



EEELabs

Electrical Engineering Elementary Labs
מעבדות יסוד בהנדסת חשמל

הפקולטה להנדסת חשמל
ע"ש אנדרו וארנה ויטרבי



הטכניון
מכון טכנולוגי לישראל

מעבדה בהנדסת חשמל
044157 א'1

ניסוי SV2 שאלות ודוח הכנה

גרסה 2.1
קיץ תשפ"ב 2022

שם המדריך	תאריך ההגשה של דוח ההכנה

סטודנט	שם פרטי	שם משפחה
1	עמיחי	שטרנליכט
2	יקיר	לוגסי

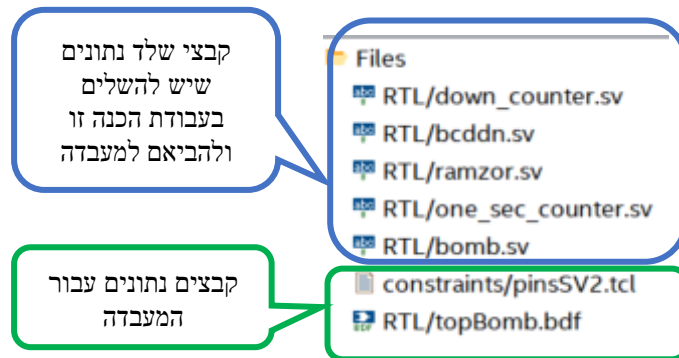
תוכן עניינים

3.....	1 פתיחת ארכיב	1
3.....	2 מונה BCD (דצימלי) יורד בעל שתי ספרות	2
3	2.1 הגדרות – מונה BCD יורד	2.1
3	2.2 מונה יורד לספרה אחת	2.2
4.....	2.2.1 ממשק – מונה יורד של ספרה אחת	2.2.1
4.....	2.2.2 טבלת אמת – מונה יורד של ספרה אחת	2.2.2
5.....	2.2.3 קוד וסימולציה – מונה יורד של ספרה אחת	2.2.3
7	2.3 מונה BCD יורד לשתי ספרות	2.3
7.....	2.3.1 ממשק – מונה יורד של שתי ספרות	2.3.1
7.....	2.3.2 טבלת אמת – מונה יורד של שתי ספרות	2.3.2
8.....	2.3.3 מימוש המונה לשתי ספרות	2.3.3
10.....	2.3.4 סימולציה	2.3.4
11.....	3 רמזור מבוקר – מימוש עם מכונת מצבים	3
11	3.1 הגדרת הדרישות	3.1
11.....	3.1.1 ממשק הרמזור	3.1.1
12.....	3.1.2 פעולת הרמזור	3.1.2
12	3.2 מכונת מצבים FSM – לבניית הרמזור	3.2
12.....	3.2.1 שרטוט דיאגרמת מצבים	3.2.1
13.....	3.2.2 הוספת הקוד של מכונת המצבים	3.2.2
18	3.3 סימולציה	3.3
19.....	4 פרויקטון – פצצה (Bomb)	4
19	4.1 הגדרת הדרישות לפצצה	4.1
19.....	4.1.1 ממשק הפצצה	4.1.1
20.....	4.1.2 פעולות הפצצה	4.1.2
20.....	4.1.3 ארכיטקטורה של הפצצה	4.1.3
21	4.2 תכנון פרויקטון הפצצה	4.2
22.....	4.2.1 תכנון של מודול הפצצה	4.2.1
22.....	4.2.2 תכנון מכונת המצבים של מודול הפצצה	4.2.2
23	4.3 מימוש מכונת המצבים של הפצצה	4.3
25	4.4 סימולציה של מודול הפצצה	4.4
26.....	5 גיבוי העבודה	5

המטרה במעבדה זו היא להעמיק את הידע בשפת SV ולתרגל שמוש במכונת מצבים.

1 פתיחת ארכיב

- **צרו תיקיה למעבדה זו בכונן Z שלכם.** הורידו מהמודל את קובץ הארכיב (.qar) של המעבדה ופתחו אותו לפרויקט בתיקייה שיצרתם.
- **שימו לב –** יש לשנות את ה PATH שמציע הקווארטוס ל PATH קצר, שאינו מכיל: אותיות בעברית, רווחים, ואת הסימן '-'.
 - ודאו שתכולת הקבצים (Project Navigator->Files) שלכם דומה לזו:



בעבודת הכנה זו תתחילו את הפרויקט ותמשיכו לעבוד על אותו פרויקט ואותם קבצים גם במעבדה.

2 מונה BCD (דצימלי) יורד בעל שתי ספרות

2.1 הגדרות – מונה BCD יורד

הגדרת המטלה: עליכם לבנות מונה ציקלי יורד בעל 2 ספרות, כאשר כל ספרה הינה ברוחב 4 ביט (BCD= Binary Coded Decimal). המונה סופר בקידוד בינארי כלפי מטה עם אפשרות לטעינת מספר סינכרונית. אפשר להניח שערכי הכניסה מהטעינה הסינכרונית הם תמיד בתחום של 0 עד 9, לכל אחת מהספרות (כלומר אין צורך לבדוק תקינות קלט).

הנחיה:

- תחילה, עליכם לכתוב מונה BCD לספרה אחת ולבדוק אותו בעזרת סימולציה.
- לאחר מכן, עליכם לבנות מונה לשתי ספרות bcddn בתכן היררכי באמצעות שני רכיבים של מונה לספרה אחת.
- הלוגיקה של המונו כולו צריכה להיות סינכרונית, מלבד סיגנל ה- TC (Terminal Count) שימומש בלוגיקה אסינכרונית.
- ניתן להיעזר בקוד של המונה העולה ממעבדה קודמת (SV1).
- יש לבצע את המשימה לפי השלבים המפורטים בדו"ח זה.

2.2 מונה יורד לספרה אחת

המונה בקובץ down_counter.sv הינו מונה דצימלי לספרה אחת הסופר מטה מ- 9 עד 0. כאשר הספירה מגיעה ל- 0 הוא מעלה את סיגנל tc אסינכרוני למשך כל הזמן שהמונה ב- 0. סיגנל זה מאפשר לשרשר מספר מונים וכך לבנות מונה בעל כמה ספרות המורכב ממספר יחידות של מונה לספרה אחת.

שלד הקוד של המונה היורד נתון לכם בפרויקט. להלן הדרישות עבור המונה:

2.2.1 ממשק – מונה יורד של ספרה אחת

הכניסות והיציאות של יחידת המונה הדצימלי היורד נתונות להלן:

Direction	Width	Name
Input	1	clk
Input	1	resetN
Input	1	loadN
Input	1	enable1
Input	1	enable2
Input	1	enable3
Input	[3:0]	datain
Output	[3:0]	count
Output	1	tc

2.2.2 טבלת אמת – מונה יורד של ספרה אחת

פעולת המונה מתוארת בטבלת האמת הבאה:

clk	resetN (Async)	loadN	enable1	enable2	enable3	datain[3:0]	count[3..0]	Description
x	0	X	x	x	x	X	4'b0000	Reset
↑	1	0	x	x	x	datain[3:0]	datain[3:0]	Load
↑	1	1	0	x	x	X	previous count	
↑	1	1	x	0	x	X	previous count	
↑	1	1	x	x	0	X	previous count	
↑	1	1	1	1	1	X	if (count == 0) count <= 4'h9; else count <= count-1;	Decrement

היציאה האסינכרונית tc מתוארת להלן:

clk	resetN (Async)	loadN	enab le1	enab le2	enab le3	datain[3:0]	count[3:0]	tc (Async)	Description
X	x	x	X	x	x	x	count[3:0]≠0	0	
X	x	x	X	x	x	x	4'b0000	1	Terminal count

הערה חשובה!!! בכמה מקבצי השלד הנתונים היה צורך להסתיר חלק מהקוד כדי שהקוד יעבור קומפילציה. לכן לפני שמתחילים לכתוב קוד יש להסיר את שורות ההערות הבאות:

`/* $$$$ remove to fill`

ולהשלים את הקוד שלכם במקומות הנדרשים המסומנים, לרוב היכן שרשום:

`//fill your code here`

2.2.3 קוד וסימולציה – מונה יורד של ספרה אחת

- השלימו את הקוד בשלד הנתון של מונה לספרה אחת `down_counter.sv`.

הנחיה:

לפי כללי קידוד נאותים (Good Practice) הקפד ש:

- הקוד יהיה קריא (שמות משמעותיים למשתנים, שימוש בקבועים וכו')
- הקפדה על הזחות נכונות
- הקוד לא מכיל קוד מיותר (למשל `count <= count`)
- משתנים שלא השתנו יחסית לערכי ה-DEFAULT שלהם לא מקבלים השמה
- הקוד המוצג בדוח צריך להיות צבעוני - השתמש ב NOTEPAD++ להעתקה לדוח

- הוסיפו את הקוד שלכם לדו"ח.

```
- module down_counter
- (
-   input logic clk,
-   input logic resetN,
-   input logic loadN,
-   input logic enable1,
-   input logic enable2,
-   input logic enable3,
-   input logic [3:0] datain,
-
-   output logic [3:0] count,
-   output logic tc
- );
-
- // Down counter
- always_ff @(posedge clk or negedge resetN)
-   begin
-
-       if ( !resetN ) begin// Asynchronous reset
-
-           count <= 4'b0;
-
-       end
-
-       else if ( !loadN )      begin // Synchronic logic
-
-           count <= datain;
-
-       end
-
-       else if (!enable1 || !enable2 || !enable3) begin
-
-           end //Synch
-
-       else begin
-
-           if(count == 4'b0) begin
-
-               count <= 4'h9;
-
-           end
-
-       end
-   end
```

```

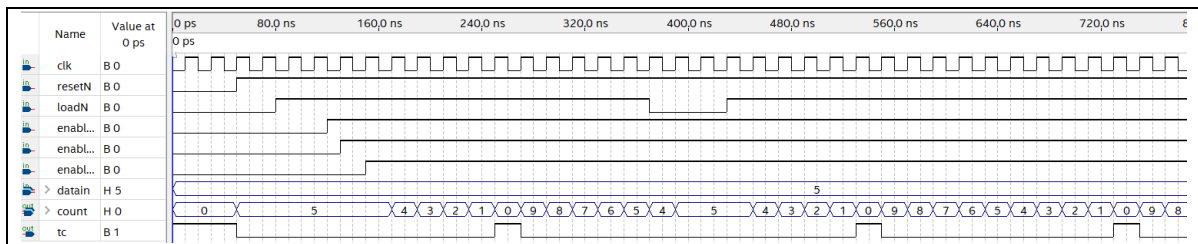
-         else begin
-
-             count <= count-4'b1;
-
-         end
-
-     end
-
- end //always
-
- // Asynchronous tc
-
- assign tc = (count == 4'b0) ? 1'b1 : 1'b0; // fill your code here
-
-
- endmodule

```

- **בצעו סינתזה** מוצלחת למונה ספרה אחת וצרפו את הסיכום לדו"ח.

Flow Summary	
Flow Status	Successful - Tue Aug 09 16:56:11 2022
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	SV2
Top-level Entity Name	down_counter
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	4
Total pins	15
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

- **בצעו סימולציה** מוצלחת למונה ספרה אחת וצרפו את התוצאות לדו"ח.
- **הקפידו להציג** את המשתנים כל אחד בפורמט הרלוונטי לו (BINARY/HEX/DECIMAL).
- **הקפידו לנצל** נכון את זמן הסימולציה - אין צורך בהרבה מחזורים.



2.3 מונה BCD יורד לשתי ספרות

בקובץ **bcdn.sv** נתון שלד של מונה שתי-ספרות הסופר מטה: מתחיל מ-99 ויורד עד 0. כאשר המונה מסיים ומגיע ל-0, הוא מעלה את סיגנל ה-tc ל-1.

- פתחו את הקובץ **bcdn.sv** והגדירו אותו כ- TOP LEVEL של הפרויקט.
- ממשו את המונה **bcdn**, בצורה הירארכית באמצעות קוד SV, משני רכיבים של "מונה יורד לספרה אחת" **down_counter**, לפי הדרישות הבאות:

2.3.1 ממשק – מונה יורד של שתי ספרות

ממשק הכניסות והיציאות של יחידת המונה הדצימלי היורד של 2 ספרות:

Direction	Width	Name
Input	1	Clk
Input	1	resetN
Input	1	loadN
Input	1	enable1
Input	1	enable2
Output	[3:0]	countL
Output	[3:0]	countH
Output	1	Tc

2.3.2 טבלת אמת – מונה יורד של שתי ספרות

פעולת המונה היורד של 2 ספרות מתוארת בטבלת האמת הבאה:

מטלה: הוסף את תיאור הפעולות בעמודה האחרונה (Description)



clk	resetN	loadN	enable1	enable2	countL[3:0]	countH[3:0]	Description
x	0	x	x	x	4'b0000	4'b0000	Reset
↑	1	0	x	x	datainL	datainH	Load
↑	1	1	0	x	previous count	previous count	
↑	1	1	x	0	previous count	previous count	
↑	1	1	1	1	countL[3:0]-1	if (countL[3:0]==0) countH[3:0]-1	Decrement

שימו לב: בפעולה רגילה של המונה (שורה אחרונה בטבלה) – כאשר אחת הספרות מגיעה ל-0, במחזור לאחר מכן היא חוזרת באופן ציקלי לערך 9. להזכירכם - סיגנל המצביע על הגעה של מונה של ספרה אחת הוא לערך 0 הוא tc של המונה.

היציאה האסינכרונית tc מתוארת להלן:

Clk	resetN	loadN	enable1	enable2	countL[3:0]	countH[3:0]	tc (Async)	Description
X	X	x	x	X	countL[3:0]≠0	countH[3:0]≠0	0	
X	X	x	x	X	4'b0000	countH[3:0]≠0	0	
X	X	x	x	X	countL[3:0]≠0	4'b0000	0	
X	X	x	x	X	4'b0000	4'b0000	1	Terminal count

2.3.3 מימוש המונה לשתי ספרות

- השלימו את הקוד שלכם בשלד הנתון של `bcdn.sv` והוסיפו אותו לדו"ח.

הנחיה:

יש לממש את הקוד באופן היררכי כך שיהיה מבוסס על המונה של ספרה אחת מהסעיף הקודם. שימו לב: `datainL` ו- `datainH` מוגדרים כפרמטרים בגודל 4 סיביות כל אחת, כלומר הם יקבלו את ערכיהם ממספרים שיוכנסו חיצונית, דרך טבלת פרמטרים שבשרשור הגרפי. הערך המוגדר בקוד הינו ערך ברירת מחדל. חשבו כיצד לשרשר את שני המונים.

```

module bcdn
(
    input logic clk,
    input logic resetN,
    input logic loadN,
    input logic enable1,
    input logic enable2,

    output logic [3:0] countL,
    output logic [3:0] countH,
    output logic tc
);

// Parameters defined as external, here with a default value - to be
// updated
// in the upper hierarchy file with the actual bomb down counting values
// -----
    parameter logic [3:0] datainL = 4'h2 ;
    parameter logic [3:0] datainH = 4'h1 ;
// -----

    logic tclow, tchigh; // internal variables terminal count

// Low counter instantiation
    down_counter lowc(.clk(clk),
                      .resetN(resetN),
                      .loadN(loadN),
                      .enable1(enable1),
                      .enable2(enable2),

```



```

        .enable3(1'b1),
        .datain(datainL),
        .count(countL),
        .tc(tc_low) );

// High counter instantiation
down_counter highc(.clk(clk),


        .resetN(resetN),
        .loadN(loadN),
        .enable1(enable1),
        .enable2(enable2),
        .enable3(tc_low),
        .datain(datainH),
        .count(countH),
        .tc(tc_high) );

assign tc = (tc_low == 1'b1 && tc_high == 1'b1);

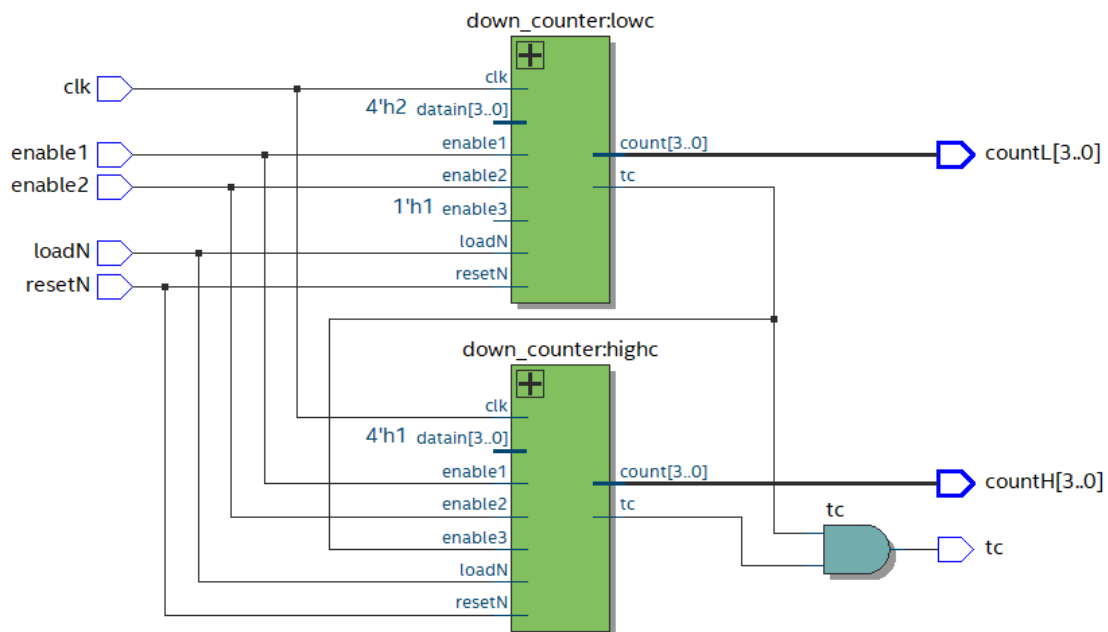
endmodule

```

- **הריצו סינתזה** מוצלחת לקובץ bcddn.sv וצרפו את הסיכום לדו"ח,

Flow Summary	
	
Flow Status	Successful - Tue Aug 09 17:45:36 2022
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	SV2
Top-level Entity Name	bcddn
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	8
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

- **הציגו את המימוש** של המונה bcddn כ- RTL VIEW והוסיפו אותו לדו"ח.



2.3.4 סימולציה

- הגדירו מה תרצו לבדוק בסימולציה – איזה מצבים מעניינים (המשיכו למלא את הטבלה והוסיפו שורות אם צריך).
- בדקו את כל הכניסות והיציאות ואת כל מקרי הקצה המעניינים.

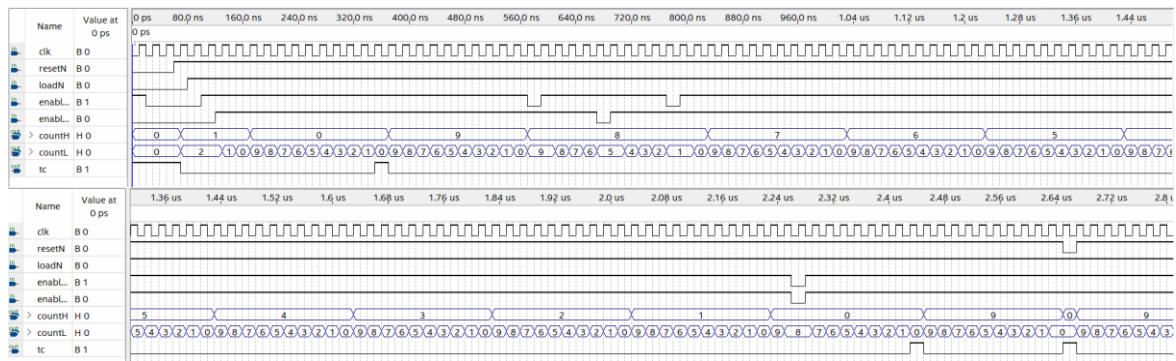
שימו לב! עליכם להשתמש בפולסים צרים (ברוחב של שעון אחד) ככניסות ה-ENABLE.

תוצאות צפויות	מצב
כל היציאות מאותחלות	יציאה מ-RESET
עושה decrement	ENABLE עם COUNT
לא משנה את הערך	ENABLE בלי COUNT
מוציא 1 רק כאשר שני המונים ב-0	TC
כיבוי tc של מונה שני הספרות כאשר עוברים ל-99	מעבר מ-0 ל-9 של שני המונים

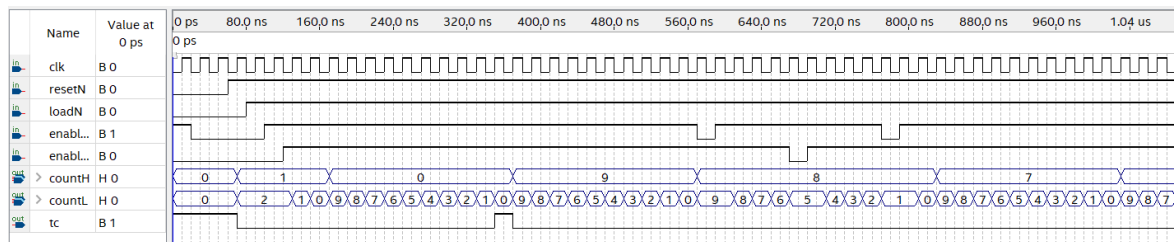
הקפידו להציג כל משתנה בפורמט הרלוונטי לו (DECIMAL/HEX/BINARY).
הריצו סימולציה מלאה למודול **bcdn**.

- הוסיפו את תוצאות הסימולציה לדו"ח. אם לא כל הבדיקות נכנסות למסך אחד, הציגו מספר מסכים לפי הצורך.

הרצה מלאה



"זום אין" לבדיקת loadN וקווי enable



3 רמזור מבוקר – מימוש עם מכונת מצבים

הגדרת המטלה: בתרגיל זה יהיה עליכם לממש רמזור תוך שימוש במכונת מצבים.

בעבודת הכנה זן אתם תממשו את מכונת המצבים על ידי הוספת הקוד שלכם בשלד קוד הנתון ותריצו סימולציה לבדיקת תקינות הקוד שלכם.

במעבדה אתם תבדקו את הרמזור על ידי צריבת התכן לכרטיס FPGA - עם נוריות שאין לכם בערכה הביתית.

- **פתחו** את קובץ השלד של הרמזור **ramzor.sv**, הגדירו אותו כ-TOP LEVEL והוסיפו אליו את הקוד שלכם לפי ההנחיות, הדרישות והפירוט שבסעיפים הבאים.

3.1 הגדרת הדרישות

לרמזור שלוש נוריות (המהוות יציאות) בצבעים אדום, צהוב וירוק, הדולקות לפי סדר זמן המוגדרים מראש. הרמזור ממומש באמצעות מכונת מצבים ופועל באופן מחזורי וקבוע, לפי הקצב של שעון המערכת. ראו הגדרות מפורטות להלן.

3.1.1 ממשק הרמזור

הכניסות והיציאות של הרמזור מתוארות בטבלה הבאה:

Direction	width	Name
Input	1	Clk
Input	1	resetN
Input	1	Switch
Output	1	redLight
Output	1	yellowLight
Output	1	Greenlight

3.1.2 פעולת הרמזור

הרמזור פועל באופן הבא:

- מצבי הרמזור הם: אדום, אדום_צהוב, ירוק, צהוב.
- הרמזור שווה בכל מצב במשך זמן נתון ואז עובר אוטומטית למצב הבא, לפי סדר קבוע ובצורה ציקלית: אדום, אדום-צהוב, ירוק, צהוב, אדום, אדום-צהוב, וכו'...
- הזמנים בהם שווה הרמזור בכל מצב מוגדרים כפרמטרים מספריים הנמדדים בעשירית השניה, ונתונים בטבלה הבא:

Parameter	Type	Default value
RED_TIME	int	44
RED_YELLOW_TIME	int	16
GREEN_TIME	int	34
YELLOW_TIME	int	26

- פעולת resetN מאפסת את הרמזור בצורה סינכרונית - כלומר מחזירה אותו למצב אדום.
 - פעולת SwitchN: לחיצה על לחצן הכניסה המחובר ל-SwitchN תגרום ל:
 - אם הרמזור היה במצב אדום תעביר את הרמזור ישירות (בצורה סינכרונית בעליית השעון הבאה) למצב אדום-צהוב, ללא המתנה.
 - אחרת - לא תעשה כלום והרמזור ימשיך לתפקד ללא שינוי.
- הערה:** שני הלחצנים resetN ו-SwitchN עובדים בלוגיקה הופכית: הם פעילים ב-'0' לוגי (על הכרטיס – זה לחצן לחוץ), ואינם פעילים ב-'1' לוגי (לחצן לא לחוץ).

3.2 מכונת מצבים FSM – לבניית הרמזור

מכונת המצבים תכיל מונה יורד בקצב של עשירית שניה שישמור את הזמן שבו אנחנו עוד צריכים להישאר במצב הנוכחי. בהגעה לזמן 0 במונה, מכונת המצבים תעבור למצב הבא, ותאתחל את המונה לזמן הדרוש של המצב החדש אליו עוברים.

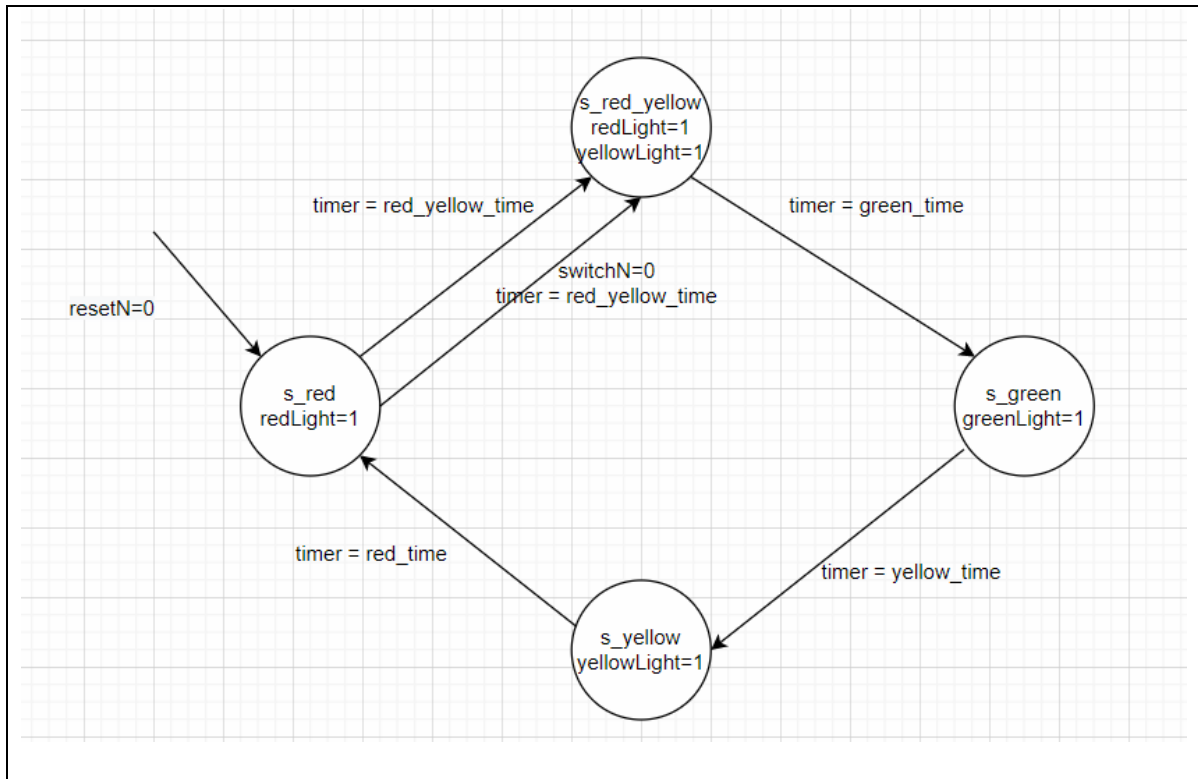
לפני שניגשים לכתובת הקוד – עליכם לתכנן את מכונת המצבים ולשרטט דיאגרמת מצבים.

3.2.1 שרטוט דיאגרמת מצבים

- תאור המצבים** – לנוחיותכם, נתון תיאור המצב הראשון, דהיינו אדום.
- השלימו את תאור המצבים האחרים בטבלה להלן:
 - את הפעולה העיקרית שעושים בכל מצב
 - את התנאים למעברים בין המצבים
 - את עידכון ערכי היציאות – יש לרשום רק יציאות שאינן Default ומתעדכנות/משתנות באותו המצב

שם המצב	פעילות עיקרית	לאיזה מצב עוברים מהמצב הנוכחי ובאילו תנאים	עידכון יציאות
s_red	מעדכנים את המונה לערך אדום צהוב	אחרי שהמונה האדום ירד לאפס - עוברים למצב אדום צהוב	מדליקים את הנורית האדומה
s_red_yellow	מעדכנים את ערך המונה לערך ירוק	אחרי שהמונה האדום_צהוב ירד לאפס - עוברים למצב ירוק	מדליקים גם את הנורית הצהובה
s_green	מעדכנים את ערך המונה לערך צהוב	אחרי שהמונה הירוק ירד לאפס - עוברים למצב צהוב	מדליקים את הנורית הירוקה
s_yellow	מעדכנים את ערך המונה לערך אדום	אחרי שהמונה הצהוב ירד לאפס - עוברים למצב אדום	מדליקים את הנורית הצהובה

- **שרטטו דיאגרמת מצבים בצורת bubble diagram של הרמזור** (בכל דרך שנוח לכם ובלבד שיהיה ברור: דף ועט, שימוש בתוכנת www.draw.io או POWER_POINT). **הוסיפו** את דיאגרמת המצבים לדוח. **להזכירכם** מציינים את resetN כחץ בודד למצב ההתחלתי ולא בכל המצבים.



3.2.2 הוספת הקוד של מכונת המצבים

בפרויקט, נתון לכם שלד של הרמזור אליו תוסיפו את הקוד שלכם במקומות המסומנים.

הסבר לשלד הקוד הנתון:

- הקוד של מכונת המצבים בנוי כשלושה משפטי **always** וממומש כמכונת MOORE:
- **always_ff** סינכרוני – שמעדכן את המצב הנוכחי ומונה הזמן הנוכחי בכל עלייה של שעון המערכת, ובמקרה של resetN מאפס את המערכת למצב נוכחי אדום ומונה הזמן שצריך לשהות בו במצב אדום (RED_TIME).
- **always_comb** שמעדכן את המצב הבא בכל עלייה של סיגנל t_sec (זהו סיגנל חיווי שעולה לערך '1' לוגי כל עשירית שניה).
- **always_comb** שמעדכן את היציאות בכל אחד מהמצבים (מכונת MOORE - ללא תלות בכניסות).

הערה: אפשר היה לבחור לבצע את עדכון המצבים והיציאות ללא הפרדה ביניהם, באותו משפט **always_comb**, כפי שנועשה בתרגיל הבא בדוח - הפצצה. שתי הדרכים חוקיות, והבחירה תלויה בהעדפות המתכנת.

- **סיגנל החיווי של עשיריות שניה** מופעל על ידי שימוש במונה השניות שהכרתם במעבדות הקודמות, **one_sec_counter**. הרמזור מכיל מופע של רכיב זה במצב turbo קבוע, כך שהיציאה שלו **one_sec**, המחוברת לסיגנל **t_sec** ביחידת הרמזור, תפיק באופן קבוע פולס כל עשירית שניה.

- **ספירת הזמן מתבצעת בעזרת מונה יורד** המתעדכן לערך הנכון לפי המצב הרלוונטי. המונה סופר כלפי מטה (מחסר 1 מערכו) לפי אות החיווי של עשיריות השניה (**t_sec**). כאשר המונה מגיע לאפס, מכונת המצבים תעבור למצב הבא, והמונה יאותחל לערך המתאים למצב החדש.

מספר כללים של כתיבת קוד נאותה:

- יש להשתמש בהגדרת המצבים כ- `enum logic`
 - שמירה על גנריות - יש להשתמש בקבועים עד כמה שניתן ולא במספרים בתוך הקוד
 - יש לעבוד עם המצב הנוכחי והמצב הבא: `ramzor_ps, ramzor_ns`
 - מכונת המצבים תעבוד עם שעון המערכת בקצב של 50MHz, ספירת הזמן של המונה תיעשה לפי סיגנל החיווי של עשירית השניה (**t_sec**)
 - עבור סימולציה - יש להקטין את זמני סיגנל החיווי (בקובץ `one_sec_counter.sv`) וזמני המצבים על ידי שינוי יחיד וברור ("ע"י סגירת כהערה ואחר כך פתיחת ההערה המתאימה לצורך צריכה לכרטיס)
 - קוד קריא והזחות נכונות
 - ללא קוד מיותר (למשל `count <= count`)
 - לא לבצע השמה למשתנים שלא השתנו יחסית לערכי ה- DEFAULT
 - הקוד המוצג בדוח צריך להיות צבועני - השתמש ב ++ NOTEPAD להעתקה לדו"ח
- **השלימו את הקוד שלכם** בשלד הנתון של הרמזור בקובץ `ramzor.sv` (ודאו שהוא מוגדר כהיררכיה העליונה).

שימו לב! עליכם לעדכן בקוד שקיבלתם גם את פעולת הלחיצה על SwitchN שפורטה לעיל.

- **בצעו סינתזה מוצלחת והוסיפו את הקוד שלך לדו"ח.**
- ```

- module ramzor
- (
- input logic clk, // 50 MHz clock
- input logic resetN,
- input logic switchN, // pedestrian switch to change the light
-
- output logic redLight, // output to the red lamp
- output logic yellowLight, // output to the yellow lamp
- output logic greenLight // output to the green lamp
-);
-
- //-----
-
- // state machine and parameters decleration
-
- enum logic [2:0] {s_red, s_red_yellow, s_green, s_yellow}
- ramzor_ps, ramzor_ns; // state machine
- logic [6:0] counter_ps, counter_ns; // present and next count down
- second timer
-
- logic t_sec = 1'b0; // A short pulse, once every tenth of a
- second.
-
-
-

```

```

- localparam logic [6:0] RED_TIME = 44; // <----- for simulation
change for example to 4
- localparam logic [6:0] RED_YELLOW_TIME = 16;
- localparam logic [6:0] GREEN_TIME = 34;
- localparam logic [6:0] YELLOW_TIME = 26;
-
- localparam logic LED_ON = 1'b1;
- localparam logic LED_OFF = 1'b0;
-
-
- //-----
-
- // Instance of 1Hz (10Hz if turbo is on) sub module as a time counter
for the state machine
- // In this example one_sec_counter is used with 10 Hz (turbo is set
constantly to 1) so that
- // t_sec will count tens of seconds as needed for the traffic lights
timer
-
- one_sec_counter one_sec_counter (.clk(clk),
-
- .resetN(resetN),
-
- .turbo(1),
-
- .one_sec(t_sec));
-
-
- //-----
-
- // 1. synchronous code, executed once every clock to update the
current state
-
- always_ff @(posedge clk or negedge resetN) // State machine logic
////
- begin
-
- if (!resetN) begin // Asynchronous reset, initialize the state
machine
- ramzor_ps <= s_red;
- counter_ps <= RED_TIME;
- end // asynch
- else begin // Synchronic logic of the state
machine; once every clock
- ramzor_ps <= ramzor_ns; // present state <= next state
- counter_ps <= counter_ns; // counter
- end
-
- end // always_ff state machine //////////////////////////////////////
-
- //-----
-
- // 2. asynchronious code : logic defining what is the next state
-
- always_comb // Update the next state //////////////////////////////////
-

```

```

-
- begin
- // 0. Set default operation values
- ramzor_ns = ramzor_ps;
- counter_ns = counter_ps;
-
-
-
- if (t_sec) begin // perform once every timer pulse (every
0.1 sec)
-
- if (counter_ps > 1) // timer didn't finish
- counter_ns = counter_ps - 7'b1; //
decrement the counter
-
- else begin // counter is 0 go to the next state
-
- case (ramzor_ps)
-
- //Note: the implementation of the
red state is already given you as an example
- s_red: begin
- ramzor_ns = s_red_yellow;
- //next state
- counter_ns = RED_YELLOW_TIME;
- // reload counter with the next value.
- end // s_red
-
- s_red_yellow: begin
- ramzor_ns = s_green; //next
state
- counter_ns = GREEN_TIME; //
reload counter with the next value.
- end // s_red_yellow
-
- s_green: begin
- ramzor_ns = s_yellow; //next
state
- counter_ns = YELLOW_TIME; //
reload counter with the next value.
- end // s_green
-
- s_yellow: begin
- ramzor_ns = s_red; //next
state
- counter_ns = RED_TIME; //
reload counter with the next value.
- end // s_yellow
-
- endcase
- end // else
- end // t_sec
-
- // logic to Check switchN and bypass the previous decision

```



```

- if (switchN == 1'b0 && ramzor_ps == s_red) begin
- ramzor_ns = s_red_yellow;
- counter_ns = RED_YELLOW_TIME; // reload counter with
the next value.
-
- end // switchN
-
-
- end // always next state //////////////////////////////////
-
- //-----
-
- // 3. Moore machine, separate allways_comb defining the outputs based
only on the current state, and not on the inputs
- // NOTE: there are several options, both are OK:
- // 1. to update the outputs separately in a second allways_comb
as in this example,
- // 2. to combine it with the next_state allways_comb, as in the
bomb state_machine example
-
- always_comb // Update the outputs //////////////////////////////////
- begin
- // 0. Set default values to all block outputs
- redLight = LED_OFF;
- yellowLight = LED_OFF;
- greenLight = LED_OFF;
-
- // Update ONLY the outputs that changed from the default value.
-
- case (ramzor_ps)
-
- //Note: the implementation of the red state is already given
you as an example
- s_red: begin
- redLight = LED_ON;
- end // s_red
-
- s_red_yellow: begin
- redLight = LED_ON;
- yellowLight = LED_ON;
-
- end // s_red_yellow
-
- s_green: begin
- greenLight = LED_ON;
- end // s_green
- s_yellow: begin
- yellowLight = LED_ON;
- end // s_yellow
- endcase
- end // always outputs //////////////////////////////////
-
- endmodule

```

### 3.3 סימולציה

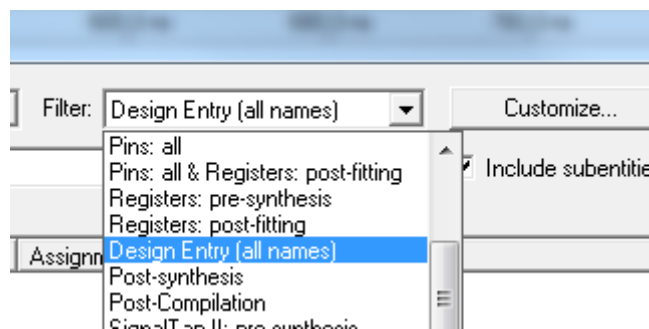
- הגדירו מה תרצו לבדוק בסימולציה – איזה מצבים מעניינים ומלאו את הטבלה בהתאם:

| תוצאות צפויות                 | מצב             |
|-------------------------------|-----------------|
| כל היציאות מאותחלות           | יציאה מ- resetN |
| הנורה RED דולקת               | מצב Red         |
| הנורה RED+YELLOW דולקת        | מצב red_yellow  |
| הנורה GREEN דולקת             | מצב green       |
| הנורה YELLOW דולקת            | מצב yellow      |
| עובר למצב RED+הנורה RED דולקת | SwitchN פעיל    |

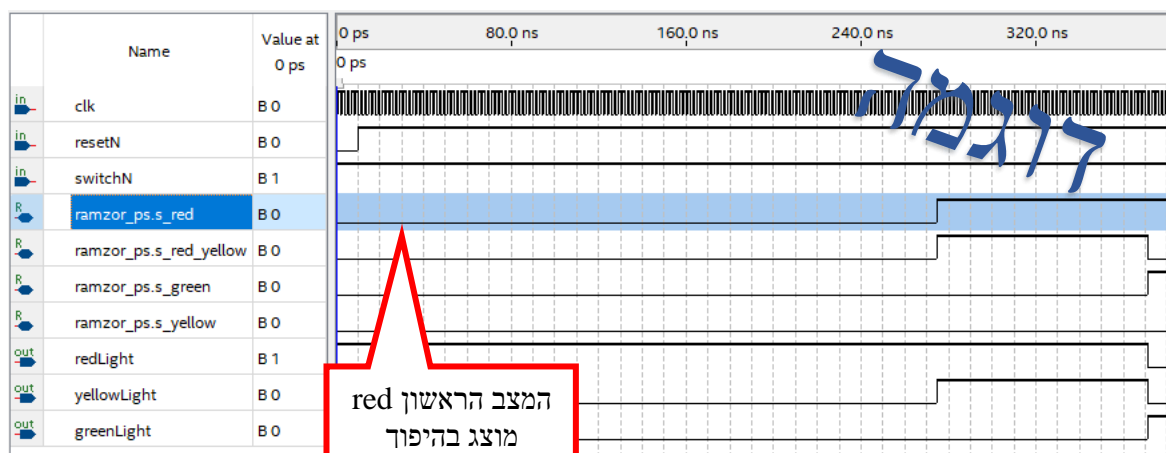
שימו לב! לפני ביצוע הסינתזה לקראת הסימולציה, יש להקטין את הקבוע במחלק התדר בקובץ one\_sec\_counter.sv המגדיר כמה מחזורים הם שנייה אחת. אל תשכחו להחזיר את המצב לקדמותו לפני קומפילציה מלאה וצריבה לכרטיס.

- בצעו סימולציה לפי ההוראות להלן.

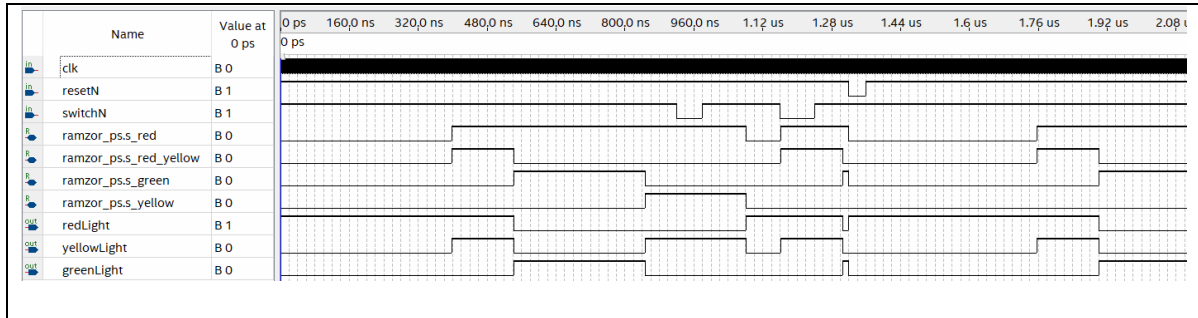
הערה: ניתן לראות בסימולציה את מצבי מכונת המצבים על ידי בחירת מסנן Filter: Design entry (all names), כפי שמופיע באיור הבא:



שימו לב – הציגו בסימולטור את כל המצבים - כל מצב בנפרד ולא כקבוצה. בנוסף, המצב הראשון שתגדירו ("אדום" בדוגמה שלנו) יוצג הפוך (inverse):



## - הריצו סימולציה והוסיפו את תוצאות הסימולציה לדו"ח.



## 4 פרויקטון – פצצה (Bomb)

מטרה: בתרגיל זה עליכם לממש מנגנון לפצצה על פי הפירוט להלן.

בעבודת הכנה זו אתם תתכננו ותשלימו את הקוד של מכונת המצבים של הפצצה ותבדקו את נכונות המימוש בסימולציה.

במעבדה אתם תשלימו את בניית הפצצה ואת בדיקתה על גבי כרטיס ה-FPGA.

### 4.1 הגדרת הדרישות לפצצה

בסעיפים הבאים יוצגו כל הדרישות של פרויקטון הפצצה באופן מפורט.

#### 4.1.1 ממשק הפצצה

הכניסות והיציאות של ההיררכיה העליונה של הפצצה מתוארות בטבלה הבאה:

| Direction | width | Name   | Mapped to         |
|-----------|-------|--------|-------------------|
| Input     | 1     | Clk    | DE10 CLOCK_50     |
| Input     | 1     | resetN | key               |
| Input     | 1     | startN | key               |
| Input     | 1     | waitN  | key               |
| Input     | 1     | Turbo  | switch            |
| Output    | [3:0] | countL | 7Seg and red leds |
| Output    | [3:0] | countH | 7Seg and red leds |
| Output    | 1     | duty50 | red led           |

## 4.1.2 פעולות הפצצה

**הזמן עד לפיצוץ הפצצה נתון: 25 שניות**

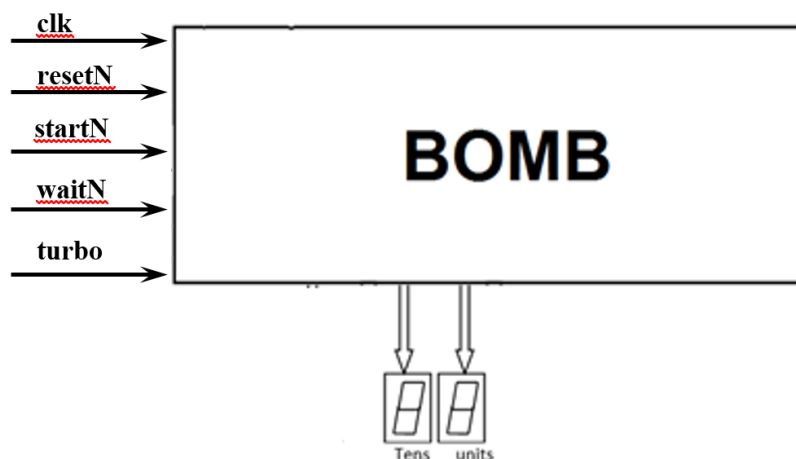
**פעולות הפצצה מוגדרות להלן:**

1. **פעולת RESET** - אסינכרונית פעילה בנמוך, תאפס את תצוגת והמונים ותמתין ל- START
2. **פעולת START** - פעילה בנמוך (תמופה ללחצן):
  - a. **בלחיצה על לחצן startN** - הפצצה רק תטען **זמן לפצצה** אך עדיין הזמן לא יתחיל לרדת.
  - b. **בשחרור הלחצן startN** - הפצצה תמנה אחורה עד ל-0 (פיצוץ!).
3. **התפוצצות הפצצה** – עם סיום הספירה, כאשר מונה הזמן בפצצה יגיע ל-0, יהיה **הבהוב של 00** בתצוגת ה7SEG בתדר של 1/2 Hz (התצוגה תראה 00 למשך שנייה ואז תכבה למשך שנייה, וכך בצורה מחזורית) עד להפעלת RESET מחדש.
4. **פעולת WAIT** - פעילה בנמוך, תמופה ללחצן:
  - a. **בלחיצה על לחצן waitN** - מונה הזמן של הפצצה יעצור כל זמן הלחיצה.
  - b. **בשחרור הלחצן waitN** – מונה הזמן של הפצצה ימשיך מאותו מקום שעצר.
  - c. **הערה:** הלחיצה יכולה להיות קצרה לכן יש לדגום אותה במכונת מצבים בקצב המהיר על ידי השעון של 50 MHz.
5. **פעולת TURBO** - תמופה לסוויץ', תאיץ את ספירת הפצצה לפי הקצב המוגדר ב- one\_sec\_counter.

## 4.1.3 ארכיטקטורה של הפצצה

1. **את הפצצה אנו נממש באמצעות ארבעה מודולים:**
  - a. **מודול של מכונת המצבים:** (ימומש בעבודת הכנה זו)
  - b. **מונה יורד של שתי ספרות:** BCDDN שמומש בסעיף קודם – ישמש לספירת הזמן עד לפיצוץ)
  - c. **מחלק תדר של 1Hz:** (קיים ממעבדה קודמת – ישמש להבהוב התצוגה פעם בשניה)
  - d. **שתי תצוגות 7Seg:** HEXSS מודול הקיים ממעבדה קודמת – ישמש להצגת הספירה)
2. **לצורך ספירת הזמן תשתמש ביחידה BCDDN** שכבר ממומשת כאשר יש לטעון לה את **זמן הפצצה** (על ידי שימוש בפרמטרים datainL ו-datainH)

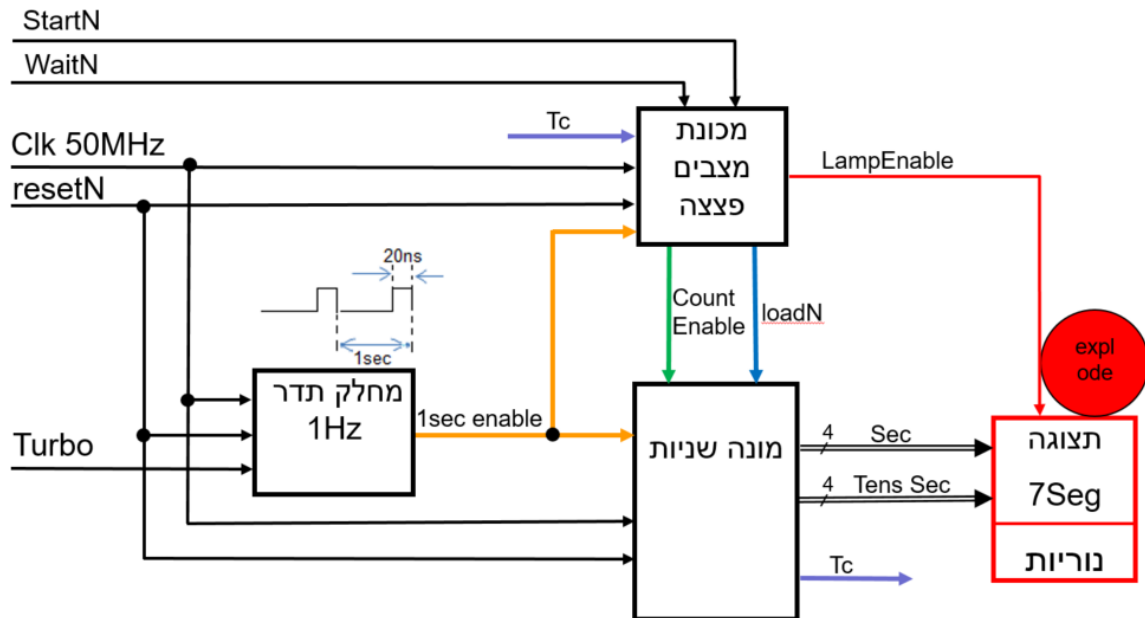
**הערה:** יש להקפיד להוסיף N כסיומת לאותות שהם active Low (לחצנים)



## 4.2 תכנון פרויקטון הפצה

**מטרה:** לאחר שהבנתם את דרישות התכנן עליכם לתכנן כל אחד ואחד מהמודולים - חלקם ימוחזרו מפרויקטים קודמים (בשינויים מתאימים) ובחלקם יש להשלים את הקוד בשלד הנתון.

**דיאגרמת הבלוקים** שלהלן מתארת את המודולים המרכיבים את המערכת וחלק מהקשרים ביניהם.



**רשמו באופן מפורט את כל אחד מהמודולים שתוצו למחזר מפרויקטים קודמים (השלימו את הטבלה):**

| שם               | הסבר פעולה                | שינויים דרושים                                                            |
|------------------|---------------------------|---------------------------------------------------------------------------|
| מונה שניות bcddn | מונה אחורה שניות          | נרצה לשנות את ערך ההתחלה של המונה ל25                                     |
| HEXSS            | תצוגה ספרתית              | לאחר הגעה ל00 נקבל lampEnable, אשר יכבה את התצוגה (בעזרת darkN) בתדר חצי. |
| מחלק תדר         | ממיר את תדר השעון ל1 [HZ] | אין שינויים                                                               |

**רשמו באופן מפורט את המידע לגבי המודול שתוצו להשלים בשלד קוד הנתון (מלאו את הטבלה):**

| שם          | הסבר פעולה   | כניסות עיקריות                                           | יציאות עיקריות                       |
|-------------|--------------|----------------------------------------------------------|--------------------------------------|
| מכונת מצבים | ניהול התהליך | startN<br>waitN<br>resetN<br>clk<br>tc<br>one sec enable | Lamp enable<br>Count enable<br>loadN |

## 4.2.1 תכנון של מודול הפצה

**בעבודת הכנה זו** אתם תממשו את מודול מכונת המצבים של הפצה לפי השלבים: תכנון, השלמת הקוד בשלד הנתון ובדיקת נכונות המימוש בסימולציה.  
**במעבדה** אתם תשלימו את הפרויקטון על ידי חיבור המודולים השונים יחד ובדיקת כלל המכלול על הכרטיס.

- תארו את הפעילות העיקרית של מודול מכונת המצבים של הפצה:

תשובה:

מקבל את הכניסות המתוארות לעיל, תפקידה לשלוט על המונה, תוך אפשרות וטעינה שלו בערך התחלתי, ניטור ערך המנייה כך שכאשר המונה יגיע ל"00" מכונת המצבים מתריעה לתצוגה וגורמת להבהוב.

- פרטו את הכניסות (ניתן להוסיף/להוריד שורות לפי הצורך):

| שם כניסה       | פעולה                                                                                                      |
|----------------|------------------------------------------------------------------------------------------------------------|
| resetN         | resetN - איפוס המונים והמכונה, count enable=0 כדי לא לאפשר ספירה עד שחרור ה-START.                         |
| startN         | loadN=0 כדי לטעון את הערך ההתחלתי "25" למונים. Count enable =1 כדי לאפשר את תחילת הספירה בעת שחרור ה-START |
| waitN          | Count enable =0, משהה את הספירה.                                                                           |
| Tc             | Count enable =0, משהה את הספירה על "00", ומדליק את lamp enable בתדר חצי ליצירת הבהוב.                      |
| Clk            | שעון הכרטיס 50M [HZ]                                                                                       |
| One sec enable | שעון שימש למימוש ההבהוב.                                                                                   |

- פרטו את היציאות (ניתן להוסיף/להוריד שורות לפי הצורך):

| שם יציאה     | פעולה                                              |
|--------------|----------------------------------------------------|
| Lamp enable  | אחראית להבהוב התצוגה לסירוגין כאשר המונה במצב "00" |
| Count enable | שולט על הספירה של המונה.                           |
| loadN        | טעינת המונה בערכו ההתחלתי.                         |

## 4.2.2 תכנון מכונת המצבים של מודול הפצה

**כעת, יהיה עליכם להגדיר את מכונת המצבים של מודול הפצה במפורש.**

השלימו בטבלה שלהלן את כל מצבי מכונת המצבים החסרים. פרטו מה עושים בכל מצב, מתי עוברים למצב הבא, מה היציאות שיש לעדכן ואיך:

| שם המצב הנוכחי | פעילות עיקרית                    | לאיזה מצב עוברים מהמצב הנוכחי ובאילו תנאים                                                          | יציאות שמעדכנים                            |
|----------------|----------------------------------|-----------------------------------------------------------------------------------------------------|--------------------------------------------|
| s_idle         | מאפסים את המונה וממתינים         | לחיצה על START מעבירה למצב ARM                                                                      | lampEnable מקבל 0 – תצוגה כבויה            |
| s_arm          | טעינת הערך ההתחלתי למונה והמתנה. | עזיבה של הלחצן START מעביר בין s_arm ל s_run                                                        | loadN=0<br>lampEnable =1<br>count enable=0 |
| S_run          | אפשר מנייה                       | ניתן לעבור למצב s_pause בעת לחיצה על WAIT וכן אם המונה מגיע ל"00" ( tc דולק) אז נעבור למצב s_HEX_ON | count enable=1<br>loadN=1                  |
| S_pause        | השהיית המנייה.                   | כאשר נעזוב את לחצן WAIT נעבור למצב s_run                                                            | count enable=0                             |

|                                 |                                                               |               |           |
|---------------------------------|---------------------------------------------------------------|---------------|-----------|
| lampEnable =0<br>count enable=0 | כאשר עוברת שנייה ( one sec enable נדלק) ונעבור למצב S_HEX_ON  | כיבוי התצוגה. | s_HEX_OFF |
| lampEnable =1                   | כאשר עוברת שנייה ( one sec enable נדלק) ונעבור למצב S_HEX_OFF | הדלקת התצוגה. | S_HEX_ON  |

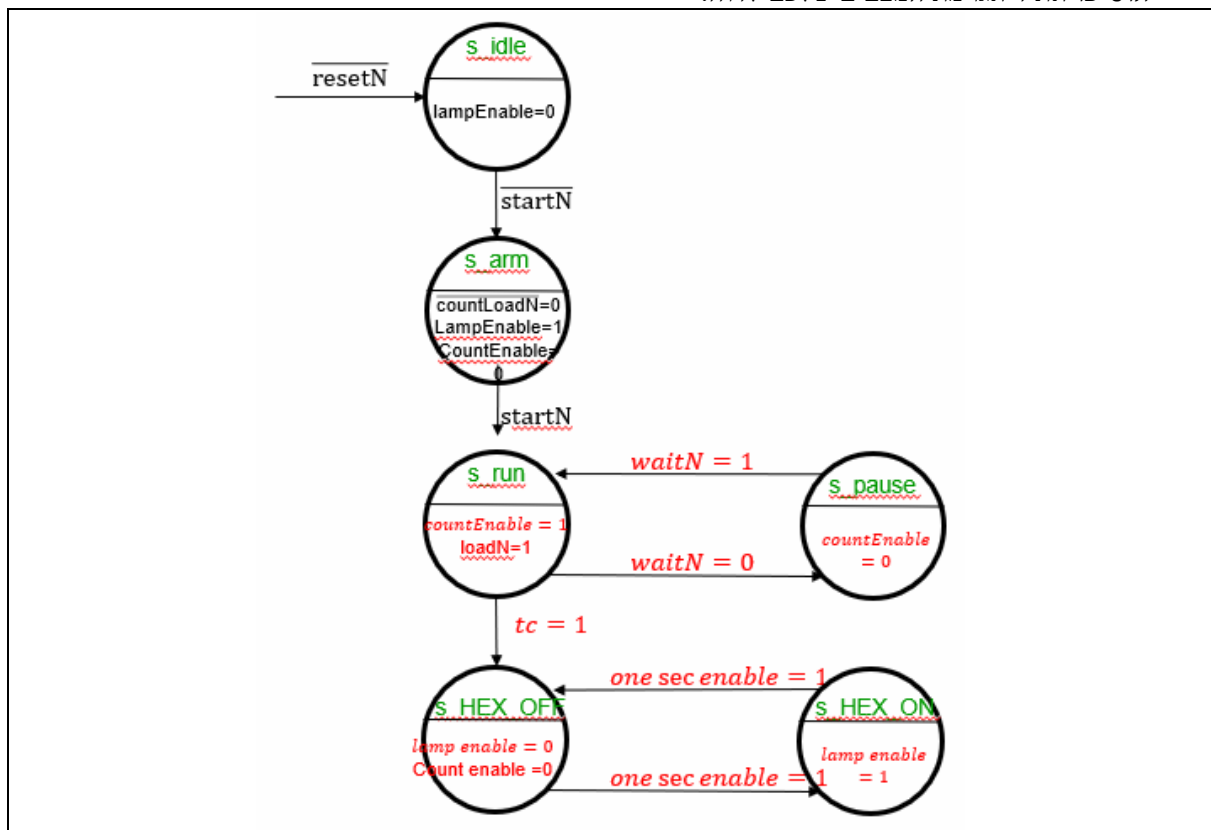
### דיאגרמת מצבים - State diagram

שרטטו את דיאגרמת המצבים של הפצצה - היעזרו בשבלונה המוכנה הנתונה במודל ושנו אותה לפי הצורך.

עליכם לפרט על גבי דיאגרמת המצבים:

- כל המעברים – יש לוודא שלכל קשת מוגדר התנאי למעבר.
- כל היציאות – פרטו רק את היציאות שיש לעדכן בכל מצב.
- יש לוודא ש- resetN מופיע רק פעם אחת בשרטוט (אסינכרוני) כחץ למצב ההתחלתי.

- הוסיפו את דיאגרמת המצבים שלכם לדוח:



### 4.3 מימוש מכונת המצבים של הפצצה

- פתחו את קובץ השלד של הפצצה הנתון בפרויקט (bomb.sv) והגדירו אותו כ- TOP LEVEL.

- השלימו את הקוד בשלד עם קוד שלכם על פי התכנון שלכם.

שימו לב: שימרו על קוד קצר ויעיל, הכתוב לפי הכללים לכתיבת קוד נאותה

- השתמשו בערכי Default ועדכנו בכל מצב רק את המשתנים המקבלים ערך שונה מה-Default

- בצעו אנליזה מוצלחת לתכן.

- הוסיפו את קוד הפצצה שלכם לדו"ח.

```

- // (c) Technion IIT, Department of Electrical Engineering 2018
- // Updated by Mor Dahan - January 2022
- //
- // Implements the state machine of the bomb mini-project
- // FSM, with present and next states
-
- module bomb
- (
- input logic clk,
- input logic resetN,
- input logic startN,
- input logic waitN,
- input logic OneSecPulse,
- input logic timerEnd,
-
- output logic countLoadN,
- output logic countEnable,
- output logic lampEnable
-);
-
- //-----
-
- // state machine declaration
- enum logic [2:0] {s_idle, s_arm, s_run, s_pause, s_lampOff, s_lampOn }
- bomb_ps, bomb_ns;
-
- //-----
-
- // 1. synchronous code: executed once every clock to update the current
- state
- always @(posedge clk or negedge resetN)
- begin
- if (!resetN) // Asynchronous reset
- bomb_ps <= s_idle;
- else // Synchronic logic FSM
- bomb_ps <= bomb_ns;
- end // always sync
-
- //-----
-
- // 2. asynchronuous code: logically defining what is the next state, and the
- ouptput
- //(not seperating to two different always sections)
- always_comb // Update nextstate and outputs
- begin
- // set all default values
- bomb_ns = bomb_ps;
- countEnable = 1'b0;
- countLoadN = 1'b1;
- lampEnable = 1'b1;
-
- case (bomb_ps)
- //Note: the implementation of the idle state is already
- given you as an example
- s_idle:
- begin
- lampEnable = 1'b0;
- if (startN == 1'b0) begin
- bomb_ns = s_arm;
- end
- end
- end // idle
-

```



```

- s_arm:
- begin
- countLoadN = 1'b0;
- if (startN == 1'b1) begin
- bomb_ns = s_run;
- end
- end // arm
-
- s_run:
- begin
- countEnable = 1'b1;
- if (timerEnd == 1'b1) begin
- bomb_ns = s_lampOff;
- end
- else if (waitN == 1'b0) begin
- bomb_ns = s_pause;
- end
- end // run
-
- s_pause:
- begin
- if (waitN == 1'b1) begin
- bomb_ns = s_run;
- end
- end // pause
-
- s_lampOff:
- begin
- lampEnable = 1'b0;
- if (OneSecPulse == 1'b1) begin
- bomb_ns = s_lampOn;
- end
- end // lampOff
-
- s_lampOn:
- begin
- if (OneSecPulse == 1'b1) begin
- bomb_ns = s_lampOff;
- end
- end // lampOn
-
- endcase
- end // always comb
-
- endmodule

```

#### 4.4 סימולציה של מודול הפצצה

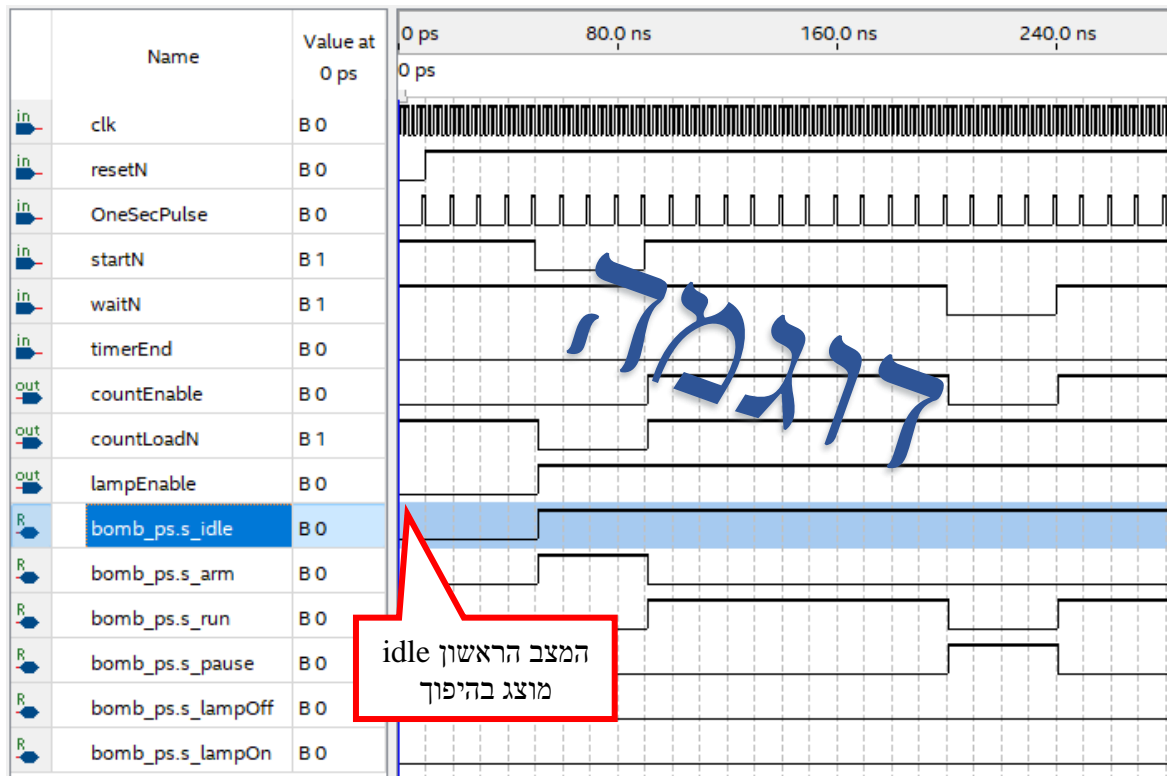
- הגדירו מה תרצה לבדוק בסימולציה – איזה מצבים מעניינים (מלאו את הטבלה והוסיפו שורות לפי הצורך):

| מצב            | תוצאות צפויות                                            |
|----------------|----------------------------------------------------------|
| יציאה מ- RESET | כל היציאות מאותחלות                                      |
| לחיצת startN   | הדלקת loadN=0 וטעינת הערך למונה                          |
| עזיבת startN   | אפשר מנייה                                               |
| לחיצת waitN    | השהיית המנייה count enable=0                             |
| עזיבת waitN    | אפשר מנייה count enable=1                                |
| הגעה ל"00"     | בדיקת התנהגות הסיגנל lamp enable, הדלקה וכיבוי לסירוגין. |

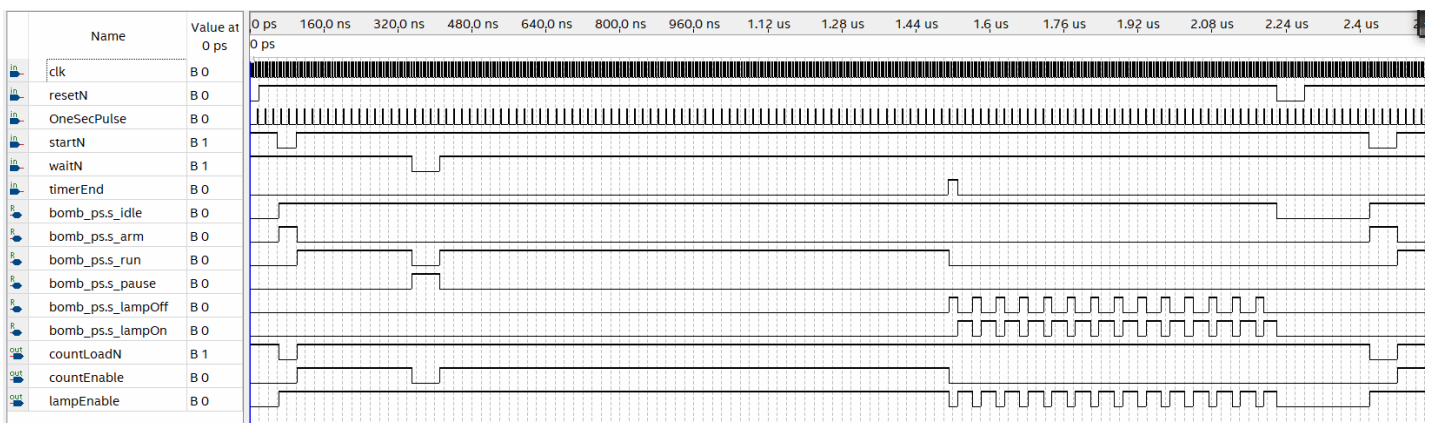
**שימו לב:** כדי לסמליץ שעון מהיר ו"שעון איטי" (כלומר כניסת סיגנל החיווי OneSecPulse) ניתן להגדיר בסימולטור את clk ואת OneSecPulse כשני שעונים עם קצבים שונים, למשל הבדל של פי 5 ביניהם. לדוגמה, אם נגדיר עבור clk: period = 2nsec, period = 10nsec: OneSecPulse, נוכל להגדיר עבור

**להזכירכם:** OneSecPulse צריך להיות פולס צר (סיגנל חיווי ולא שעון ממשי) – לא גל סימטרי – לכן יש לקבוע לו Duty cycle קטן מ- 50% (למשל 12%).

**שימו לב:** בסימולציה המצב הראשון, Idle, יוצג הפונד כפעיל בנמוך (inverse). יתר המצבים יופיעו בצורה תקינה - פעילים בגבוה. הצג כל מצב בשורה נפרדת כמו בדוגמה להלן:



- הריצו סימולציה של מכונת המצבים והוסיפו את תוצאות הסימולציה לדו"ח.



## 5 גיבוי העבודה

שמרו דו"ח זה גם כקובץ WORD וגם כ- PDF והעלו את קובץ ה- PDF למודל.

שמרו את הפרויקט וגם צרו ממנו קובץ ארכיב (באמצעות Project -> Archive Project). שימו לב לשנות את השם שמציע הקוורטוס לשם קצר יותר שאינו מכיל: תווים בעברית, רווחים ו/או את הסימן '!' ומכיל את התאריך (ואפשר גם את השעה) של הגיבוי, למשל SV2\_PreWork\_13\_01\_2026 העלו את קובץ הארכיב (.QAR) למודל, כיוון שתצטרכו אותו בניסוי. להזכירכם - יש להביא עמכם למעבדה את כל הקבצים – כיוון שתשתמשו בהם. מומלץ לגבות את הדו"ח והפרויקט גם באמצעים אחרים.

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

מלא את הטופס