הפקולטה להנדסת חשמל ומחשבים
ע"ש אנדרו וארנה ויטרבי

EEELabs
Electrical Engineering Elementary Labs
מעבדות יסוד בהנדסת חשמל

הטכניון
מכון טכנולוגי לישראל

מעבדה בהנדסת חשמל 1א'
044157

ניסוי SV1
דוח הכנה

גרסה 3

אביב תשפ"ב 2022

שם המדריך		סטודנט	שם פרטי	שפ משפחה
קבוצה – יום ומועד (בוקר/אחה"צ)	חמישי בוקר	1	עמיחי	שטרנליכט
תאריך ההגשה		2	יקיר	לוגסי

תוכן עניינים

הערות שחשוב לקרוא לפני תחילת העבודה:

1. בכל התרגילים הבאים השפה לכתיבת הקוד היא System Verilog או בקיצור SV.
2. בכתיבת הקוד חובה להשתמש בשמות המודולים, הכניסות והיציאות המופיעים בהגדרת התרגילים.
3. שם הקובץ צריך להיות כשם המודול.
4. יש להקפיד לתת שמות לתיקיות ולקבצים רק באנגלית (אותיות לועזיות וספרות) ללא סימנים וללא רווחים. ה- "_" (hyphen) מותר.
5. ה- PATH לקובץ צריך להיות קצר וגם הוא באנגלית לפי הכללים הנ"ל.
6. לכתיבת קוד בתרגיל זה יש להשתמש בקבצים הנתונים, הכלולים בקובץ הארכיב הנתון במודל. הם יופיעו בפרויקט שייפתח מקובץ הארכיב.
7. תמיד יש להגדיר את הקובץ שעליו עובדים כהיראכיה עליונה או בקיצור כ- TOP.
8. יש להשלים את הקוד לפי הדרישות וההנחיות בקבצי השלד הנתונים. בדרך כלל מקומות אלה המסומנים בהערה

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

9. בסוף התהליך יש להעתיק את הקוד **בצורה קריאה דרך ה- NOTEPAD++** לקובץ זה במקומות המסומנים לכך.
10. בכמה מקבצי השלד הנתונים היה צורך לסגור חלק מהקוד כדי לעבור קומפילציה. לכן לפני שמתחילים לכתוב קוד יש להסיר הערות מסומנות ב-

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

11. **ולהשלים את הקוד** שלכם במקומות המסומנים ב-

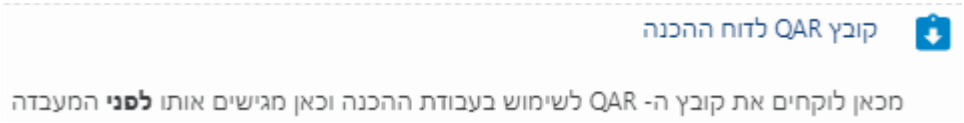
```
//fill your code here
```

12. לפני תחילת העבודה על הדוח **יש לפתוח** את שני קבצי העזר הבאים, הנתונים במודל באזור "מידע טכני כללי":
    - המדריך לשפת SV - "**System Verilog Cook Book**" שנותן מידע על כתיבת קוד בשפת SV.
    - המדריך "**Quartus17 Cook Book**" המפרט כיצד לעשות את הפעולות השונות בקוורטוס.

1 פתיחת ארכיב

מטרה: להוריד קובץ ארכיב מהמודל ולפתוח אותו לפרויקט.

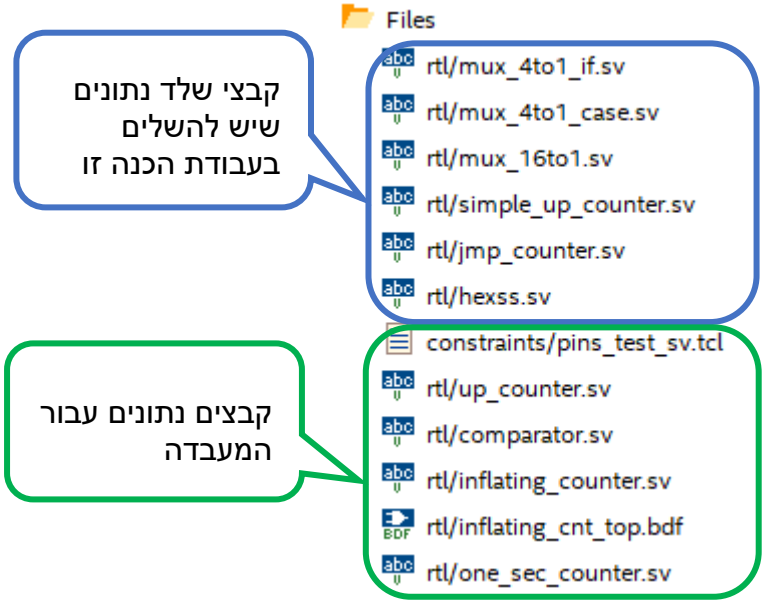
1. **צור תיקיה** למעבדה זו (אך ורק באנגלית). הורד מהמודל קובץ ארכיב של המעבדה ופתח אותו לפרויקט בתיקייה שיצרת.



2. יש לשנות את ה- PATH שמציע הקוורטוס ל- PATH קצר, אך ורק באנגלית (אותיות לועזיות וספרות) ללא סימנים וללא רווחים.

   ה- "_" (hyphen) מותר.

3. **וודא** תכולת קבצים כזו בפרויקט:



2 בניית MUX בשיטות שונות

מטרה: לממש בתוכנה רכיב Multiplexer 4->1 (או בקיצור MUX) בשתי שיטות: תחילה באמצעות משפט IF קומבינטורי ושנית באמצעות משפט CASE קומבינטורי.

הגדרות ה- Module interface וה- Truth table של ה- MUX נתונות להלן:

MUX - Module interface

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

MUX - Truth table

data_in[3:0]	select[1:0]	Outd
	00	data_in[0]
	01	data_in[1]
	10	data_in[2]
	11	data_in[3]

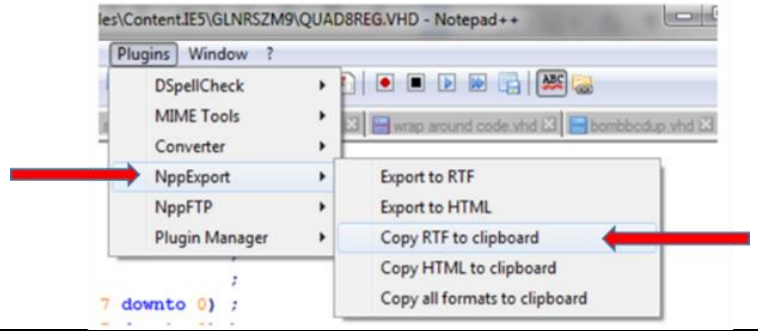
2.1 מימוש MUX באמצעות IF – קומבינטורי

מטרה: להשלים קוד בקובץ נתון למימוש MUX תוך שימוש רק בפונקצית IF - קומבינטורי.

1. **פתח את הקובץ** בשם **mux_4to1_if.sv והגדר** אותו כ- Top Level Entity (או בקצרה כ- TOP). מודול זה הינו שלד של רכיב Multiplexer.
2. **השלם את הקוד** שלך לפי הדרישות להלן במקומות המסומנים:

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

3. **כתוב קוד** שמתאר את פעולת הרכיב באמצעות התניית if . השתמש רק בהשמות מסוג BLOCKING =.
4. **בצע אלאבורציה (Analysis & Elaboration)** לתכן ותקן שגיאות סינטקס אם ישנן כאלה.
5. **העתק את הקוד** שלך לדוח כך שהעריכה הצבעונית תשמר, לשם כך השתמש בהעתקה **דרך ה- NOTEPAD++** (כפי שמוסבר להלן בקצרה).

	**<u>הערה</u>**: להעתקת טקסט צבעוני באמצעות ה- **NOTEPAD++** ראה הסברים מפורטים יותר ב- COOKBOOK.

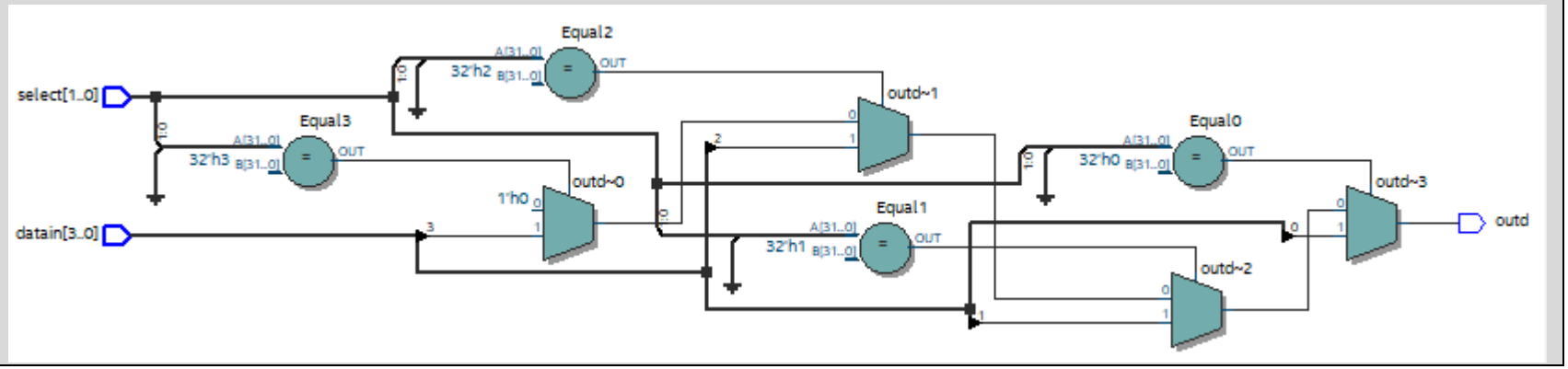
6. הוסף את הקוד של ה- MUX עם IF - קומבינטורי שהשלמת לדו"ח אחרי אלאבורציה מוצלחת.

```
3. module mux_4to1_if
4.  (
5.    input logic [3:0] datain,
6.    input logic [1:0] select,
7.    output logic outd
8.    );
9.
10.         always_comb
11.         begin
12.               outd = 0;
13.               if(select == 0) begin
14.                     outd = datain[0];
15.                     end
16.               else if(select == 1) begin
17.                     outd = datain[1];
18.                     end
19.               else if(select == 2) begin
20.                     outd = datain[2];
21.                     end
22.               else if(select == 3) begin
23.                     outd = datain[3];
24.                     end
25.               else begin
26.               end
27.         end
28.
29.   endmodule
```

```
endmodule
```

7. **הצג** את המימוש שלך כ- RTL VIEW (Tools -> Netlist Viewers -> RTL Viewer), על מנת לבדוק באופן גרפי את המודול בתוכנה.

8. **הוסף RTL VIEW של ה- MUX עם IF - קומבינטורי לדו"ח (היעזר גם ב- Quartus Cook Book). אפשר כתמונה דרך העתקת מסך.**



2.2 מימוש MUX באמצעות CASE – קומבינטורי

מטרה: להשלים קוד בקובץ נתון למימוש MUX תוך שימוש בהתניית CASE - קומבינטורי.

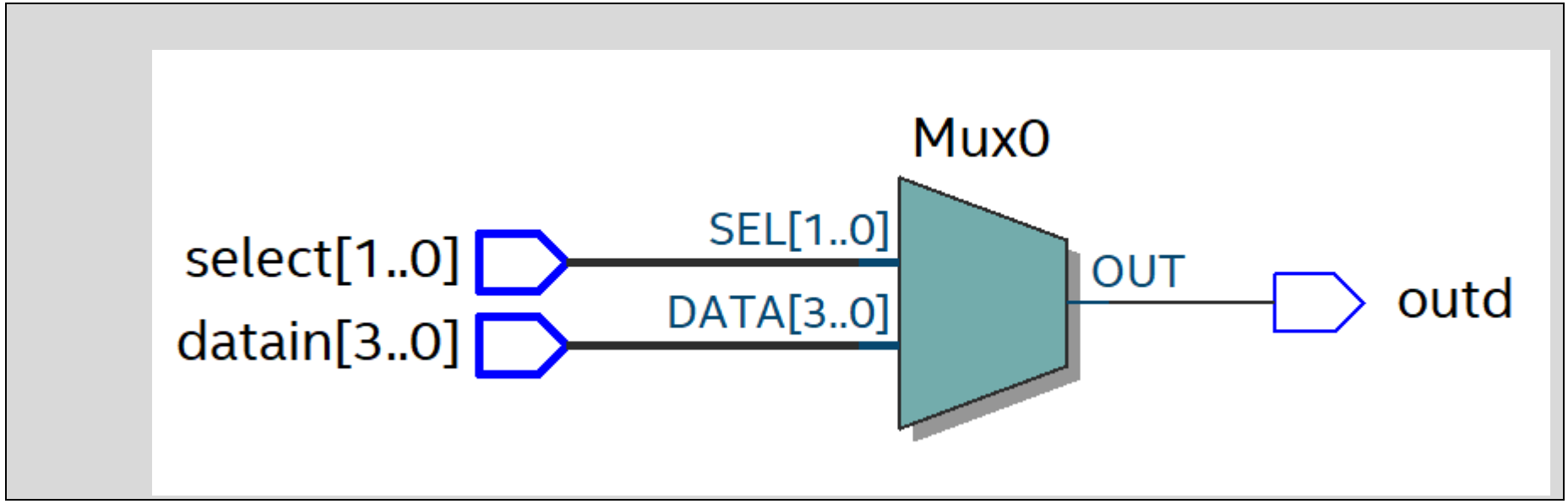
1. **פתח** את הקובץ בשם **mux_4to1_case.sv** **וקבע** אותו כ- TOP. גם מודול זה הינו Multiplexer.
2. **השלם את** הקוד שמתאר את הרכיב באמצעות התניית **CASE** קומבינטורי. השתמש רק **בהשמות BLOCKING.**
3. **הרץ** אלאבורציה.
4. **הוסף את הקוד לדו"ח.**

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

      always_comb
      begin
            outd = 0;
            case(select)
            0: begin
                  outd = datain[0];
                  end
            1: begin
                  outd = datain[1];
                  end
            2: begin
                  outd = datain[2];
                  end
            3: begin
                  outd = datain[3];
```

```
                end
            default: begin
                            end
            endcase
        end
```

```
endmodule
```

5. הוסף את המימוש כ- RTL VIEW של ה- MUX עם CASE - קומבינטורי לדו"ח.



3 בניית MUX הירארכי

מטרה: לבנות רכיב הירארכי, מסוג Multiplexer 16->1, תוך שמוש ברכיבי בסיס קיימים מסוג Multiplexer 4->1 (שנבנו קודם לכן).

הסבר והנחיות: נתון המודול בשם: **mux_16to1.sv**, שהינו שלד לרכיב Multiplexer בעל:

- 16 כניסות מידע din (וקטרו באורך 16),
- 4 כניסות בחירה select (וקטור באורך 4)
- ויציאת outd של ביט אחד.

ניתן לבנות רכיב כזה תוך שמוש ב- 5 רכיבי Multiplexer 4->1. לצורך כך אפשר ורצוי להשתמש ברכיבי ה- MUX שנבנו בתרגיל הקודם.

הערה: בכתיבת הקוד מומלץ להיעזר בחומר העזר Verilog workshop 1.

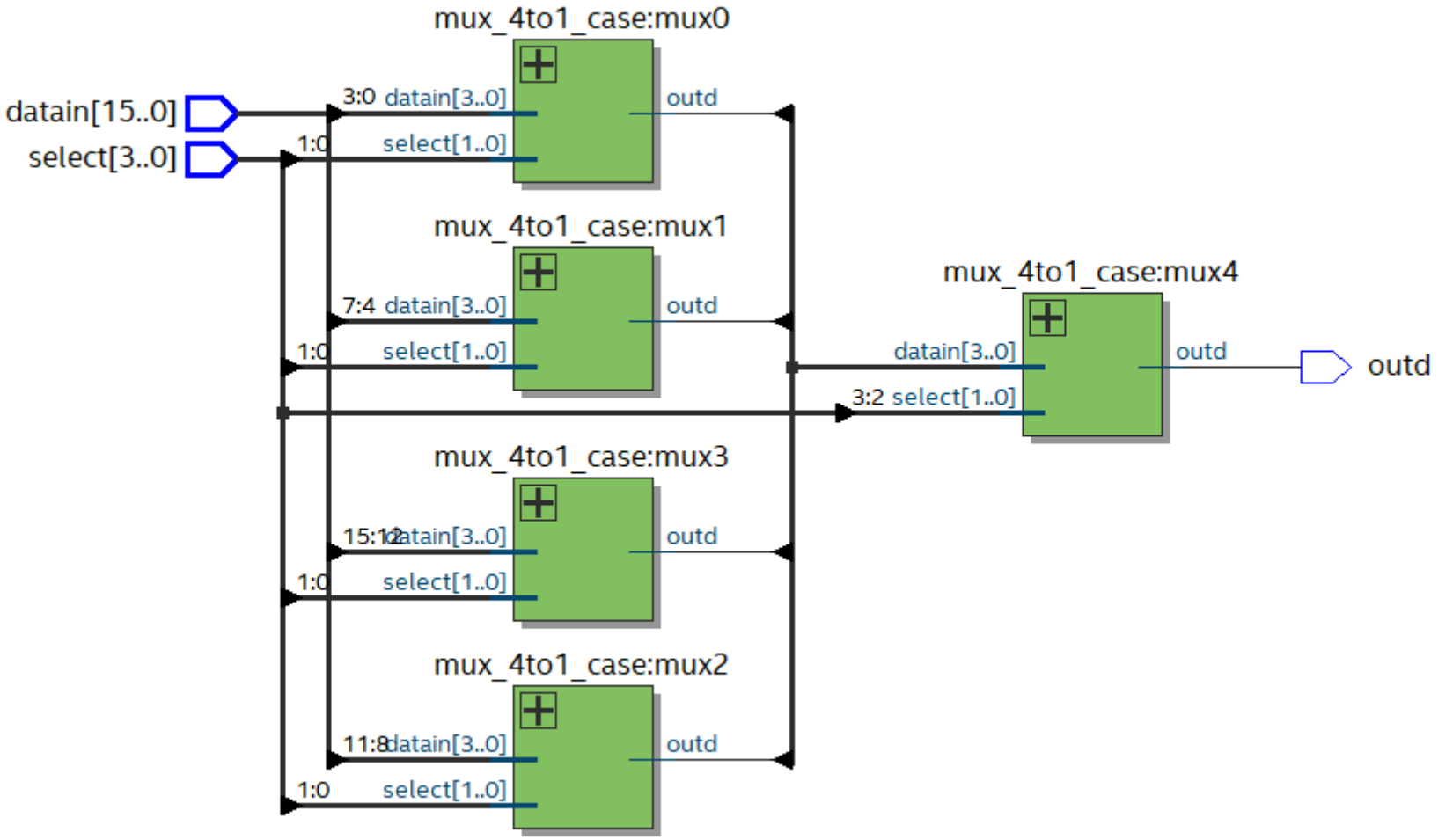
1. **פתח** את הקובץ mux_16to1.sv וקבע אותו כ- TOP.
2. **יש לממש** תכן הירארכי ב- Verilog על ידי שימוש ברכיב עם **case** ובצוע instantiation (הפעלת המודול).
3. **יש להשתמש** בהעברה פרמטרים באופן מפורש, כמו בדוגמה להלן של בורר 2 ל-1:

```
mux2 mux2_1 (.i0(i0),.i1(i1),.sel(sel[0]),.y(mux2_y0));
```

4. **השלם** את הקוד שלך **ובצע אלאבורציה.**
5. **הוסף את הקוד של ה- MUX ההיראכי לדו"ח.**

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

      logic [3:0] muxi; // Holds the intermediate results

      mux_4to1_case mux0 ( datain[3:0],select[1:0], muxi[0]); // plesae use the full
                                                  // way of passing parameters not the short one as in this example

      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),.select(select[3:2]), .outd(outd));

endmodule
```

6. הוסף את המימוש כ- RTL VIEW של ה- MUX ההיראכי **לדו"ח.**


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

4 מונה סינכרוני עולה

מטרה: לבנות מונה סינכרוני (מופעל שעון) עולה, שסופר מ- 0 עד 15 באופן מחזורי.

נתון קובץ של מונה בינארי סינכרוני עולה **simple_up_counter.sv.**

נתונים:

simple_up_counter.sv - Module interface

Direction	Type	Width	Name
Input	logic	1	clk
Input	logic	1	resetN
Output	logic	[3:0]	count

simple_up_counter.sv - Truth table

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

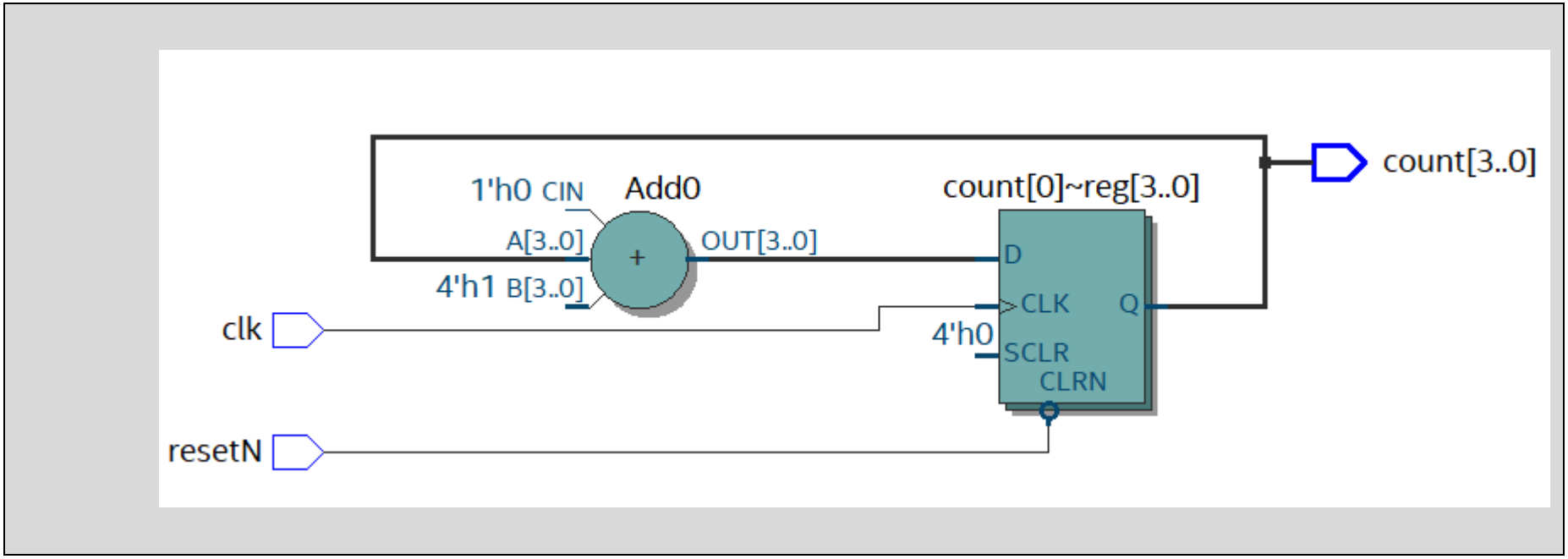
הנחיה: **אין** לממש את המונה כמכונת מצבים.

1. פתח את הקובץ simple_up_counter.sv, קבע אותו כ- TOP.
2. **השלם** את הקוד שלך.
3. **בצע סינתזה** (Analysis & Synthesis), היות ובתרגיל זה יש להריץ סימולציה בשלב הבא.

4. הוסף את הקוד של מונה סינכרוני עולה לדו"ח.

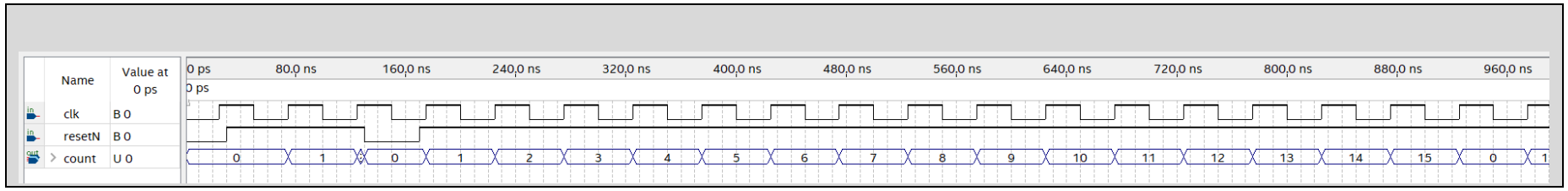
```
module simple_up_counter
      (
   // Input, Output Ports
   input logic clk,
   input logic resetN,
   output logic [3:0] count
   );

   always_ff @( posedge clk or negedge resetN )
   begin

            if ( !resetN ) begin // Asynchronic reset
                  count <= 4'b0;
            end
            else begin
                  count <= count + 4'b1;
            end
      end // always
endmodule
```

5. הוסף את המימוש כ- RTL VIEW של מונה סינכרוני עולה לדו"ח.



6. **צור** קובץ WAVEFORM והרץ סימולציה של המעגל. יש לוודא **שהסימולציה מכסה את כל המצבים המעניינים** של הכניסות.

7. **הוסף את תוצאות הסימולציה של מונה סינכרוני עולה לדו"ח.**



5 מונה סינכרוני עם קפיצות

מטרה: לבנות מונה סינכרוני עם קפיצות.

נתון הקובץ **jmp_counter.sv** - שלד של מונה בינארי סינכרוני עולה עם קפיצות: המונה מתחיל לספור מ- 0, **סופר עד 5, קופץ ל- 11**, ממשיך לספור עד 15, חוזר ל- 0, ושוב סופר מ- 0 עד **jmp_from**, קופץ ל- **jmp_to**, סופר עד 15, מתאפס וכן הלאה ממשיך בצורה מחזורית.

נתונים:

jmp_counter.sv - Module interface

Direction	Type	Width	Name
input	logic	1	clk
input	logic	1	resetN
output	logic	[3:0]	count

jmp_counter.sv - Truth table

CLK	resetN	count[3:0]	count next
x	0	4'b0000	4'b0000
↑	1	jmp_from	jmp_to
↑	1	4'b1111	4'b0000
↑	1	Else: count	count+1

הנחיה: **אין** לממש את המונה כמכונת מצבים.

1. **פתח** את הקובץ **jmp_counter.sv**, קבע אותו כ- TOP
2. **השלם** את הקוד כנדרש.

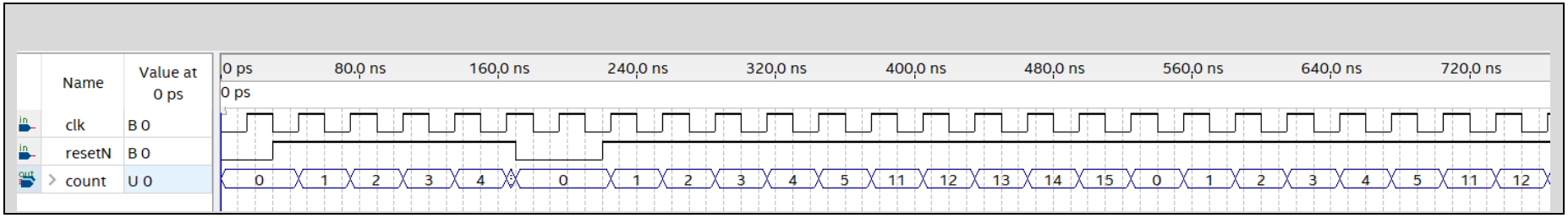
   שים לב לעדכן בקוד שלך את הפרמטרים **jmp_from** ו- **jmp_to** לערכים הנתונים.

3. **הרץ סינתזה.**

4. העתק את הקוד של המונה עם קפיצות שכתבת לכאן.

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

// Internal or local parameters/variables declarations

      localparam jmp_from = 4'b0101; // <------ assign here the right value
      localparam jmp_to = 4'b1011;   // <------ assign here the right value


   always_ff @( posedge clk or negedge resetN )
   begin

            if ( !resetN ) begin // Asynchronic reset
                  count <= 4'b0;
            end
            else if(count == jmp_from) begin
                  count <= jmp_to;
            end
            else begin
                  count <= count + 4'b1;
            end
      end // always
endmodule
```

5. **צור קובץ WAVEFORM והרץ** סימולציה של המעגל.
6. **הוסף את תוצאות הסימולציה של המונה עם קפיצות לדו"ח.**



6 תצוגת 7Segment עם הדלקה וכיבוי מלאים

מטרה: **לתכנן רכיב צירופי** בשם **hexss.sv** שממיר מספר מקוד בינארי ללקוד המתאים לתצוגת Seven Segment.

נתונים:

hexss.sv - Module interface

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

תצוגות Seven Segment בעלת המבנה המרחבי הבא:

hexss.sv - 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

<u>הסבר והנחיות</u>: **נתון** לך שלד של רכיב בשם **hexss.sv** בו יש להשלים את הקוד: להמרת מילה בינארית ברוחב 4 סיביות לתצוגת Seven Segment עבור כל 16 הצירופים האפשריים של 4 הסיביות. לממיר יש גם שתי כניסות בקרה של סיבית אחת כל אחת, **LampTest** ו- **darkN**, אשר תפקידן נתון בטבלת האמת לעיל.

<u>הנחיה</u>: יש לממש את טבלת ההמרה בעזרת מערך דו ממדי, אותו אפשר להגדיר כך:

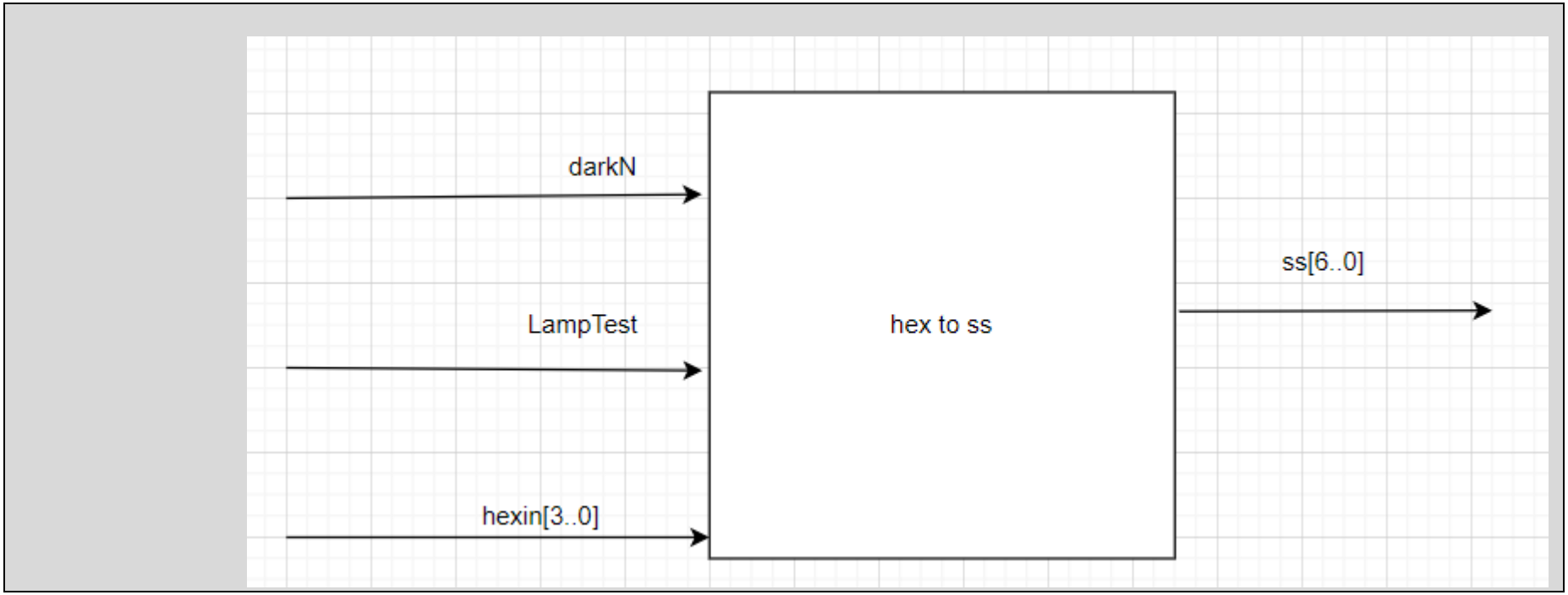
```
logic [0:15] [6:0] SevenSeg =
                    {     7'h40, //0
                          7'h79, //1
                          ..........
                    };
```

<u>ולא</u> להשתמש בפקודת CASE (שהוא גם פתרון אפשרי).

<u>הערה</u>: רכיב זה יהיה שימושי בניסויים הבאים ובפרויקט הסופי.

6.1 ארכיטקטורה של המודול 7Segment

1. שרטט **בעפרון** או בכלי ציור (לא בקווארטוס, למשל ב- www.draw.io) סמל גרפי של הרכיב (כניסות ויציאות).
2. **הוסף את השרטוט לדוח.**



6.2 קוד של המודול 7Segment

1. **השלם** את הקוד המתאים **והרץ** סינתזה מוצלחת.
2. **העתק את הקוד השלם לכאן.**

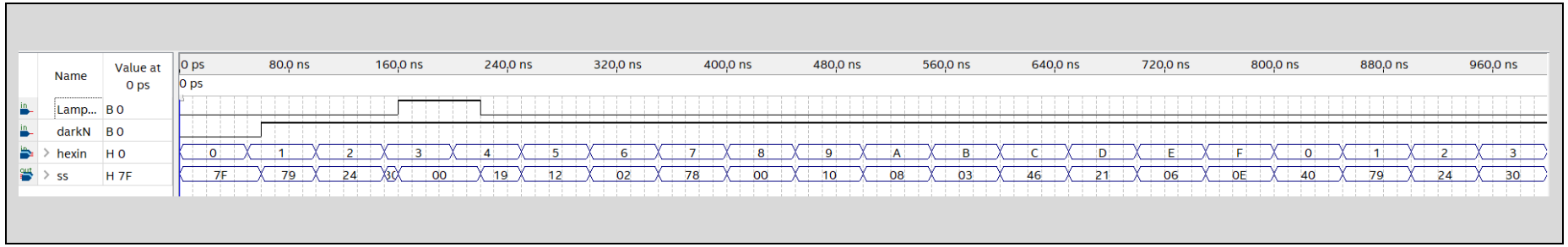
```
module hexss
      (
      input logic [3:0] hexin, // Data input: hex numbers 0 to f
      input logic darkN,
      input logic LampTest,   // Aditional inputs
      output logic [6:0] ss   // Output for 7Seg display
      );

// Declaration of two-dimensional array that holds the 7seg codes
logic [0:15] [6:0] SevenSeq =
                        {     7'h40,7'h79,7'h24,7'h30,
                              7'h19,7'h12,7'h02,7'h78,
                              7'h00,7'h10,7'h08,7'h03,
                              7'h46,7'h21,7'h06,7'h0E};
always_comb
begin
      if (!darkN) begin
            ss = 7'h7F;
            end
      else if(LampTest == 1'b1) begin
            ss = 7'h00;
            end
      else begin
            ss = SevenSeq[hexin];
      end
end

endmodule
```

6.3 סימולציה של המודול 7Segment

1. **צור קובץ WAVEFORM והרץ** סימולציה של המעגל.

הערה: בסימולציה ניתן לקבוע בקלות ערכים עוקבים לכניסה hexin (0 עד F) בעזרת **כלי המונה, ה- Count Value** שנמצא על סרגל הכלים בחלון הסימולציה, ה- Simulation Waveform Editor.

2. **הוסף את תוצאות הסימולציה לדו"ח.**



7 הגשה וגיבוי העבודה

1. **שמור** את הדוח רגיל וכ- PDF, **והעלה את ה- PDF למודל**.
2. **שמור** את הפרויקט רגיל וגם **כארכיב (באמצעות Project -> Archive Project)**. פעולת הארכיב יוצרת קובץ עם סיומת ***.qar** . שים לב לשנות את השם שמציע הקוורטוס לשם קצר באנגלית ושמכיל את התאריך (ושעה) של הגיבוי, למשל SV1_PreLab_22_2_22.
3. **גבה** קובץ זה, **העלה למודל והבא למעבדה** כי תצטרך אותו בניסוי.

<u>הערה</u>: 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