

הטכניון - מכון טכנולוגי לישראל
הפקולטה להנדסת חשמל



מעבדה 1א'
044157

ניסוי SV1
שאלות ודוח הכנה

גרסה 2.57
קיץ 2020

על פי חוברות של עמוס זסלבסקי, 2009

13/08/2020	תאריך כתיבת הדו"ח
אלון מזרחי	שם המדריך

שם פרטי	שם משפחה	סטודנט
ליאור	דביר	1
נועם	אילתה	2

תוכן עניינים

1	פתיחת ארכיב	3
2	תרגיל תכנון MUX בשיטות שונות	3
2.1	מימוש MUX באמצעות IF – קומבינטורי	3
2.2	מימוש MUX באמצעות CASE – קומבינטורי	6
3	תרגיל תכנון MUX הירארכי	7
4	מונה עולה סינכרוני	8
5	מונה סינכרוני עם קפיצות	9
6	תצוגת 7Segment עם הדלקה וכיבוי מלאים	11
7	גיבוי העבודה	14

הערות חשובות :

1. בכל התרגילים הבאים השפה לכתובת הקוד היא System Verilog או בקיצור SV
2. בכתובת הקוד חובה להשתמש בשמות המודולים, הכניסות והיציאות המופיעים בהגדרת התרגילים
3. יש לתת שמות קבצים ותיקיות באנגלית בלבד וללא רווחים וללא מקף -
4. שם הקובץ צריך להיות כשם המודול
5. תמיד יש להגדיר את הקובץ שעליו עובדים כהירארכיה עליונה או בקיצור כ- TOP
6. יש להשלים את הקוד שלך לפי הדרישות בקבצי השלד הנתונים במקומות המסומנים ב- `// fill your code here`
7. בסוף התהליך יש להעתיק את הקוד בצורה קריאה דרך ה- NOTEPAD++ לקובץ זה במקומות המסומנים לכך
8. בכמה מקבצי השלד הנתונים היה צורך לסגור חלק של הקוד כדי לעבור קומפילציה. לכן לפני שמתחילים לכתוב קוד יש להסיר את ההערות המסומנות ב-

`/* $$$$$$ remove to fill`

ולהשלים את הקוד שלכם במקומות המסומנים ב-

`//fill your code here`

1 פתיחת ארכיב

הורד מהמודל קובץ ארכיב של המעבדה ופתח אותו לפרויקט בדיסק שלך.
ודא תכולת קבצים כזו:

File Name	Type
rtl/mux_4to1_if.sv	SystemVerilog HDL File
rtl/mux_4to1_case.sv	SystemVerilog HDL File
rtl/mux_16to1.sv	SystemVerilog HDL File
rtl/simple_up_counter.sv	SystemVerilog HDL File
rtl/jmp_counter.sv	SystemVerilog HDL File
rtl/hexss.sv	SystemVerilog HDL File
constraints/pins_test_sv.tcl	Tcl Script File
rtl/up_counter.sv	SystemVerilog HDL File
rtl/comparator.sv	SystemVerilog HDL File
rtl/inflating_counter.sv	SystemVerilog HDL File
rtl/inflating_cnt_top.bdf	Block Diagram/Schematic File
rtl/one_sec_counter.sv	SystemVerilog HDL File

הפרויקט שתתחיל אותו כעת בעבודת ההכנה תמשיך אותו במעבדה.

2 תרגיל תכנון MUX בשיטות שונות

שים לב! לכתובת קוד בתרגיל זה העזר בקבצים הנתונים לך במודל.

2.1 מימוש MUX באמצעות IF – קומבינטורי

השלם את הקוד במודול בשם mux_4to1_if. לשם כך פתח את הקובץ והפוך אותו ל-TOP. השלם את הקוד שלך לפי הדרישות להלן במקומות המסומנים:

```
./ fill your code here
```

כתוב את הקוד שמתאר את הרכיב באמצעות **התניית if**.
השתמש רק בהשמות BLOCKING. =

בצע אנליזה (Analysis & Elaboration) לתכן ותקן שגיאות סינטקס אם ישנן כאלה.

Module interface

Direction	Type	
input	logic	data_in[3:0]
input	logic	sel[1:0]
output	logic	outd

Truth table

data_in[3:0]	sel[1:0]	outd
--------------	----------	------

	00	data_in[0]
	01	data_in[1]
	10	data_in[2]
	11	data_in[3]

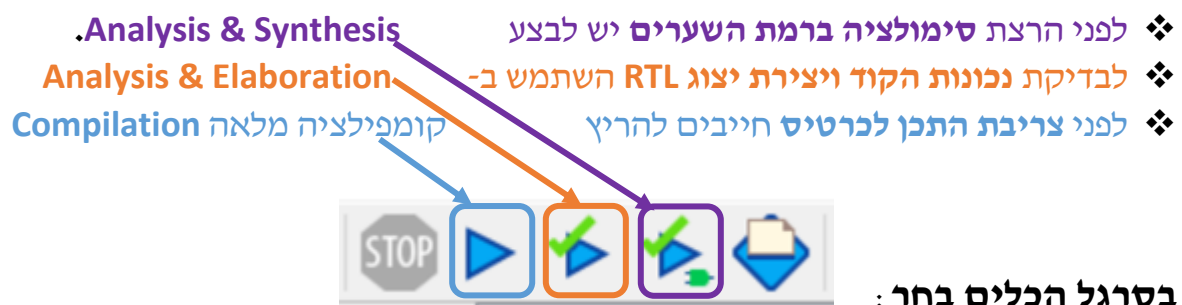
תזכורת לשלבי הקומפילציה השונים בקוורטוס והשימושים שלהם. כפי שהוסבר כל שלב יותר מתקדם מקודמו ומכיל את השלבים הקודמים, לכן גם אורך יותר זמן.

❖ **בשלב ההרחבה (Elaboration) – אלבורציה**, התכן נבדק מבחינה סינטקטית וסמנטית, ספציפית לשפת התכנות בה כותבים.

❖ **בשלב האנליזה (Analysis)** נבדק התכנון ההירארכי של התכן ונבנית טבלת קשרים בין האלמנטים השונים, עדיין כבלוקים ב-RTL (Register-Transfer Level).

❖ **בשלב הסינטזה (Synthesis)** הקוד הנתון מתורגם למשאבי החומרה בתוך ה-FPGA ברמת השערים הלוגיים מהם מורכבת החומרה. בשלב זה גם נוצרים ה-netlists שמאפשרים לבצע סימולציה (פונקציונלית) **בקומפילציה מלאה (Compilation)** מתבצעים כל השלבים הנוספים, הכוללים בין היתר תכנון הצורה והחיבוריות של החומרה, ניתוח זמנים, וגם הפקת קובץ הצריבה, הדרוש לשם צריבת התכן לכרטיס.

בהתאם לשלב הפיתוח יש להשתמש בפעולה המתאימה **זה חוסך המון זמן קומפילציה!**



אחרי אנליזה מוצלחת (**Analysis & Elaboration**) הוסף את הקוד שלך לדו"ח.

```

module mux_4to1_if
(
  input logic [3:0] datain,
  input logic [1:0] sel,
  output logic outd
);

  always_comb
  begin

    if (sel == 0) begin

```

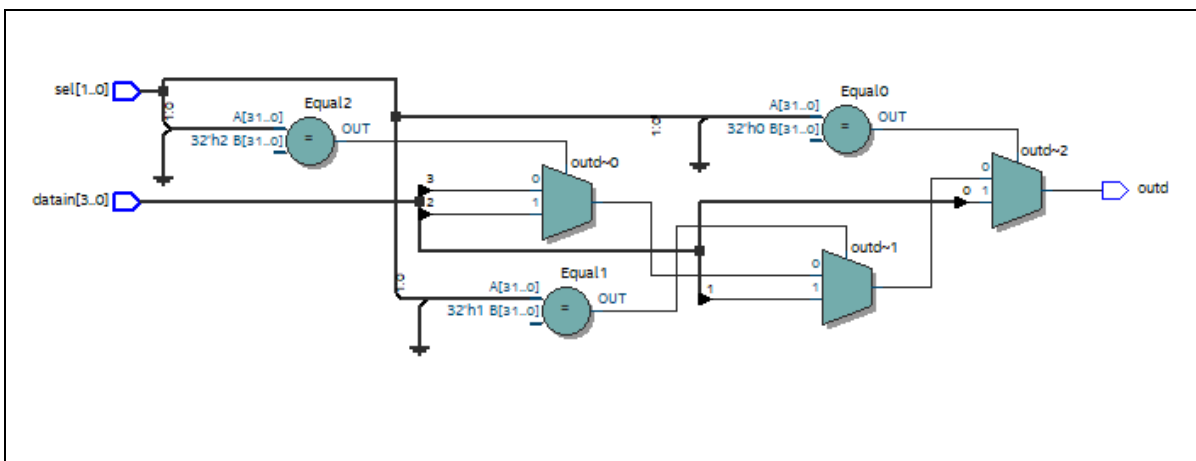
```

        outd = datain[0];
    end
    else if (sel == 1) begin
        outd = datain[1];
    end
    else if (sel == 2) begin
        outd = datain[2];
    end
    else begin
        outd = datain[3];
    end
end

end
endmodule

```

על מנת לבדוק באופן חזותי/גרפי את תכנון המודול שכתבת הצג את המימוש שלך כ-
 RTL VIEW (Tools -> Netlist Viewers -> RTL Viewer) והוסף אותו לדו"ח (היעזר גם
 ב-Quartus Cook Book).



2.2 מימוש MUX באמצעות CASE – קומבינטורי

פתח את הקובץ בשם mux_4to1_case וקבע אותו כ- TOP. המודול הינו Multiplexer
Module interface

Direction	Type	
input	logic	data_in[3:0]
input	logic	sel[1:0]
output	logic	outd

Truth table

data_in[3:0]	sel[1:0]	outd
	00	data_in[0]
	01	data_in[1]
	10	data_in[2]
	11	data_in[3]

השלם את הקוד שמתאר את הרכיב באמצעות **CASE התניית** לא מסונכרנת שעון.
השתמש רק בהשמות BLOCKING.
בצע אנליזה והוסף את הקוד ואת המימוש כ- RTL VIEW לדו"ח.

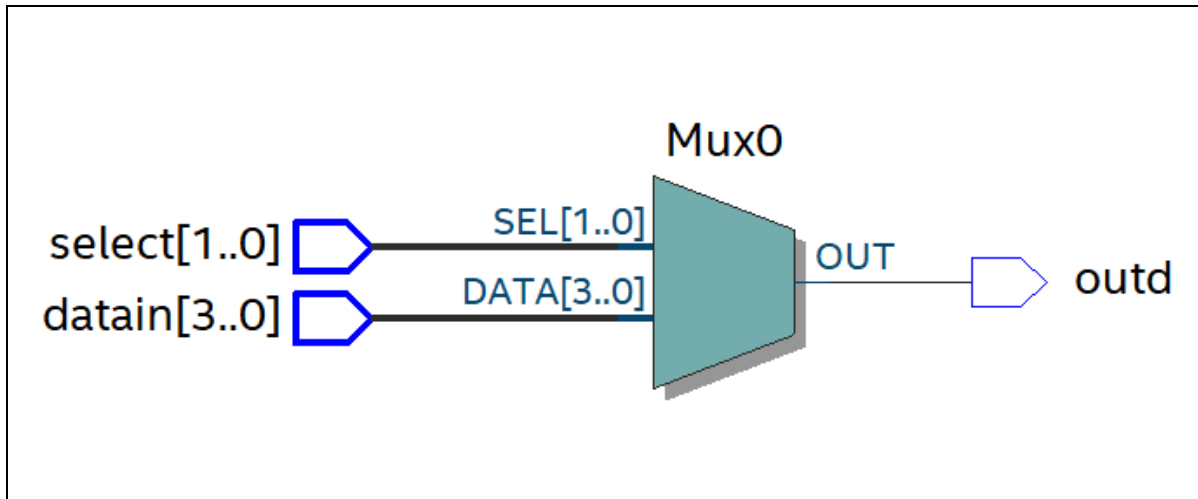
```
module mux_4to1_case
(
    input logic [3:0] datain,
    input logic [1:0] select,
    output logic outd
);

always_comb
begin

    case (select)
    0: outd = datain[0];
    1: outd = datain[1];
    2: outd = datain[2];
    3: outd = datain[3];
    endcase

end

endmodule
```



3 תרגיל תכנון MUX הירארכי

השלם את הקוד במודול בשם: mux_16to1. המודול מממש Multiplexer בעל 16 כניסות מידע din (וקטור באורך 16), 4 כניסות בחירה sel (וקטור באורך 4) ויציאת outd של ביט אחד.

המערכת מורכבת מ- 5 רכיבי 1 => 4 Multiplexer מהתרגיל הקודם.

ממש תכן הירארכי ב- Verilog על ידי שימוש במימוש עם case ובצוע instantiation (הפעלת המודול).

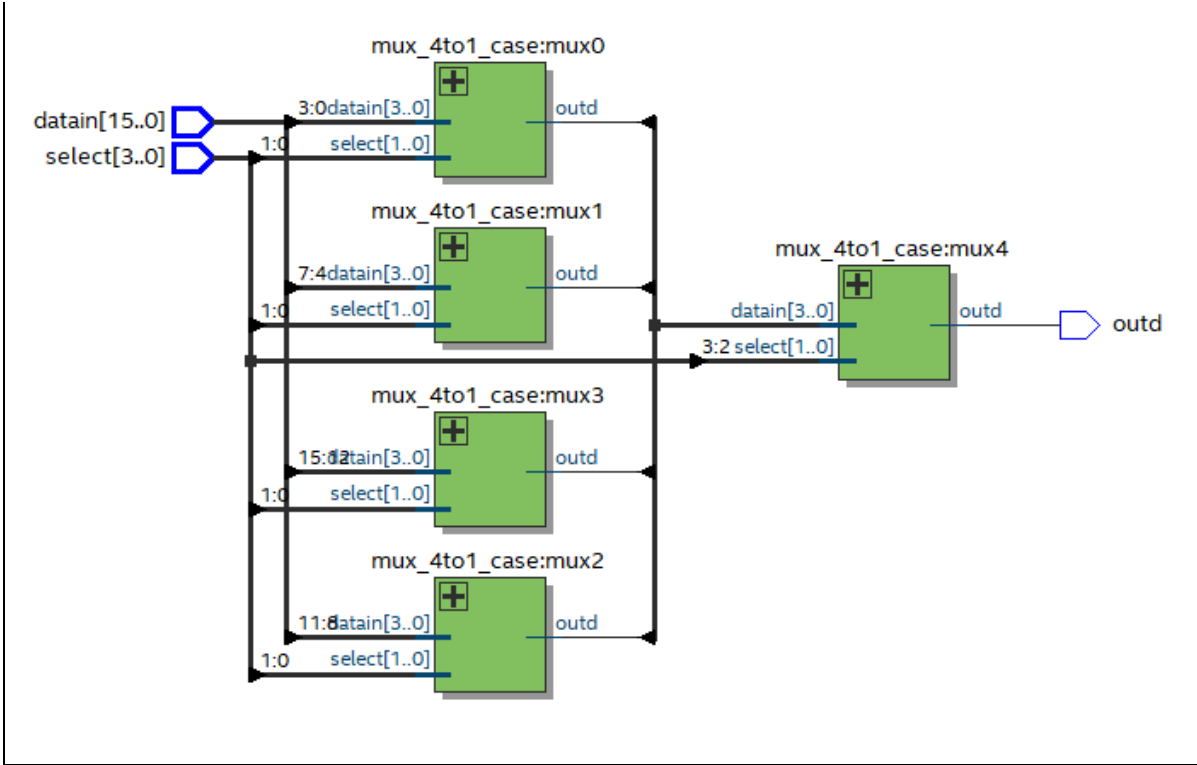
העזר בחומר העזר 1 Verilog workshop. בצע אנליזה והוסף את הקוד ואת המימוש כ- RTL VIEW לדו"ח.

```
module mux_16to1
(
    input logic [15:0] datain,
    input logic [3:0] select,
    output logic outd
);

    logic [3:0] muxi;

    mux_4to1_case mux0 ( .datain(datain[3:0]), .select(select[1:0]), .outd(muxi[0]));
    mux_4to1_case mux1 ( .datain(datain[7:4]), .select(select[1:0]), .outd(muxi[1]));
    mux_4to1_case mux2 ( .datain(datain[11:8]), .select(select[1:0]), .outd(muxi[2]));
    mux_4to1_case mux3 ( .datain(datain[15:12]), .select(select[1:0]), .outd(muxi[3]));
    mux_4to1_case mux4 ( .datain(muxi[3:0]), .select(select[3:2]), .outd(outd));

endmodule
```



4 מונה עולה סינכרוני

השלים את הקוד במודול של מונה בינארי סינכרוני עולה `simple_up_counter`.

Module interface

Direction	Type		Name
input	logic		clk
input	logic		resetN
output	logic	[3:0]	count

Truth table

CLK	resetN	count[3:0]	count next
x	0	4'b0000	4'b0000
↑	1	count	count+1

היות ובתרגיל זה יש להריץ גם סימולציה, כעת בצע סינתזה (Analysis & Synthesis).
הוסף את הקוד ואת המימוש כ- RTL VIEW לדו"ח.


```

module simple_up_counter
(
  // Input, Output Ports
  input logic clk,
  input logic resetN,
  output logic [3:0] count
);

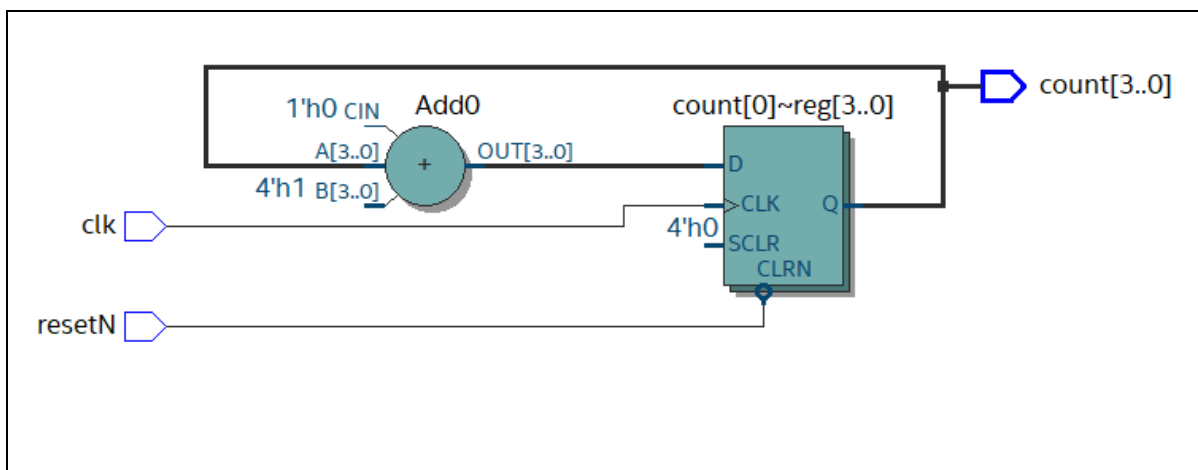
always_ff @( posedge clk or negedge resetN )
begin

  if ( !resetN ) begin // Asynchronous reset
    count <= 0;
  end else begin
    count <= count + 1;
  end

end

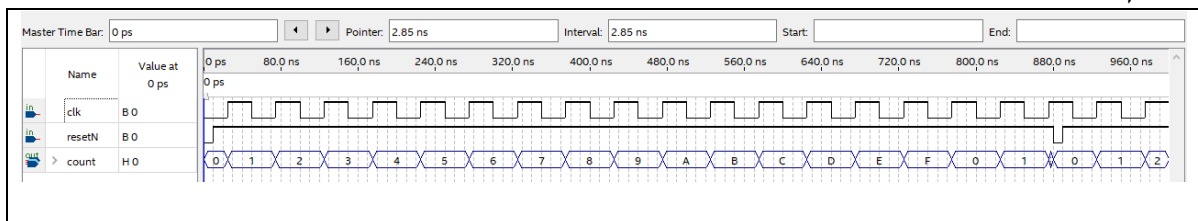
endmodule

```



צור קובץ WAVEFORM והרץ סימולציה של המעגל.

הוסף את תוצאות הסימולציה לדו"ח.



5 מונה סינכרוני עם קפיצות

השלם את הקוד במודול של מונה בינארי סינכרוני עולה `jmp_counter.sv`.

Module interface

Direction	Type		Name
input	logic		clk
input	logic		resetN
output	logic	[3:0]	count

Truth table

CLK	resetN	count[3:0]	count next
x	0	4'b0000	4'b0000
↑	1	count	count+1
↑	1	4'b0110	4'b1011
↑	1	4'b1111	4'b0000

הסופר עד 15 עם קפיצות: המונה מתחיל לספור מ-0, סופר עד 6, קופץ ל-11, ממשיך לספור עד 15, חוזר ל-0, ושוב סופר מ-0 עד 6, קופץ ל-11, סופר עד 15, מתאפס וכן הלאה ממשיך בצורה מחזורית.

בצע סינתזה והעתק את הקוד שכתבת לכאן.

```
module jmp_counter
(
  // Input, Output Ports
  input logic clk,
  input logic resetN,
  output logic [3:0] count
);

always_ff @( posedge clk or negedge resetN )
begin

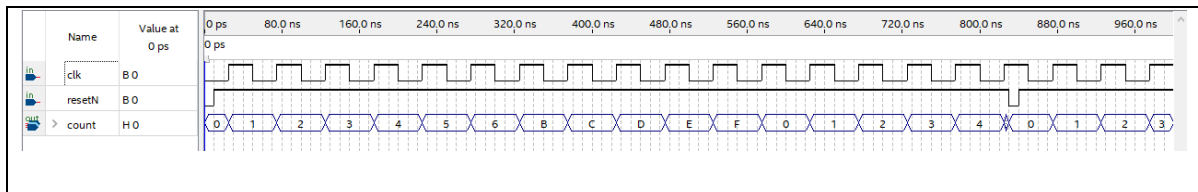
  if ( !resetN ) begin // Asynchronous reset
    count <= 0;
  end else if (count == 6) begin
    count <= 11;
  end else begin
    count <= count + 1;
  end

end

endmodule
```

צור קובץ WAVEFORM והרץ סימולציה של המעגל.

הוסף את תוצאות הסימולציה לדו"ח.



6 תצוגת 7Segment עם הדלקה וכיבוי מלאים

Module interface

Direction	Type	
input	logic	darkN
input	logic	LampTest
input	logic	hexin[3:0]
output	logic	ss[6:0]

Truth table

darkN	LampTest	hexin[3:0]	ss[6:0]
1'b0	x	4'hxx	7'b1111111
1'b1	1'b1	4'hxx	7'b0000000
1'b1	1'b0	4'h0	7'b1000000
1'b1	1'b0	4'h1	7'b1111001
1'b1	1'b0	4'h2	7'b0100100
1'b1	1'b0	4'h3	7'b0110000
1'b1	1'b0	4'h4	7'b0011001
1'b1	1'b0	4'h5	7'b0010010
1'b1	1'b0	4'h6	7'b0000010
1'b1	1'b0	4'h7	7'b1111000
1'b1	1'b0	4'h8	7'b0000000
1'b1	1'b0	4'h9	7'b0010000
1'b1	1'b0	4'hA	7'b0001000
1'b1	1'b0	4'hB	7'b0000011
1'b1	1'b0	4'hC	7'b1000110
1'b1	1'b0	4'hD	7'b0100001
1'b1	1'b0	4'hE	7'b0000110
1'b1	1'b0	4'hF	7'b0001110

בשאלה זו יהיה עליך לתכנן ממיר צירופי מקוד בינארי ברוחב 4 סיביות לתצוגת Seven Segment עבור כל 16 הצירופים האפשריים של 4 הסיביות. לממיר עוד 2 כניסות בקרה של סיבית אחת כל אחת, **LampTest** ו **darkN** אשר תפקידן נתון בטבלת האמת.

נתון לך שלד של רכיב בשם **HEXSS**, בו תשלים את הקוד שלך. רכיב זה יהיה שימושי בהמשך בניסויים הבאים ובפרויקט הסופי.

תצוגות Seven Segment בעלת המבנה המרחבי הבא :

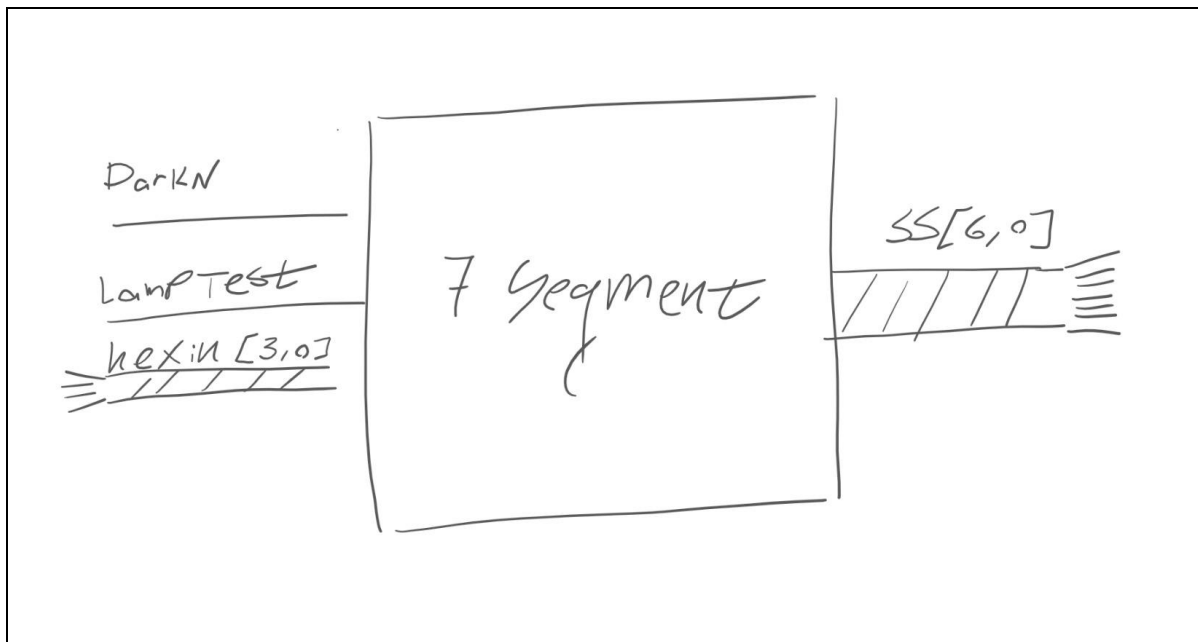


הערה: מותר להיעזר בקוד HEXSS שקיבלת במעבדה סכמת 1 בתור דוגמה

יש לממש את טבלת ההמרה בעזרת מערך דו ממדי, אותו אפשר להגדיר כך:
ולא להשתמש בפקודת CASE.

```
logic [0:15] [6:0] SevenSeg =  
    {  
        7'h40, //0  
        7'h79, //1  
        7'h24, //2  
        7'h30, //3  
        7'h19, //4  
        7'h12, //5  
        7'h02, //6  
        7'h78, //7  
        7'h00, //8  
        7'h10, //9  
        7'h08, //10  
        7'h03, //11  
        7'h46, //12  
        7'h21, //13  
        7'h06, //14  
        7'h0e, //15  
    };
```

סרטט בעפרון סמל גרפי של הרכיב (כניסות ויציאות)



בצע סינתזה והעתק את הקוד שכתבת לכאן.

```

module hexss
(
  input logic [3:0] hexin,          // Data input: hex numbers 0 to f
  input logic darkN,
  input logic LampTest,           // Additional inputs
  output logic [6:0] ss           // Output for 7Seg display
);

always_comb
begin
  logic [0:15] [6:0] SevenSeg =
  {
    7'h40, //0
    7'h79, //1
    7'h24, //2
    7'h30, //3
    7'h19, //4
    7'h12, //5
    7'h02, //6
    7'h78, //7
    7'h00, //8
    7'h10, //9
    7'h08, //10
    7'h03, //11
    7'h46, //12
    7'h21, //13
    7'h06, //14
    7'h0e, //15
  };

  if (!darkN) begin

```

```


        ss = 7'h7f;
    end else if (LampTest) begin
        ss = 7'h00;
    end else begin
        ss = SevenSeg[hexin];
    end

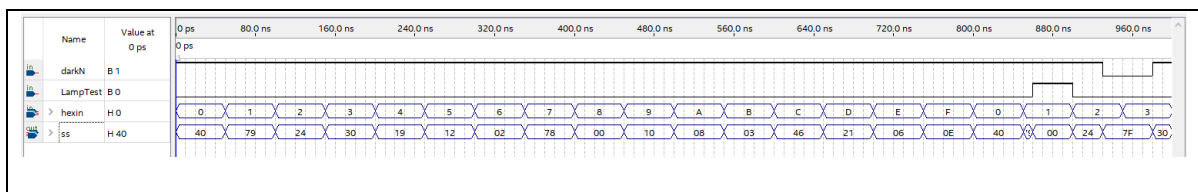
end

endmodule

```

צור קובץ WAVEFORM והרץ סימולציה של המעגל.

שים לב בסימולציה ניתן לקבוע בקלות את ערכי הכניסה hexin העוקבים 0 עד F בעזרת **כלי המונה, ה- Count Value**  **Simulation Waveform Editor**. ה-
הוסף את תוצאות הסימולציה לדו"ח.



7 גיבוי העבודה

שמור את הפרויקט רגיל וגם **כארכיב (באמצעות Project -> Archive Project)**.

פעולת הארכיב יוצרת קובץ עם סיומת *.qar. אותו תגבה, **העלה למודל והבא למעבדה** כי תצטרך אותו בניסוי.

באופן כללי יש לשמור, לגבות ולהביא למעבדה את כל קבצי הקוד והפרויקטים שכתבתם כי תשתמשו בהם במהלך הניסויים ובפרויקט הסופי.

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

מלא את הטופס

שמור דו"ח זה כ- PDF והעלה למודל

