שני המסכים של העמדה) נדלקים.



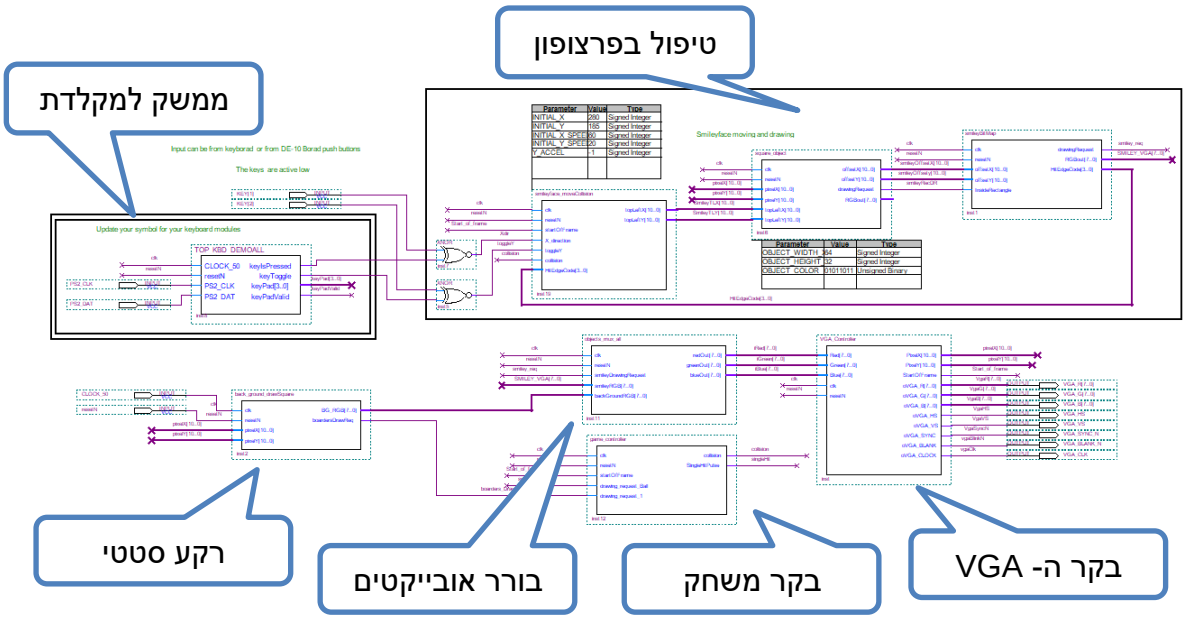
## *2.2 הפעלת יישום ה- VGA*

- הורד מהמודל את קובץ הארכיב של מעבדת VGA ופתח אותו בתיקייה משלו.
- בפרויקט שנפתח פתח את ה- TOP: TOP_VGA_DEMO_WITH_MSS_ALL.bdf.

ה- TOP מחולק לאזורים (כמו באיור הבא) ובכל שלב במעבדה זו נתמקד באזור אחר:

בשלב זה נתמקד בממשק ה- VGA ובמודולים השונים שמרכיבים אותו (ללא הצלילים). תחילה נבדוק שיישום ה- Smiley בכללותו עובד נכון.

- בצע סינתזה ל- TOP, הרץ קובץ הדקים (tcl) ואז הרץ קומפילציה מלאה.
- צרוב את הפרוייקט לכרטיס, ודא שיישום ה- Smiley עובד נכון ואשר זאת עם המדריך.




***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 9:00***

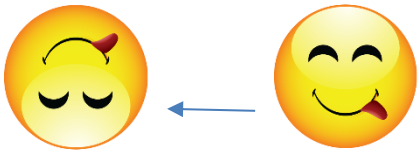
## 3 ביצוע פעולות להרחבת היישום

אם תרצה לא לדרוס את הקובץ המקורי שמור אותו בשם אחר ומכאן והלאה תעבוד איתו, תוסיף/ תעדכן מודולים כפי שייתואר בהמשך.

### *3.1 שינוי BITMAP - הפיכת ה- SMILEY*

**<u>משימה</u>**: לשנות את קובץ ה- BITMAP. ספציפית יש לסובב את הביטמאפ עצמו ב- 180°. – (לא תמונת ראי)

- פתח את מודול הביטמאפ smileyBitMap.sv והבן את אופן פעולתו
- שנה את הקוד כך שיציג את הסמיילי מסובב כמו בתמונה.



**<u>הנחיה</u>**: בקובץ הביטמאפ שים לב באיזה מהצירים יש לעשות שינוי כך שמה שהיה למעלה עכשיו יוצג למטה.

```
if (InsideRectangle == 1'b1 )  // inside an external bracket
RGBout <= object_colors[OBJECT_HEIGHT_Y-1-offsetY][offsetX];
```

**שים לב!** מכאן והלאה המשך לעבוד עם הפרצופון המסובב.

***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 9:10***

# 4 חיבור מכלולים נוספים

## ~~4.1 חיבור הממשק למקלדת~~

**~~משימה~~**: ~~לחבר את המקלדת שבנית במעבדת DEBUG לפרוייקט זה.~~

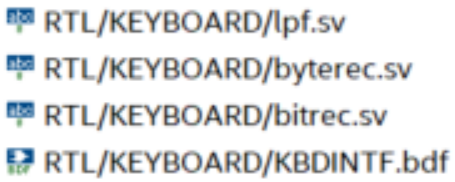
### ~~4.1.1 הוספת הממשק למקלדת לפרוייקט~~

~~בקובץ QAR שהורדת נתון לך מודול TOP_KBD_DEMOALL שמכיל את הקבצים:~~

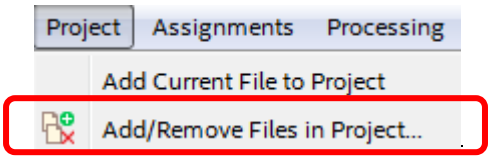
- ~~**keyToggle_decoder.sv** – קובץ נתון שהכרת במעבדה קודמת~~
- ~~**keyPad_decoder.sv** – קובץ נתון המתרגם את המקשים (0-9 ו- a-f) לקוד לביטמאפ אותו ניתן להציג על המסך~~
- ~~**KBDINTF** – קובץ חסר, שאמור להכיל את הממשק למקלדת~~

~~עליך לייבא את הקובץ KBDINTF שבנית במעבדת DEBUG לפרוייקט זה.~~

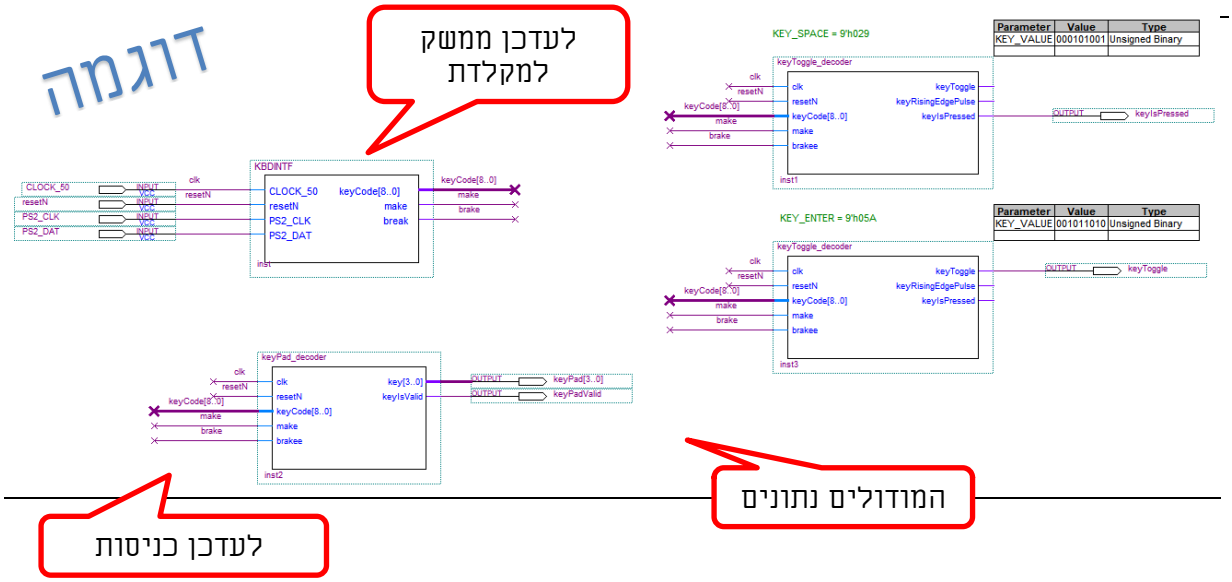
- ~~לשם כך:~~
  - ~~העתק מפרוייקט המקלדת (ממעבדת DEBUG) את הקבצים הרלוונטיים של הממשק למקלדת, לתיקית הפרוייקט הנוכחי RTL/KEYBOARD:~~




  - ~~שים לב להעתיק את כל הקבצים לתיקיית הפרוייקט, ולא לקחת רק את הקישור למקום אחר. הוספת קבצים שלא בתיקיית הפרוייקט, תקשה על שמירתם והעברתם למחשב אחר~~
  - ~~הוסף את הקבצים שהעתקת לפרוייקט הראשי בעזרת:~~




~~הקובץ TOP_KBD_DEMOALL אחרי ההשלמות צריך להיראות בערך כך:~~



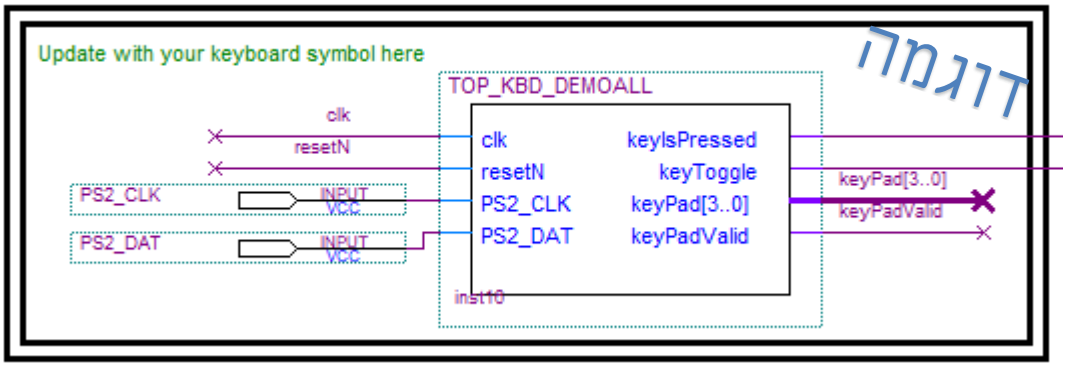

- ~~צור Symbol עבור הממשק למקלדת~~
- ~~ב- TOP_KBD_DEMOALL~~
  - ~~הוסף את ה- Symbol של הממשק למקלדת במקום המסומן~~
  - ~~הוסף את החיבורים הנדרשים; שים לב להשלים גם את הכניסות למודול ה- keyPad_decoder~~
  - ~~בצע אנליזה לקובץ TOP_KBD_DEMOALL~~
  - ~~צור סימבול עבורו.~~

~~על ידי File –> Create/Update –> Create Symbol File for Current File.~~

### ~~4.1.2 בדיקת תקינות של ממשק המקלדת~~

**~~משימה~~**: ~~לבדוק תקינות של פעולת המקלדת ביישום הפרצופון~~

- ~~חזור להיררכיה העליונה והגדר אותה כ- TOP.~~
- ~~עדכן את הסימבול של ממשק המקלדת בהיררכיה העליונה של היישום (על ידי מקש ימני על הסימבול ובחירת הפעולה Update Symbol).~~




- ~~בצע קומפילציה מלאה והורד את היישום לכרטיס.~~
- ~~בדוק את פעולת המקשים 0-9, a-f, SPACE, ENTER.~~

- שים לב ששני לחצנים הוגדרו כעוקפים של המקלדת בקובץ tcl. נסה אותם וראה איזה לחצן מתאים לאיזה מקש.

כיצד אתה משפיע על תנועת הפרצופון מהמקלדת ועם הלחצנים?
**תשובה**: לחיצה על KEY[0] מבצעת איפוס ע"י RESETN
החזקה של KEY[1] מבצעת שינוי בכיוון תנועת הסמיילי על ציר X
לחיצה קצרה על KEY[2] מבצעת שיחלוף של כיוון תנועת הסמיילי בציר Y

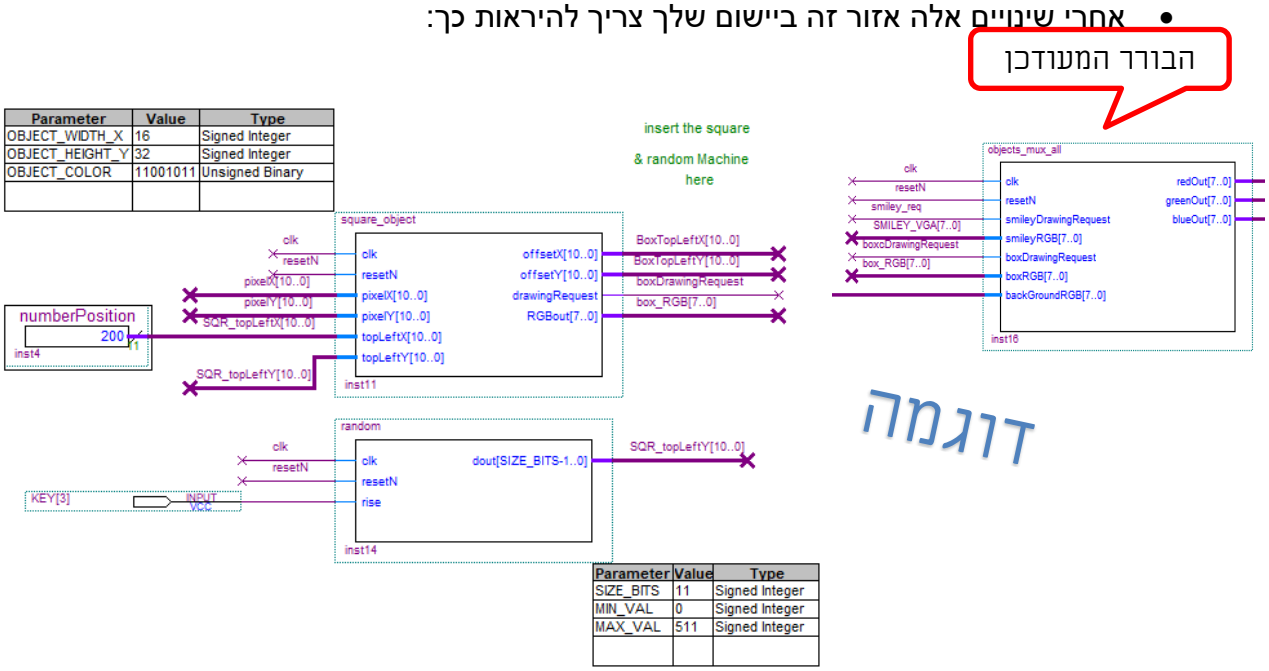

***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 9:20***

## 4.2 הוספת מלבן מעל הרקע הסטטי ומכונת RANDOM

**משימה**: להוסיף מעל הרקע הסטטי מלבן נייח (אוביקט נפרד), שאפשר לקבוע את מיקומו חיצונית, ולחברו לבורר העדיפויות.

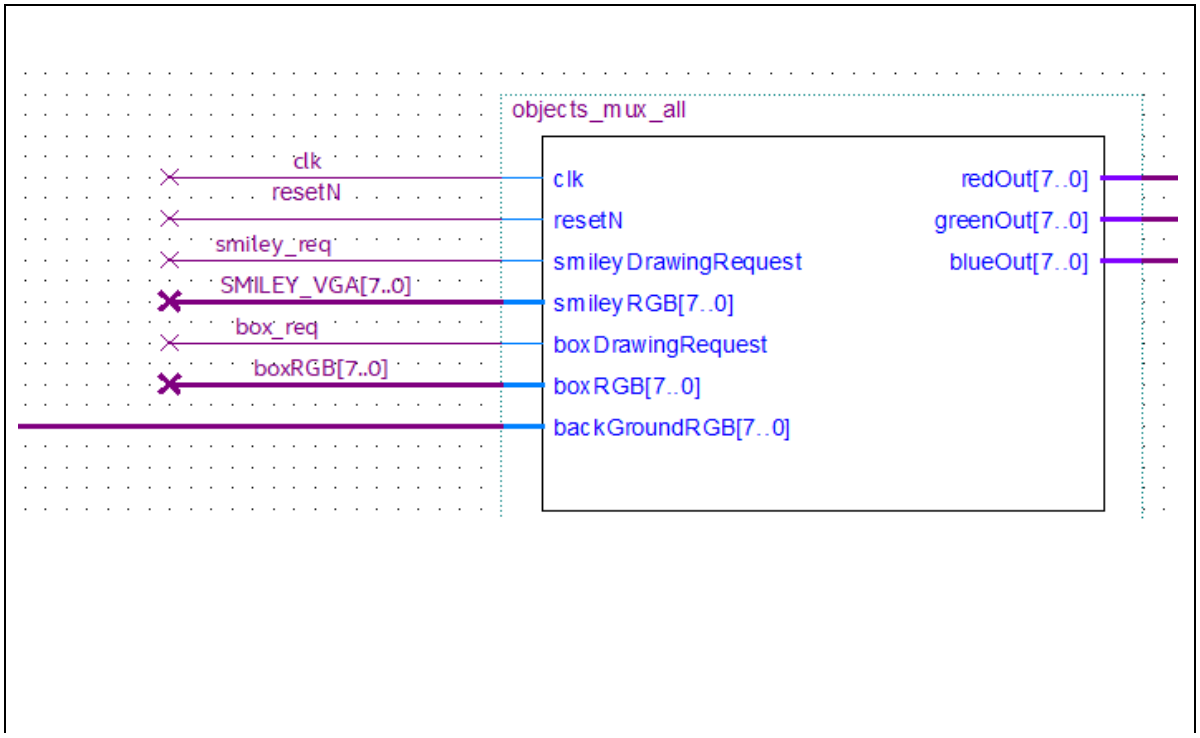
- הוסף ל- TOP של הפרויקט שלך, רכיב (instance) נוסף מסוג square_object (הרכיב כבר קיים בפרויקט אך יש ליצור Symbol שלו). ראה שרטוט להלן.
- ניתן לקבוע את גודלו ולבחור את צבעו של המלבן כרצונך, באמצעות פרמטרי הרכיב.
- יש למקם את המלבן החדש במיקום אופקי קרוב למסלול תנועתו של הפרצופון, כך שהוא והמלבן יגעו אחד בשני לפחות לרגע קט במהלך תנועת הפרצופון.
- קבע את הקואורדינטה topLeftX באופן ידני.
  - לשם כך השתמש ברכיב מסוג LPM_CONSTANT הקיים בשרטוט.
  - (בעתיד, להוספת רכיב נוסף מסוג זה ניתן להעזר ב - COOK BOOK).
- הקואורדינטה topLeftY תיקבע באופן אקראי.
  - לשם כך חבר מכונת RANDOM (קיימת בפרוייקט, צור Symbol שלה) אשר תופעל בעזיבת הלחצן key[3].
  - שים לב שיש להתאים את גודל וקטור המוצא של מכונת RANDOM לגודל וקטור הכניסה של מודול המלבן (על יד קביעת פרמטר של הרכיב).
- כמו כן עליך להוסיף עוד זוג כניסות ולוגיקה מתאימה לבורר העדיפויות (mux) לטיפול במלבן. בלוגיקה קבע עדיפותו מעל הרקע ומתחת לפרצופון.
  - פתח קובץ נתון בפרויקט שלך objects_mux_all והשלם בו את הכניסות והלוגיקה הדרושות עבור האובייקט הנוסף (המלבן).
  - צור Symbol למודול זה.
  - בהיראכיה העליונה החלף את הבורר הישן בחדש וחבר אליו את המלבן.

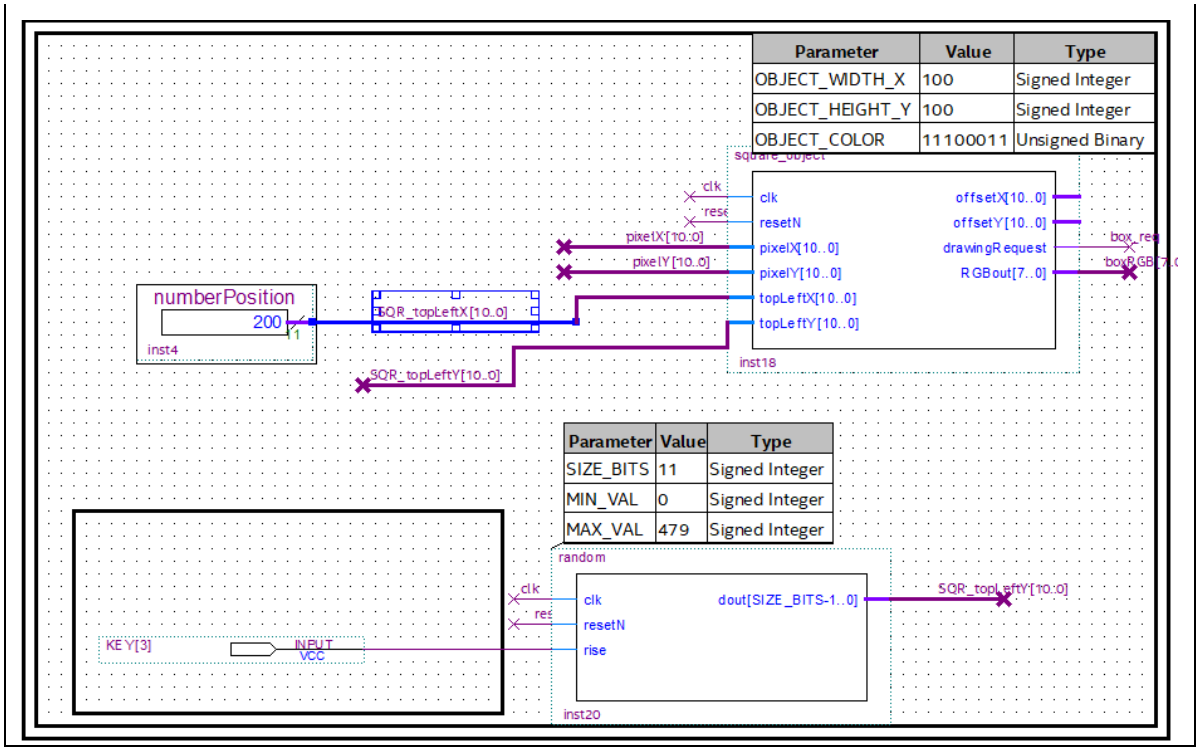
- אחרי שינויים אלה אזור זה ביישום שלך צריך להיראות כך:




| Parameter | Value | Type |
|---|---|---|
| OBJECT_WIDTH_X | 16 | Signed Integer |
| OBJECT_HEIGHT_Y | 32 | Signed Integer |
| OBJECT_COLOR | 11001011 | Unsigned Binary |
| | | |

| Parameter | Value | Type |
|---|---|---|
| SIZE_BITS | 11 | Signed Integer |
| MIN_VAL | 0 | Signed Integer |
| MAX_VAL | 511 | Signed Integer |
| | | |






```
//-- Alex Grinshpun Apr 2017
//-- Dudy Nov 13 2017
// SystemVerilog version Alex Grinshpun May 2018
// coding convention dudy December 2018
// (c) Technion IIT, Department of Electrical Engineering 2019

module      objects_mux_all    (
//          --------    Clock Input
                                input       logic clk,
                                input       logic resetN,
            // smiley
                                input       logic smileyDrawingRequest, //
two set of inputs per unit
                                input       logic [7:0] smileyRGB,
            // add the box here
                                input    logic boxDrawingRequest,
                                input    logic [7:0] boxRGB,

            // background
                                input       logic [7:0] backGroundRGB,

                                output      logic [7:0] redOut, // full 24
bits color output
                                output      logic [7:0] greenOut,
                                output      logic [7:0] blueOut

);

logic [7:0] tmpRGB;


assign redOut       = {tmpRGB[7:5], {5{tmpRGB[5]}}}; //--  extend LSB to
create 10 bits per color
assign greenOut  = {tmpRGB[4:2], {5{tmpRGB[2]}}};
```

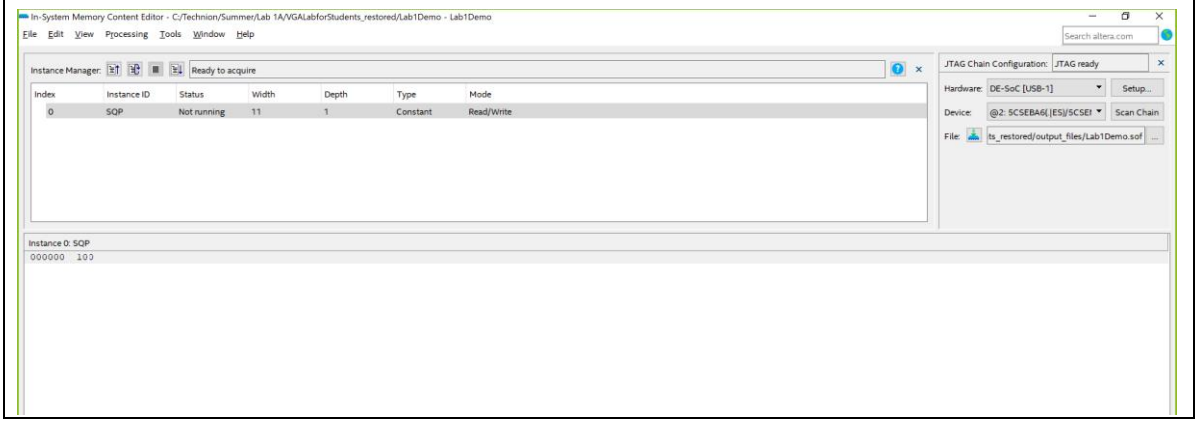
```
assign blueOut       = {tmpRGB[1:0], {6{tmpRGB[0]}}};

//
always_ff@(posedge clk or negedge resetN)
begin
      if(!resetN) begin
                  tmpRGB       <= 8'b0;
      end
      else begin
            if (smileyDrawingRequest == 1'b1 )
                  tmpRGB <= smileyRGB;  //first priority

            //add code here

            else
                  if(boxDrawingRequest == 1'b1 )
                        tmpRGB <= boxRGB;
                  else
                        tmpRGB <= backGroundRGB ; // last priority
            end ;
      end

endmodule
```

***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 9:47***

## 4.3 תרגול שימוש בעורך הזכרון, ה - ISMCE

**<u>משימה</u>**: לשנות את מיקומו של אובייקט המלבן באמצעות ה- ISMCE.

- הפעל את ה- ISMCE (העזר ב - COOK BOOK).
- שנה את הקואורדינטה topLeftX תוך כדי הפעלת היישום עד שהמלבן חותך את המסלול הפרצופון במקום אחד לפחות.



## 4.4 תרגול שימוש בנתח הלוגי, ה- SIGNAL_TAP

**<u>משימה</u>**: לבדוק בעזרת הנתח הלוגי איך קובעים מיקום אקראי למלבן הנוסף

- הפעל את הנתח הלוגי (העזר ב - COOK BOOK) וקבע את כל הפרמטרים שלו.
- הצג בנתח הלוגי את האותות שנראים לך רלוונטים במקרה זה.

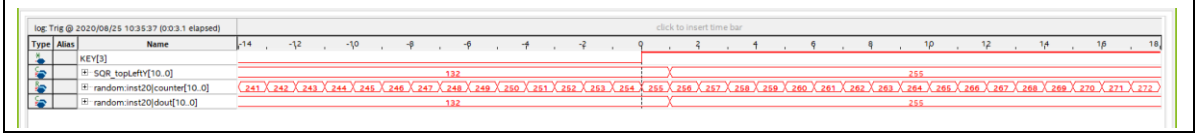
כיצד תקבע את תנאי ה- Trigger במקרה זה?

**<u>תשובה</u>**: נקבע את תנאי הטריגר לעליית המתח בקלט KEY[3]

- לחץ מספר פעמים על הלחצן בכרטיס שמייצר **rise** (איזה לחצן זה?) וראה כיצד זה משפיע על מיקום המלבן בזמן שהמערכת פעילה.

האם השינוי קורה בלחיצה או בעזיבה ? מדוע ?

**<u>תשובה</u>**: הוא קורה בעזיבה, כלומר בשחרור הכפתור KEY[3]



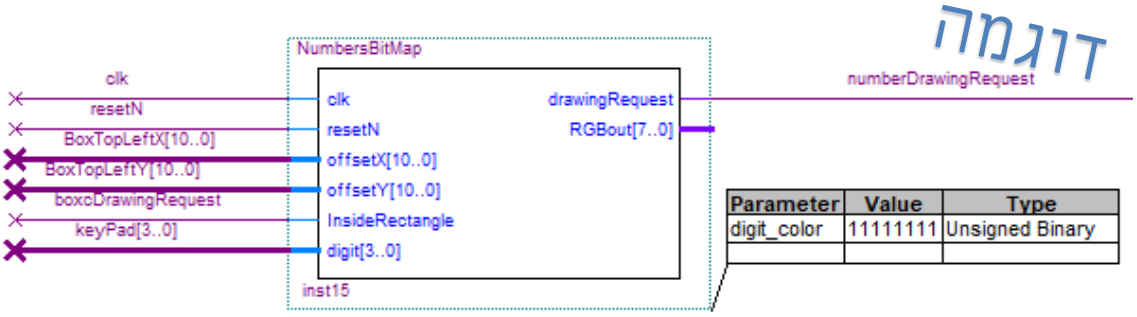

***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 10:40***

## 4.5 הוספת ספרות/אותיות ליישום

**<u>מטרה</u>**: להוסיף ספרות/אותיות כאובייקטים גרפיים מעל הרקע.

נתון לך מודול שמכיל ביטמאפ של ספרות ואותיות 0 עד F, **NumbersBitMap**.

- פתח אותו והבן את פעולתו.



חבר את הממשק לארבעה מפסקים שיתנו תא קוד הספרה

**שים לב**: צורת חיבור זו דומה לצורת החיבור של הפרצופון, מלבן שקובע את גבולות הצורה ומודול שנותן את התוכן שלה, כביטמאפ. לכן מימדי שני הרכיבים האלה צריכים להיות מתואמים.

- השתמש במלבן מסעיף קודם (הוספת אובייקט מלבן מעל הרקע הסטטי):
  - קבע את גודלו של המלבן ל- x = 16, y = 32, כדי להתאים לגודל הביטמאפ של האותיות והספרות שנרצה להציג
  - בקשת השרטוט של המלבן נכנסת לכניסת הביטמאפ של המספרים (חיבור זה כבר קיים באמצעות שמות החוטים)
- חבר את ה- drawing request של הביטמאפ לכניסה של ה- mux object במקום הבקשה לשרטוט של המלבן (אפשר גם באמצעות שמות חוטים)
- קמפל וצרוב לכרטיס.
- בדוק שהספרות /אותיות מוצגות נכון על המסך
- בדוק את פעולת המקשים 0-9, a-f,

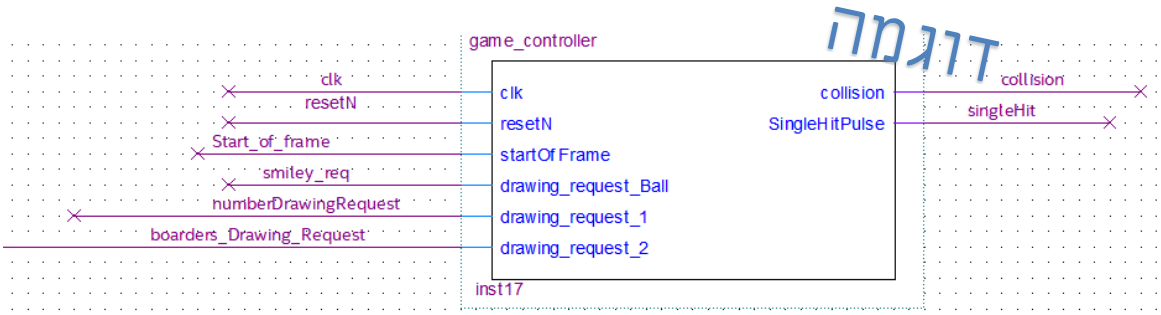
## *4.6 אינטגרציה ובקרת משחק*

**משימה**: להוסיף בקר משחק ליישום: כאשר הפרצופון מתנגש באובייקט (המלבן הנוסף שמעל הרקע) הפרצופון יידחה, ישנה כיוון תנועה וימשיך את תנועתו כמקודם.
כמו כן, נרצה כי בכל התנגשות על המסך (לא משנה בין מה למה) יוגרל מחדש מיקומו של המספר על המסך.

- לשם כך חבר את יציאת ה- SingleHitPulse של הבקר לכניסה rise במקום לחצן key[3] כך שבכל התנגשות מכונת ה- RANDOM תגריל מחדש את מיקומו של המספר.

התאם את הרכיב **game_controller.sv** הקיים לדרישה (אפשר לשמור אותו בשם אחר, למשל **game_controller_all.sv**, כדי לא לדרוס את הקובץ הקיים):

- הוסף לרכיב הנוכחי game_controller.sv כניסה נוספת שתזהה התנגשות עם המלבן החדש, כך ש- COLLISION יתקיים אם יהיה בו זמנית drawing request של הסמיילי ושל לפחות אחת הכניסות, לא משנה איזו.
- **הכניסות שלו**: בקשת השרטוט של הפרצופון,של המלבן הנוסף (עם המספרים) ושל גבולות הרקע
- **היציאות שלו:** אות שיהיה 1 לוגי אם יש התגשות בין המלבן והפרצופון, ו- flag המוציא 1 למשך מחזור שעון יחיד כאשר יש התנגשות, חשבו לאן יש לחבר יציאה זו
- הוסף לוגיקה המזהה התנגשות בין הפרצופון והמלבן, כלומר 3 בקשות השרטוט המתקיימות באותו זמן באותו מקום. דוגמה לרכיב כזה:




- קמפל ועשה לו Symbol.

## ב- TOP של היישום

- הוסף את המודול game controller החדש ליישום.
- חבר את הכניסות/יציאות החדשות לפי הצורך .
- קמפל, הורד לכרטיס ובדוק שהיישום עובד כמו שהתכוונת . אם לא, תקן בהתאם.

| |
|---|
| Game Controller |

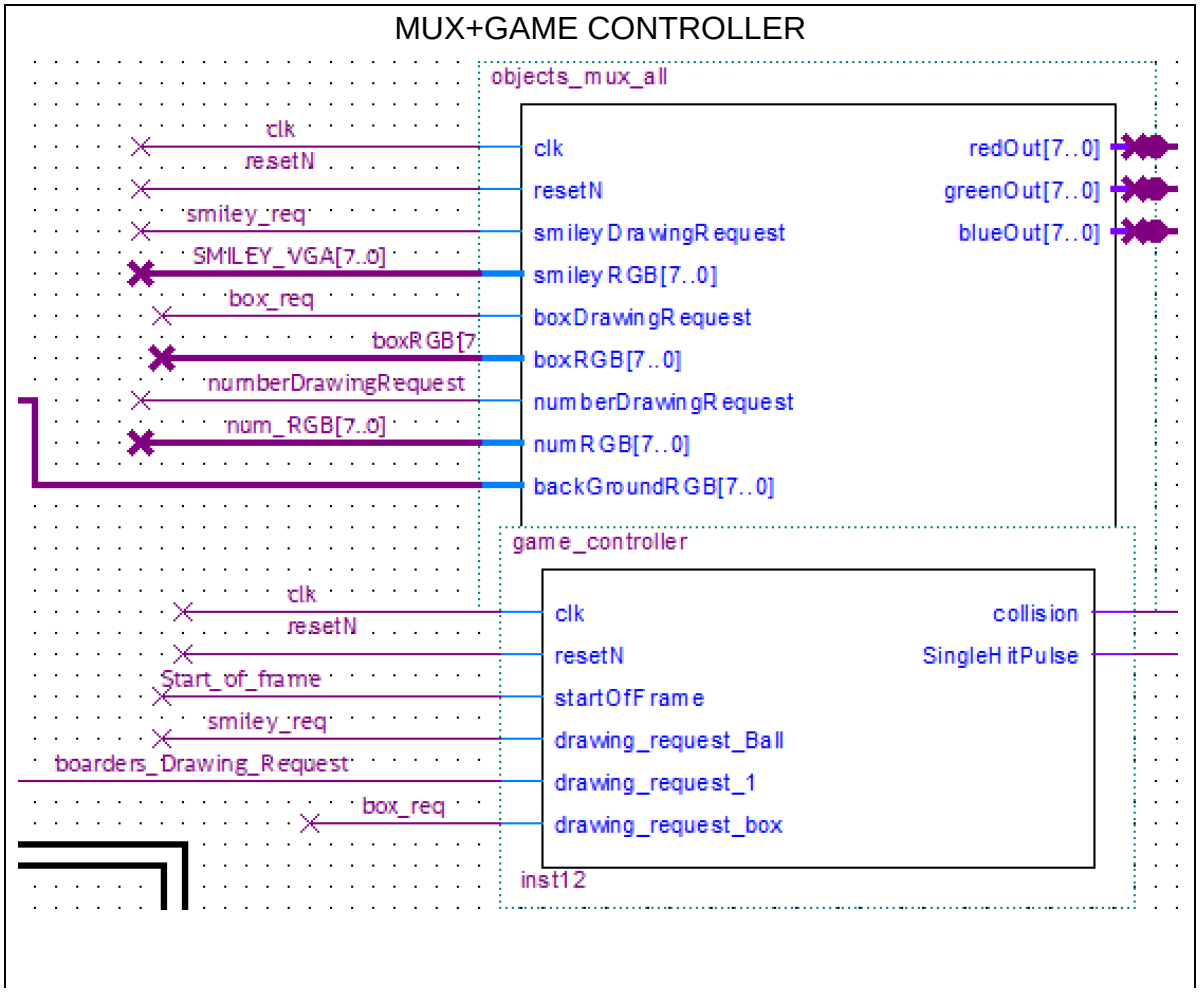
```
// game controller dudy Febriary 2020
// (c) Technion IIT, Department of Electrical Engineering 2020


module      game_controller   (
                  input       logic clk,
                  input       logic resetN,
                  input logic startOfFrame,  // short pulse every start
of frame 30Hz
                  input logic drawing_request_Ball,
                  input logic drawing_request_1,
                  input logic drawing_request_box,

                  output logic collision, // active in case of collision
between two objects
                  output logic SingleHitPulse // critical code,
generating A single pulse in a frame
);

logic box_smiley_collision,box_edge_collision, edge_smiley_collision;
assign box_smiley_collision = (drawing_request_Ball &&
drawing_request_box) ;
assign box_edge_collision = (drawing_request_box&&  drawing_request_1);
assign edge_smiley_collision = (drawing_request_Ball &&
drawing_request_1);
assign collision = (edge_smiley_collision || box_smiley_collision);
logic flag ; // a semaphore to set the output only once per frame /
regardless of the number of collisions

always_ff@(posedge clk or negedge resetN)
```

```
begin
      if(!resetN)
      begin
            flag  <= 1'b0;
            SingleHitPulse <= 1'b0 ;
      end
      else begin

                  SingleHitPulse <= 1'b0 ; // default
                  if(startOfFrame)
                        flag = 1'b0 ; // reset for next time
                  if ( ( box_edge_collision || box_smiley_collision ) &&
(flag == 1'b0)) begin
                        flag  <= 1'b1; // to enter only once
                        SingleHitPulse <= 1'b1;
                  end ;

      end
end

endmodule
```

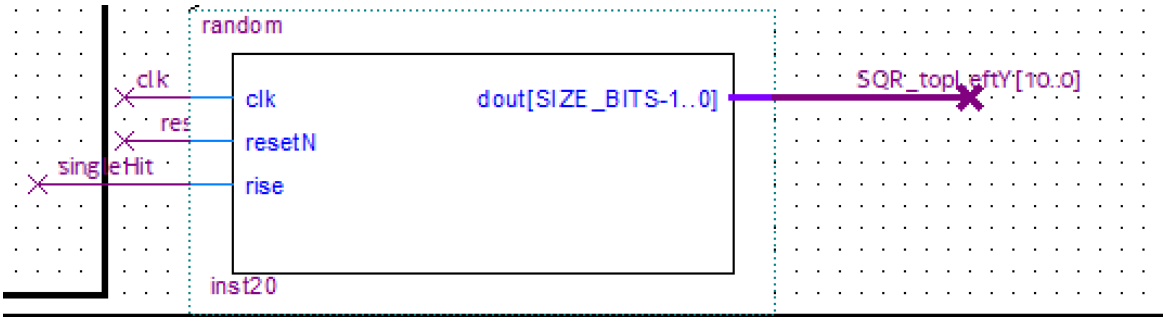
## MUX

```
//-- Alex Grinshpun Apr 2017
//-- Dudy Nov 13 2017
// SystemVerilog version Alex Grinshpun May 2018
// coding convention dudy December 2018
// (c) Technion IIT, Department of Electrical Engineering 2019

module      objects_mux_all   (
//          --------    Clock Input
                              input       logic clk,
                              input       logic resetN,
            // smiley
                              input       logic smileyDrawingRequest, //
two set of inputs per unit
                              input       logic [7:0] smileyRGB,
            // add the box here
                              input    logic boxDrawingRequest,
                              input    logic [7:0] boxRGB,
            // add the box here
                              input    logic numberDrawingRequest,
                              input    logic [7:0] numRGB,

            // background
                              input       logic [7:0] backGroundRGB,

                              output      logic [7:0] redOut, // full 24
bits color output
                              output      logic [7:0] greenOut,
                              output      logic [7:0] blueOut

);

logic [7:0] tmpRGB;
```

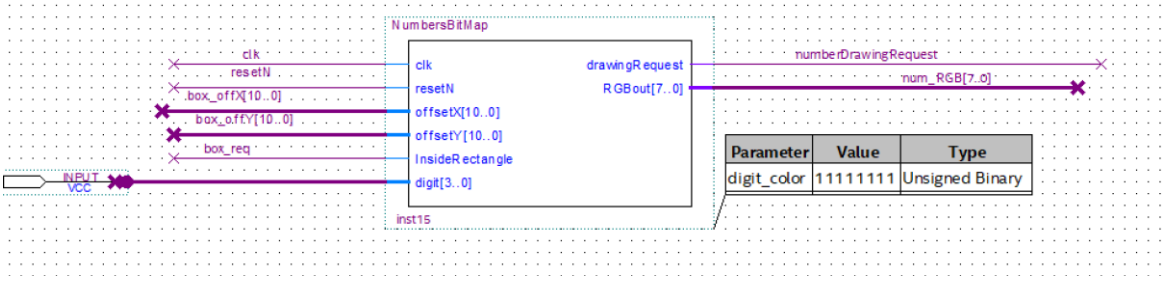
```
assign redOut        = {tmpRGB[7:5], {5{tmpRGB[5]}}}; //--  extend LSB to
create 10 bits per color
assign greenOut   = {tmpRGB[4:2], {5{tmpRGB[2]}}};
assign blueOut       = {tmpRGB[1:0], {6{tmpRGB[0]}}};

//
always_ff@(posedge clk or negedge resetN)
begin
      if(!resetN) begin
                  tmpRGB       <= 8'b0;
      end
      else begin
            if (smileyDrawingRequest)    tmpRGB <= smileyRGB;
            else if(numberDrawingRequest) tmpRGB <= numRGB;
            else if(boxDrawingRequest) tmpRGB <= boxRGB;
            else tmpRGB <= backGroundRGB ; // last priority
            end ;
      end

endmodule
```



# Randomizer

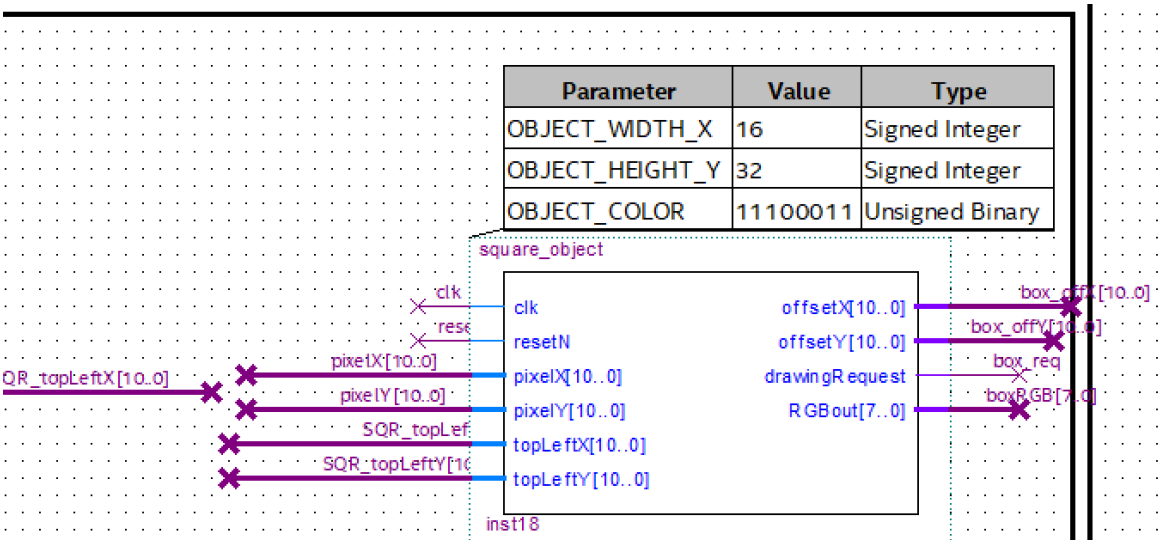



# NumbersBitMap




| Parameter | Value | Type |
|---|---|---|
| digit_color | 11111111 | Unsigned Binary |

# Box- Connected to NumbersBitMap




| Parameter | Value | Type |
|---|---|---|
| OBJECT_WIDTH_X | 16 | Signed Integer |
| OBJECT_HEIGHT_Y | 32 | Signed Integer |
| OBJECT_COLOR | 11100011 | Unsigned Binary |

***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 12:45***

## 4.7 תרגול שני של שימוש בנתח הלוגי, ה- SIGNAL_TAP

**משימה**: לבדוק בעזרת הנתח את כיוון התנועה לפני ואחרי ההתנגשות

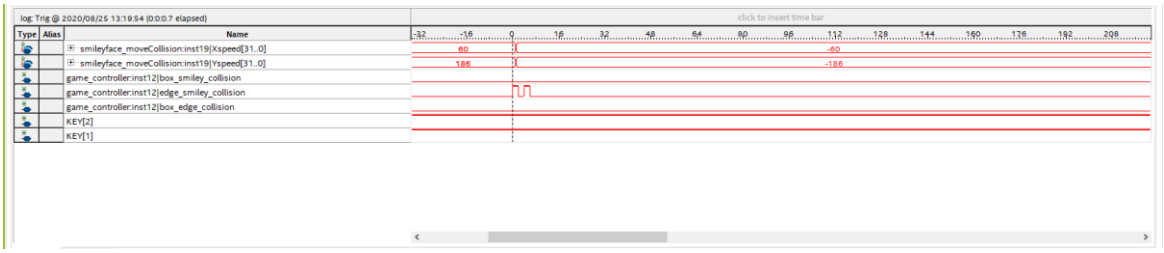
- הפעל שוב את הנתח הלוגי .
- הצג בנתח הלוגי את האותות שנראים לך רלוונטים במקרה זה.
- הצג גם אותות פנימיים של מודול למשל SPEEDY SPEEDX
- בחר RADIX נכון כולל SIGNED INT כשצריך
- הגדר טריגר על מנת לדגום את האותות סביב רגע ההתנגשות

בחן את השפעת שני מקשי המקלדת על תנועת הסמיילי.

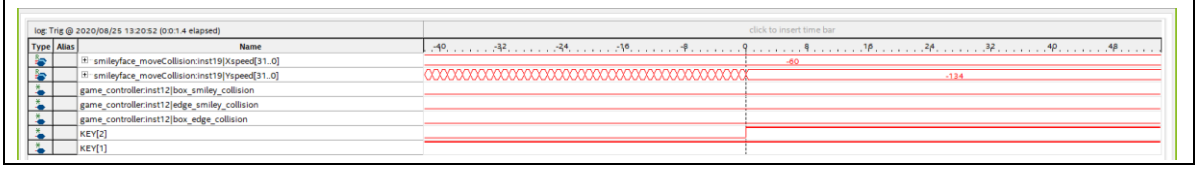
באחד מהם הפעולה אינה "נקייה" הסבר את התופעה ומדוע היא מתחוללת, מדוע היא לא קיימת בציר השני. (אין צורך לתקן את הבעיה)

כמו שניתן לראות בתרשים הבא, כאשר KEY[2] למטה, מתבצע שחלוף רב של ערכי מהירות Y, בניגוד ללחיצה ארוכה על KEY[1], זה קורה כי אנו לא עובדים דרך אמולטור מקלדת שמבצע הרכשה של האותות כניסה לסיגנל "נקי", אלא ישירות מול "הברזלים".

רכישה סטנדרטית




KEY[2] לא נקי




***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 13:30***

# 5 עבודה על הפרוייקט إستفتاح -סיפתח

- שב עם המדריך לדון על השקף המתאים מהמצגת עבור סכמת המלבנים הכללית של הפרויקט שלך
- שנה את הקוד כך שיתאים לתנועת אחד האובייקטים בפרוייקט שלך

- למשל - נוון את תנועת הסמילי כך שינוע רק מימין לשמאל ( רק בציר X) , נוון את התנועה ב Y ובכל פעם שמתנגש בקצה הימני, משנה את מיקומו לקצה השמאלי

מלא את הסעיפים הבאים ולאחר מכן העתק אותם לדוח המסכם של הפרויקט (במודל)

### 5.1 מטרות הספתח

רשמו כאן מה אתם מצפים להשיג מהספתח

| |
|---|
| אנו רוצים לעשות את הדברים הבאים: |

- לשנות את רקע המשחק לצורה מתאימה,
- לשנות את תנועת הסמיילי להיות נשלטת ע"י החזקה של מקשי הFPGA,
- לשנות את האות לזוז רנדומלית מלמעלה למטה בצורה רציפה

### 5.2 תיאור הספתח

שימו כאן צילום של ה TOP שביצעתם במעבדה

| |
|---|
| מדיבור עם אברהם זה יושלם לבד לא בתוך המעבדה, אלא במהלך הימים הקרובים. התחלנו בלבצע את שינוי הרקע והתנועה במעבדה. |

### 5.3 דיון ומסקנות עם המדריך

רשמו כאן את עיקרי הדברים, ודגשים חשובים להמשך העבודה, מה אתם הולכים לעשות עד המעבדה הבאה

| |
|---|
| להגיע לשלב של משחק מנוון לפחות- עצים שמתקדמים לעברנו(לפחות אחד), אופנוע אחד שזז רנדומית על המסך. |
| ***קרא למדריך, רשום את השעה בה הוא ראה את המעגל: 14:04*** |

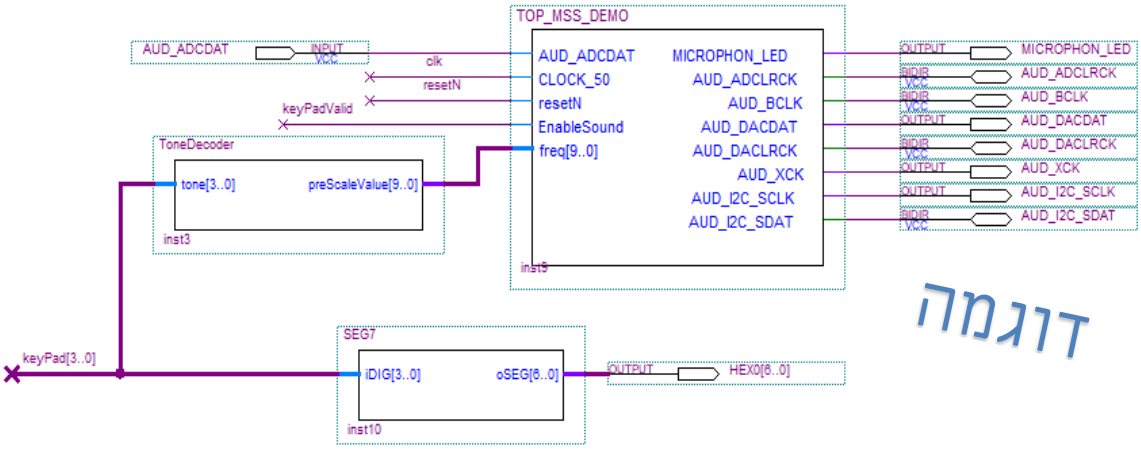
## 6 גיבוי

- שמור את הפרוייקט כארכיב QAR והעלה למודל – הוא ישמש כבסיס לפרוייקט שלך
- שמור דוח זה כ PDF והעלה למודל

## 7 הרחבת רשות: הוספת צלילים

**<u>משימה</u>**: להפעיל את הצלילים של היישום על ידי הפעלת ממשק השמע.

בשלב זה נתמקד בממשק השמע, המודול TOP_MSS_DEMO. פירוט על אופן פעולתו מופיע בחומר הרקע למעבדה זו. בקצרה, יישום זה משתשמש בפלטפורמת ה- MSS של המערכת ומייצר אות דיגיטלי אותו המערכת ממירה לאות אנלוגי אותו ניתן לשמוע באוזניות או רמקולים. במקרה זה האות הוא סינוס דיגיטלי בתדר שהמשתמש יכול לשנות וכך לקבוע את הצליל הנשמע.



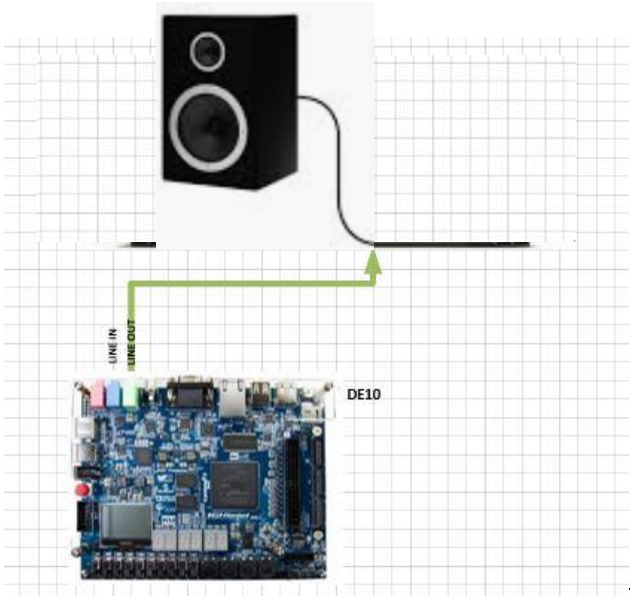

התבונן במודולים של הקובץ TOP_MSS_DEMO וזהה את הרכיבים הנ"ל.

- התבונן בחיבורים אל ממשק השמע ונסה להבין כיצד המשתמש משנה את הצליל הנשמע.

| |
|---|
| הסבר כיצד המשתמש יכול לשנות את הצליל הנשמע.<br>**<u>תשובה</u>** : |

חבר את יצאת השמע של הכרטיס אל הרמקולים של המסך (כבל חיבור קיים על שולחן המעבדה שלך) או לאוזניות או רמקולים חיצוניים.

**<u>שים לב</u>**: השליטה בעוצמת השמע היא דרך שני הכפתורים השמאליים של המסך הימני שבמעבדה.




הורד את היישום לכרטיס ובדוק שהוא עובד נכון. השמע צלילים שונים על ידי שינוי התדר.

*קרא למדריך, רשום את השעה בה הוא ראה את המעגל:*

*רשום את השעה בה סיימת את המעבדה:*