

הטכניון - מכון טכנולוגי לישראל  
הפקולטה להנדסת חשמל



מעבדה 1

MSS  
שאלות ודו"ח הכנה  
DE-10

גרסה 1.45

קיץ 2018

מחברים: אלכס גרישנפון

	תאריך הגשת דו"ח ההכנה
	שם המדריך

שם פרטי	שם משפחה	סטודנט
ברק	זן	1
בועז	טייטלר	2

## תוכן עניינים

1	מימוש מונה כתובות.....	2
2	מימוש מיישר חצי גל, גל שלם .....	3
3	שימוש בטבלאות למימוש פונקציה מורכבת .....	7
3.1	שימוש בטבלה ב VHDL .....	6
4	DPR .....	8

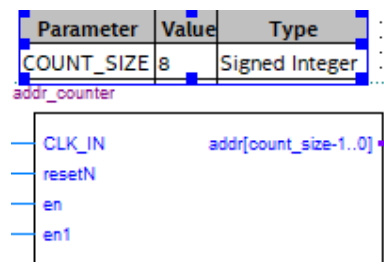
## הנחיות

- קובץ זה הוא גם התבנית לדו"ח המכין, יש לשמור ב PDF ולהגיש במודל.
- במעבדה זו נבחן מימושם שונים של כניסות ויציאות אנלוגיות ועיבוד אותות, בחלקים מהם נשתמש בפרויקט לצורך יצירת צלילים.

## 1 מימוש מונה כתובות

- ✓ תכנן מונה לפי התיאור הבא: הרכיב **addr\_counter** הוא מונה 8 ביט שמספק את 256 הכתובות. המונה יהיה סינכרוני ועבוד לפי עלית שעון, יספור אך ורק כאשר גם כאשר **en** וגם **en1** מקבלים '1'. כרגיל, **resetN** הינה כניסת איפוס א-סינכרונית הפעילה בנמוך. השתמש בקובץ **addr\_counter.vhd** שנמצא ב moodle כחלק מהפרויקט.
- ✓ שימו לב: אורך וקטור היציאה (המוגדר כעת ל-8 ביטים) הינו גודל גנרי וניתן לשינוי. בתכן שלכם עליכם להתייחס לגודל גנרי כללי כך שגם אם נשנה את אורך וקטור היציאה, ל-16 ביטים למשל, המונה יעבוד כהלכה. לשם כך, ניתן להשתמש בכך שכאשר וקטור בגודל מסויים מנסה לגדול מעבר ערך המקסימלי שלו (ב-8 ביטים- 255) אזי מתבצע wraparound והוקטור חוזר לערך 0. כך למשל, כאשר המונה יהיה "11111111" וננסה להוסיף לו 1, נקבל "00000000".

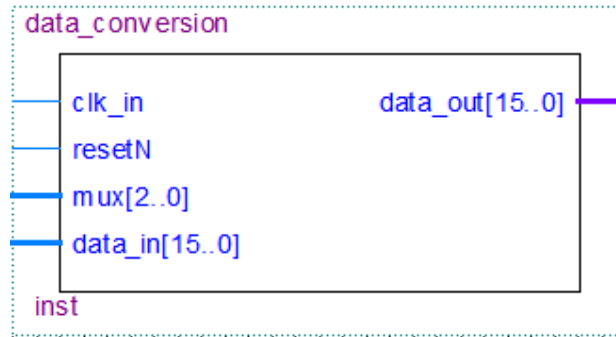
CLOCK	50	resetN	en	en1	count[COUNT_SIZE..0]
x	0	x	x	x	0000
↑	1	0	1	1	count
↑	1	1	1	1	count+1
↑	1	1	1	0	count



## 2 מימוש מיישר חצי גל, גל שלם

בניסויים האנלוגיים, ראינו ונראה שימושים למגברי שרת במגברים הופכי מופע, מעגלי יישור חד דרכי ודו דרכי, מעגלי קטימה ועוד.

כתוב ב-VHDL יחידה (Entity+Architecture) אשר מממשת באופן סיפרתי "מעבד אותות ספרתי". שמור את הקובץ בשם `data_conversion.vhd`.



היחידה מקבלת 16 ביט בכניסה ומוציאה 16 ביט לאחר עיבוד. לא לשכוח עובדים ב 2s complement



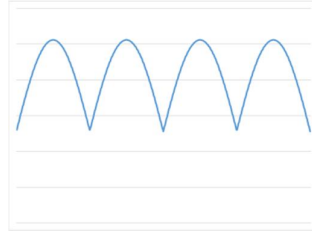
המערכת תהיה סינכרונית – ותוציא את המידע אחרי השהיה של שעון אחד, שים לב לשימוש נכון ב SIGNALS וב VARIABLES

למיישר כניסת פיקוד: `mux[2..0]` בעלת מספר מצבים

הכניסות והיציאות למיישר תהינה אותות בינריים של 16 ביט בייצוג signed integer.

כל התכן חייב להיות סינכרוני ובעל כניסת איפוס אסינכרונית הפעילה בנמוך.

תאור	שרטוט	כניסת	description
		SW4- SW2 MUXה	
אות כניסה, מהמחולל אמפליטודה +/- 400mV תדר 1000 Hz			
אות יציאה זהה לאות כניסה Bypass		000	Bypass מעביר את האות ללא שינויים

<p>Half Wave - מיישר חצי  גל. רק חלק חיובי, כל ערך  שלילי <math>= 0</math></p>	001		<p>אות יציאה. מעביר חלק  חיובי  Half wave</p>
<p>Invert  הופך את האות. (מכפיל  ב -1)</p>	010		<p>אות יציאה. הופך מופע  invert</p>
<p>Full Wave - מיישר גל  מלא. חלקים שלילים  יהפכו להיות חיוביים לפי  complement s2</p>	011		<p>אות יציאה. מעביר  חלק חיובי והופך  שלילי full wave</p>
<p>Quantization-Two-  מאפסת את שתי הסיביות  התחתונות של המספר כך  שהרזולוציה קטנה יותר )  המספר 0x4777 יהפוך ל  ( 0x4774</p>	100		Quantization-Two
<p>Quantization-  ELEVEN - מאפסת את  10 הסיביות התחתונות  של המספר.  ( המספר 0x4777 יהפוך  ל 0x4400 )</p>	101		Quantization- Eleven.
<p>HALF - מחלקת בשנים  את המספר כך שהתחום  הדינמי קטן בחצי .  ( המספר 0x4300 יהפוך  ל 0x2180 )</p>	110	$Y = x / 2$	HALF - מחלקת בשנים

<p>- OneFiftyPrecent</p> <p>מגדיל את המספר ל 150% מערכו שהתחום הדינמי גדל פי אחד וחצי במידת הצורך המעגל נכנס ל"רוויה".</p> <p>(המספר 0x3480 יהפוך ל 0x4EC0</p>	111	$Y = x * 3/2$	<p>OneFiftyPrecent -</p> <p>מגדיל את המספר ל 150% מערכו</p>
--	-----	---------------	---

```

use ieee.numeric_std.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;

```

```

ENTITY data_conversion IS
  GENERIC ( COUNT_SIZE : INTEGER := 16);
  PORT (
    CLK_IN      : IN STD_LOGIC;
    resetN      : IN STD_LOGIC;
    MUX         : IN std_logic_vector(2 downto 0);
    data_in     : IN std_logic_vector(COUNT_SIZE - 1 downto 0);

    data_out    : out std_logic_vector(COUNT_SIZE - 1 downto 0)
  );
END data_conversion;

```

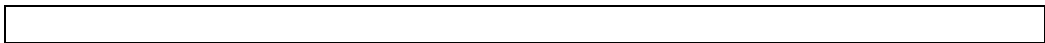
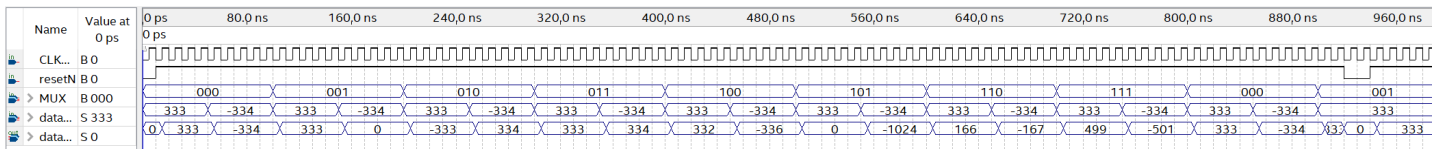
```

architecture arch_data_conversion of data_conversion is
begin
  process(CLK_IN, resetN)
  begin
    if resetN = '0' then
      data_out <= (others => '0');
    elsif rising_edge(CLK_IN) then
      case MUX is
        when "000" =>
          data_out <= data_in;
        when "001" =>
          if data_in < 0 then
            data_out <= (others => '0');
          else
            data_out <= data_in;
          end if;
        when "010" =>
          data_out <= (not data_in) + 1; -- inverse
        when "011" =>
          if data_in < 0 then
            data_out <= (not data_in) + 1;
          else
            data_out <= data_in;
          end if;
        when "100" =>
          data_out(COUNT_SIZE - 1 downto 2) <= data_in(COUNT_SIZE - 1 downto 2);
          data_out(1 downto 0) <= (others => '0');
        when "101" =>
          data_out(COUNT_SIZE - 1 downto 10) <= data_in(COUNT_SIZE - 1 downto 10);
          data_out(9 downto 0) <= (others => '0');
        when "110" =>
          data_out(COUNT_SIZE - 1 downto 0) <= data_in(COUNT_SIZE - 1) &
            data_in(COUNT_SIZE - 1 downto 1);
        when "111" =>
          data_out(COUNT_SIZE - 1 downto 0) <= data_in(COUNT_SIZE - 1 downto 0) +
            (data_in(COUNT_SIZE - 1) & data_in(COUNT_SIZE - 1 downto 1));
      end case;
    end if;
  end process;
end arch_data_conversion ;

```

בצע סימולציה והראה שכל אחד מהמצבים עובד. צרף את תוצאות הסימולציה של כל אחד מהמצבים לדו"ח ההכנה (יש להביא את הקובץ למעבדה על מנת לבדוק שהתכן עובד על גבי הפלטפורמה).

שימו לב שכל החישובים נעשים ב  $2s$  complement!



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



### 3 שימוש בטבלאות למימוש פונקציה מורכבת

לפעמים הדרך הפשוטה לממש פונקציה מסובכת היא על ידי טבלה של קבועים שמכילים מראש בחומר העזר במודל נתון קובץ בשם `SinTable.vhd` בו מופיעה טבלה לאות סינוס הממירה את מספר השורה לערך של סינוס הזווית.

דוגמה לבנית טבלה:

כיוון שאין floating point ב VHDL כל הערכים מנומרים למספרים שלמים. סינוס הזווית מבוטאת במספר בין 0x0000 ובין 0x3E80, משורה ראשונה עד שורה 256, והמוצא מופיע ב-SIGNED INT.

החישובים הדרושים לבניית האות נעשים בקובץ אקסל, `Sinetable.XLSX`, הנתון גם. הטבלה בקובץ ה-VHDL נוצרה על ידי העתקת העמודה המסומנת בצהוב, הנותנת את ערכי אות הסינוס ב-*hexadecimal* מטבלת האקסל. ראה את הנוסחאות המשמשות ליצירת הטכסט בקובץ האקסל הנתון.

#### 3.1 שימוש בטבלה ב VHDL

במודל מופיע מימוש מלא של המעגל מלבד הקוד של מונה הכתובת אותו יש עליכם לכתוב, כמתואר בסעיף 1, ולצרף אותו לתכן המוכן שקיבלתם. במונה זה תשתמשו גם בניסוי שתעשו במעבדה.

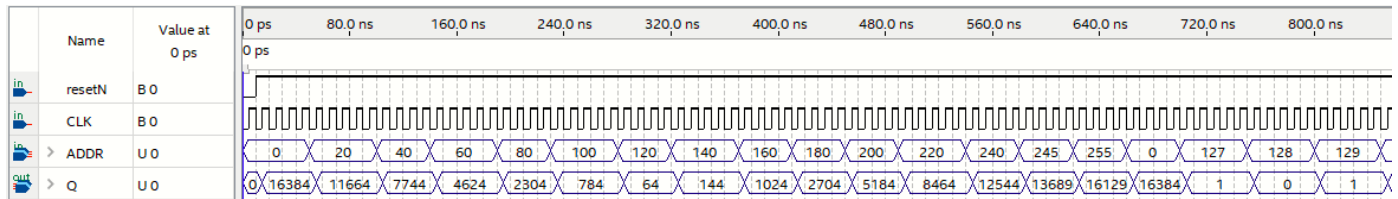
עליך לכתוב ב-VHDL יחידה (Entity Architecture) בשם `SquareTable.vhd` אשר מממשת באופן סינרטי פונקציה של ערך חזקה שניה. הפונקציה מקבלת ערך **8 ביט** Signed int של X שערכו בין -1.0 ל-1.0. ומוציאה את הערך של  $X^2$ , (16 ביט), שערכו בין -1.0 ל-1.0.

בנה את הפונקציה המתאימה בקובץ אקסל, לפי דוגמת הסינוס. מה יהיו הערכים המנומרים עבור קובץ ה-VHDL? רשום להלן את 20 הערכים הראשונים. (המתחילים ב-1.0 = X)

תשובה תשובה:

הכניסות יהיו בין -128 ל-128 (שולחים כתובת בין 0 ל-125), והיציאות בין 0 ל-16384  
"X"3931,"X"3A24,"X"3B19,"X"3C10,"X"3D09,"X"3E04,"X"3F01,"X"4000  
,"X"3751,"X"3840

התאם את הקוד שלך בקובץ `SquareTable.vhd` לפי דוגמת הסינוס על ידי בחירת העמודה המתאימה והעתקתה לקובץ ה-VHDL. סמלט אותו על ידי הכנסת ערכים בכניסה (למשל כל ערך עשירי) והוסף את הסימולציה לדו"ח.



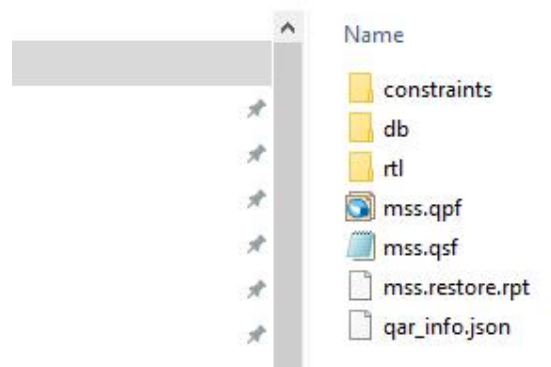
שמור את הקבצים והבא אותם למעבדה על מנת לבדוק שהתכן עובד על גבי הפלטפורמה.

## DPR 4

השתמשו במודול recorder\_module.bdf אשר נמצא בmoodle בקובץ mss\_students.qar :

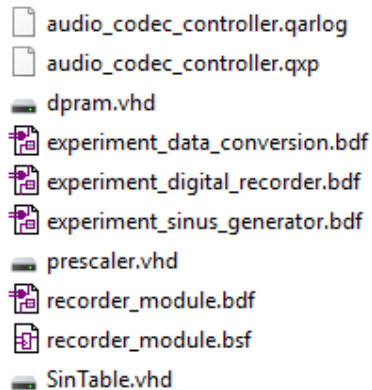
✓ פתח מהמודל את הפרויקט mss\_students.qar ובחר לאן לחלץ את קבצי הפרויקט.

mss\_students\_restored



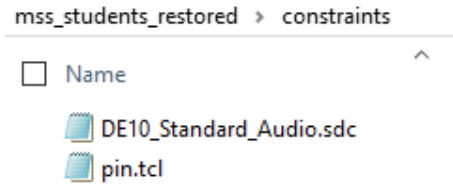
> MSS-DE10 > mss\_students\_restored > rtl

Name



✓ קובץ הפינים בשם pin.tcl נמצא ב :



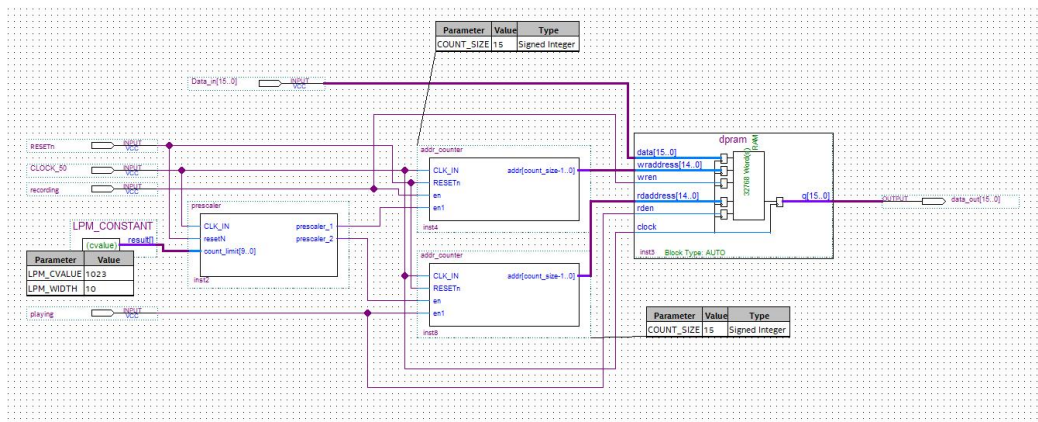


- ✓ העתק לתיקיה rtl את הקובץ addr\_counter.vhd שתכנתת בעבודת הכנה.
- ✓ עבור לתפריט Project Navigator – Files והגדר את הקובץ recorder\_module.bdf
- ✓ הנמצא בתיקיה rtl כ- TOP-Level Entity - (CTRL-SHIFT-V)
- ✓ קמפל – (מספיק להריץ רק סינטזה)



דרוש לבנות מעגל השהיה, שייקח דגימות של אות ויוציא אותן מושהה וללא שינוי צורה ביציאה :

כדי ליצור השהייה מקסימאלית בין אות הכניסה לבין אות היציאה. לרשותך התקן זיכרון מסוג *Dual Port RAM* בעל גודל קבוע של N מילים. למעגל שתי כניסות מפסקים READ ENABLE ו- WRITE ENABLE שכאשר שניהם הם ב-'1' המונים מתקדמים (מאופשרים).



שים לב שליחידה PRESCALE יש שתי יציאות בהפרש של שעון, הדבר נועד למנוע מצב של קריאה וכתובה בו זמנית מהזיכרון, דבר שיהיה בעייתי אם שתי הכתובות הן זהות. שנה את ערך ה- LPM\_CVALUE של prescaler ל- 63.

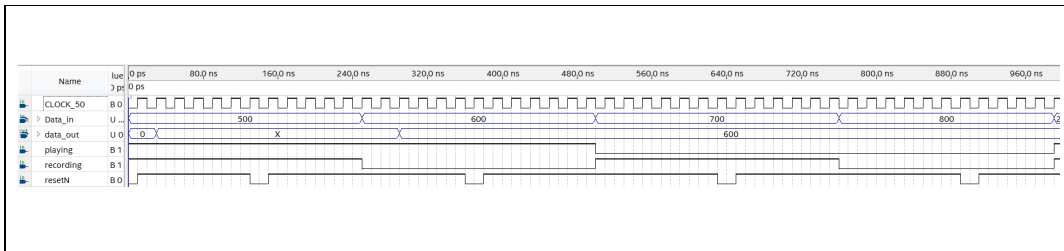
סמלץ את המעגל, וודא שהוא מתפקד כראוי.



באיזה אופן תכתוב לזיכרון ותקרא ממנו על מנת לקבל השהייה מכסימלית.

תשובה: על מנת לקבל השהייה מקסימלית ננצל את התכונה של הזכרון לזכור עד N מילים. תחילה נכתוב את ה-N מילים הראשונות, ואז ננסה לסנכרן בין קריאה מהזכרון לבין כתיבת המילה הבאה מאות הכניסה לתוך הזכרון. בסופו של דבר נקבל השהייה של N מילים או N עליות שעון.

במודל מופיע מימוש מלא של המעגל, עליך רק לבצע לו סימולציה חכמה שתבדוק את מצבי המערכת סמלט את המעגל, הכנס מידע משתנה בכניסה שנה את כניסות ה ENABLE וודא שהמעגל מבצע את פעולת ההקלטה. הוסף את תוצאות הסימולציה הבוחנות את כל המקרים ומראות שהמעגל אכן מתפקד כראוי:



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

**מלא את הטופס**