סקירה על 2 סוגי המנועים

בניגוד למנוע צעד החד-קטביים, יחדות דו-קוטביות דורשות בקר יותר מתחוכם. מנועים דו-קטביים מתאפינים ביחס גודל/מומנט פיתול שלהם ומספקים ממונט פיתול גבוהה יותר מאשר מנועים חד-קטביים. מנועים דו-קטביים בנויים מליפופים מופרדים אשר צריכים להיות מופעילם בכיוונים שונים(יש צורף בשינוי הקוטביות בזמן הפעולה) כדי ליצור את הרצף הנכון. כן בא האתגר של הבקר. מנועי צעד דו-קטביים משתמשת באותה תבנית בינרית כמו מנועים חד-קטביים כאשר ה-1 וה-0 מיצגים את הקוטבים ולא הפעלה וכיבוי.
(נלקח מאתר "רובוטיקה")

ביצוע צעד שלם מתבצע ע"י שליחת פולס דרך קו יחיד
וביצוע חצי צעד מתבצע ע"י שימוש ב 2 פאזות צמודות וע"י כך כך נוצר מומנט שמחזיק את כיוון המנוע במצב "ביניים" ניתן לראות זאת באיור המצורף.

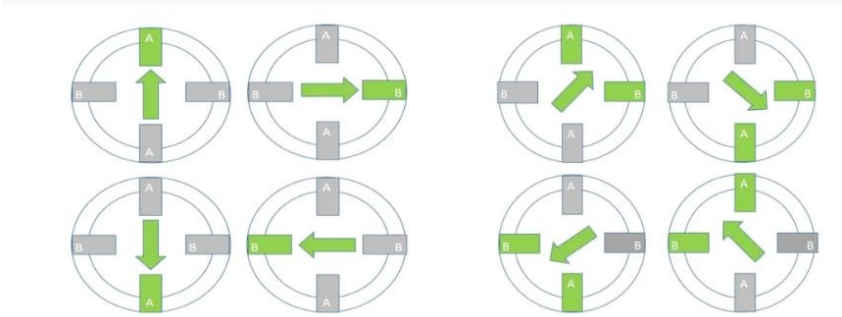


Fig 1 – One phase on – full step Fig2 – Two phase on – full step

האיור הבא מתאר את רצף כל פעולות המנוע בחצי צעד.

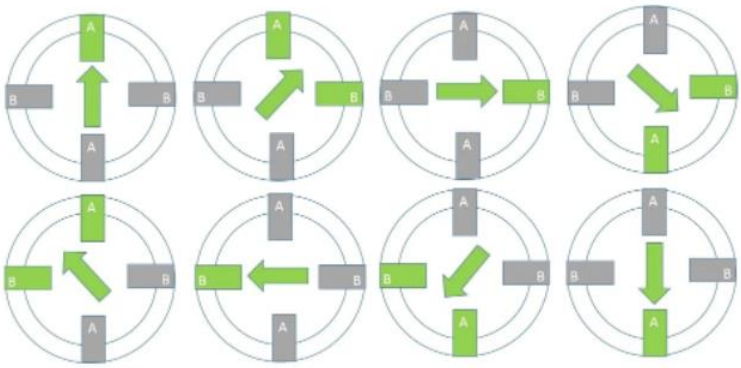
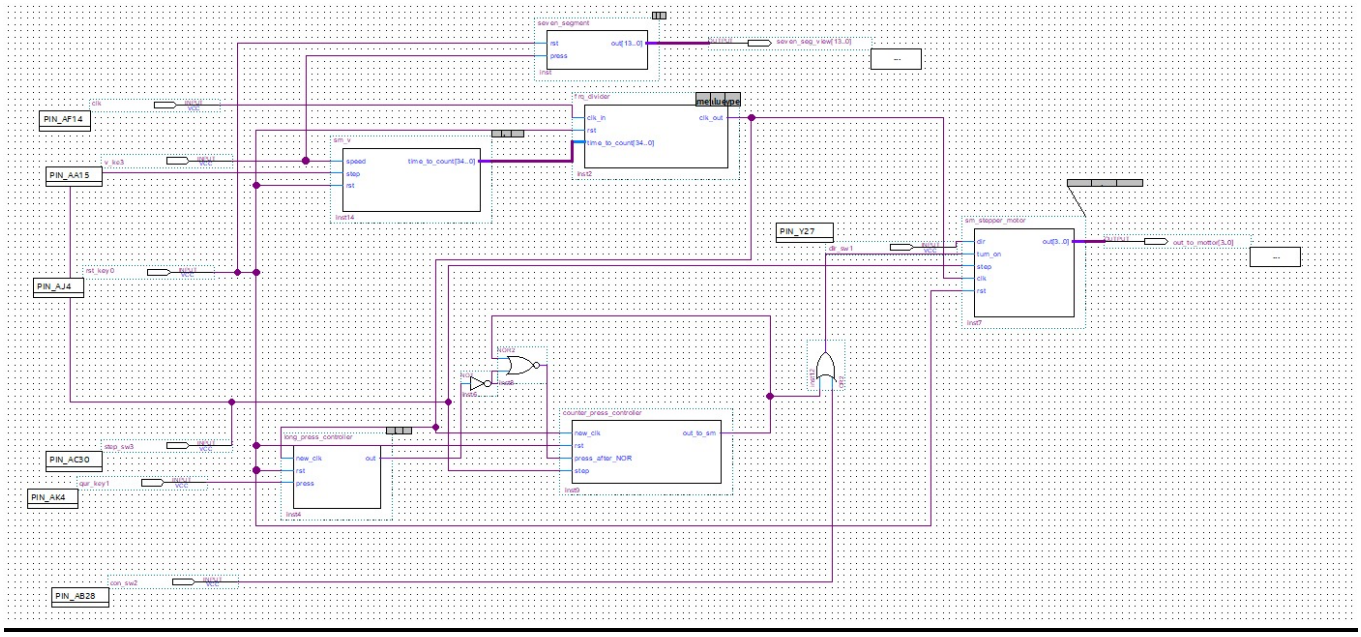


Fig3 - One-two phase on - half step

סכימת הפרוייקט:



ראשית נציין את המתגים והסוויצים שהגדרנו :
<u>כיוון התנועה</u> : sw1 (ימינה =1)
<u>שליטה על המהירות</u>: key3
<u>עבודה רציפה</u>: sw2 (רציפה =1)
<u>הפעלה רגעית</u>: key1
<u>גודל הצעד</u>: sw3 (צעד שלם/step =1)
<u>אתחול (rst)</u>: key0

<u>חיבור מהמנוע לשנאי:</u>

yellow	Out1
BLUE	OUT2
GREEN	OUT3
RED	OUT4

כחול וצהוב הם זוג (out2
אדום וירוק זוג

<u>חיבור שנאי לכרטיס:</u>

מהשנאי	החיבור של השנאי	לכרטיס	החיבור של הכרטיס
purple	In1		GPIo0
Blue	In2		GPo2
Green	In3		1GPo
Yellow	In4		3GPo

Dir=1 == ימינה

סרטון פעולת המנוע :
https://www.youtube.com/watch?v=PZxoFN5Q81s

מערכות הבקרה על רבע הסיבוב

הבלוק של long_press_count

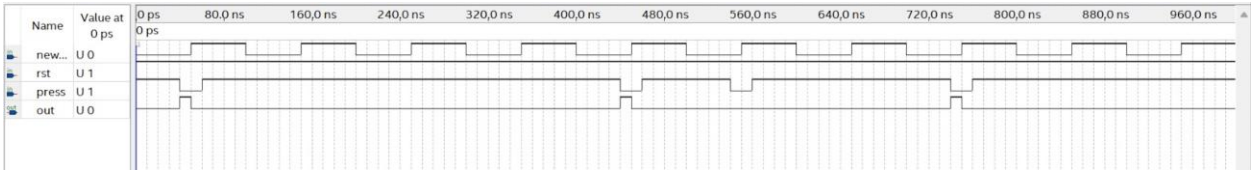
מערכת בקרה ללחיצה ארוכה :

בבלוק זה :
כניסות :
Rst,new_clk
יציאה:
press

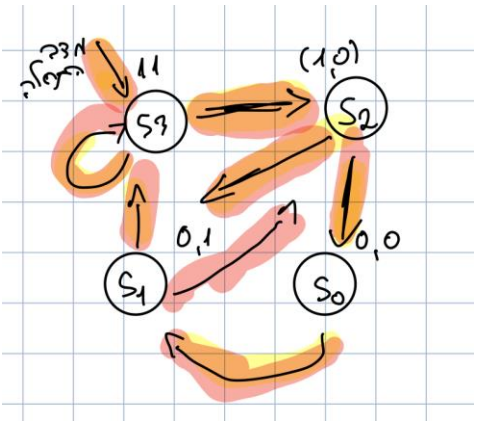
בבלוק זה מימשנו מכונת מצבים שתוציא לנו פולס יחיד - רק כאשר לא לחצנו ולאחר מכן לחצנו (למנוע לחיצה ארוכה)
למצב זה קראנו "s2"
נדגיש כי לחיצה מיוצגת ע"י הספרה "0" , ובמצב "שגרה" מיוצג על ידי "1"
ניתן לראות בסימולציה את שאר המצבים .

עבור המצבים הבאים לא נקבל יציאה עם ערך 1
S0 - (0,0) מצב של לחיצה ארוכה שבו לא נרצה להוציא פולס
S1 - (0,1) הרגע לאחר הלחיצה שבו לא נרצה פולס
S3 - (1,1) מצב של שאין לחיצות כלל

סימולציה :



תמונה לבדיקת הסימולציה כך שנעבור על כל המצבים האפשריים :

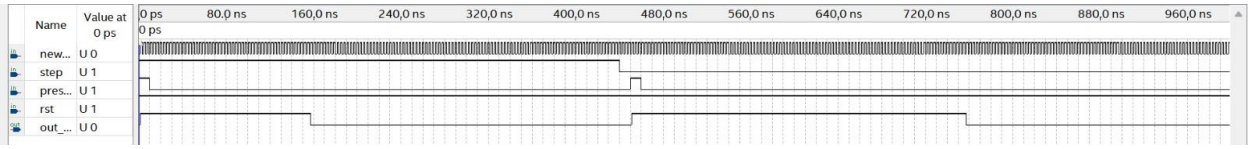


הבלוק : counter_press_controller

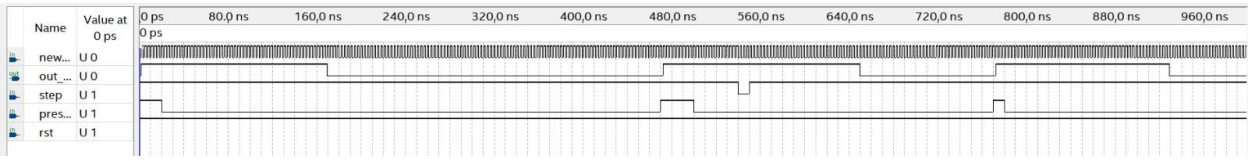
בבלוק זה :
כניסות :
Rst,new_clk,press_after_nor,step(full/half_step)
יציאה:
Out_to_sm

בלוק זה מומש על מנת לוודא שהמנוע יסתובב רבע מעגל .
בנינו קאונטר עבור בקרת הלחיצה כך שבמקרה של חצי צעד הקאונטר יוציא לנו פולס עבור במשך 99 מחזורי שעון, ועבור צעד שלם יתן לנו פולס במשך 49 מחזורי שעון.***
***נשים לב כי לאחר בקרת שעון גילינו שיש פער של מחזור שעון (המנחה הנחה אותנו לא להתעסק עם זה)

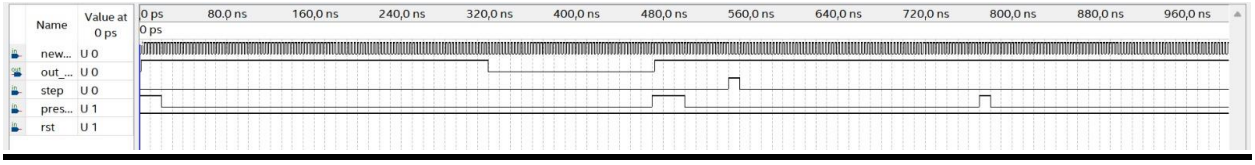
סימולציה :
ראשית נבדוק את הפונקציונליות הבסיסית של הבלוק :



נבחין כי הבלוק הנ"ל לא אחראי על השעון לכן בשינוי הצעד תוך כדי העבודה לא נוכל להבחין בשינוי השעון בסימולציה , אך בכל מקרה בדקנו האם יש תופעה לא רצויה בשינוי גודל הצעד תוך כדי פעולה .
הסימולציה הנ"ל נותנת דגש עבור "צעד שלם":



הסימולציה הנ"ל נותנת דגש עבור "חצי צעד" :



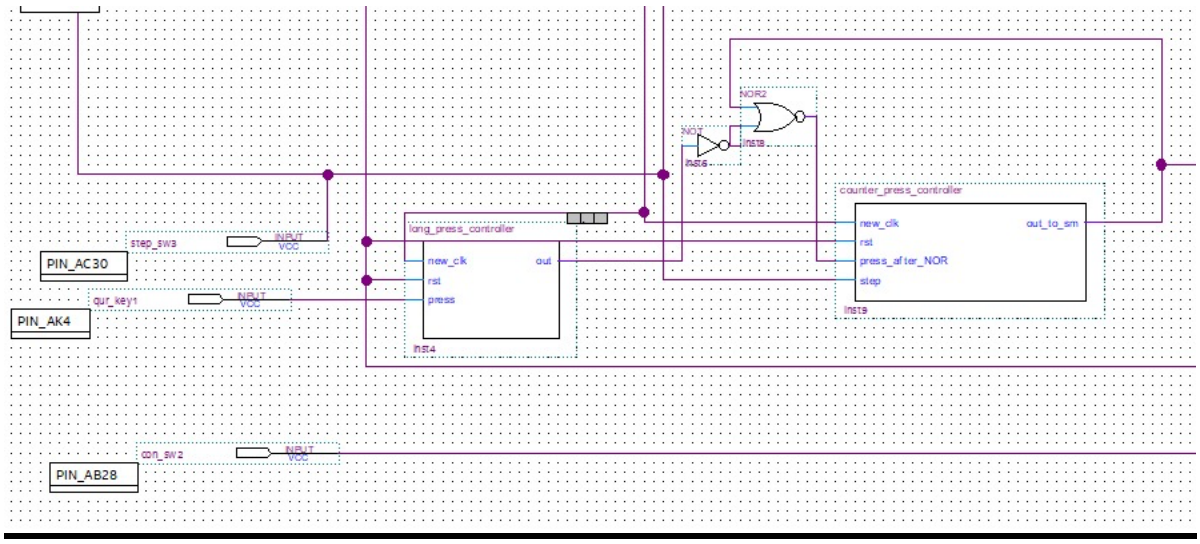
כך וידאנו כי עברנו על כל המצבים האפשריים .

הסבר על החומרה בבלוק על בקרות רבע הסיבוב:

לבדוק
מטרת החומרה היא להוציא לנו פולס למכונת המצבים (״sm_stepper_motor״) רק כאשר המנוע לא מסתובב וגם מתקבלת לחיצה בודדת על לחצן רבע הסיבוב.
את זאת מימשנו בצורה הבאה

הפלט של הבלוק ״counter_press_controller ״ מוציאה ״-לוגי1״ כאשר המנוע מסתובב ו״-לוגי0״ כאשר אין פעולה של המנוע .

היציאה מהבלוק ״long_press_count ״ עוברת דרך מהפך ולאח״כ נכנסת לשער ״nor״ כאשר הכניסה השניה של ה-״nor״ מגיעה מהיציאה של ״counter_press_controller ״ .
רק כאשר הפלט של ״counter_press_controller ״ הוא 1 והיציאה של ״ counter_press_controller״ הוא 0 שער ה-״nor״ יוציא לנו 1 ויגרום ל ״ counter_press_controller״ להתחיל לספור .



יצרת התדר המבוקש

נשים לב כי הכרטיס עובד עם שעון בתדר של 50[MHz]

בלוק: sm_v

זהו בלוק שמטרתו להוציא את המספר הנכון בו השעון שאנו יוצרים בבלוק לאחר מכן ייתן לנו שעון נכון ע"פ מצב המהירות :

כדי להגיע ל ttc במשוואה הבאה :

$$\frac{Freq\ of\ the\ Fpga}{New\ freq * 2} = time\ to\ count\ (ttc)$$

את התדר החדש חישבנו לפי כמה קליקים לשניה נזדקק לעשות :

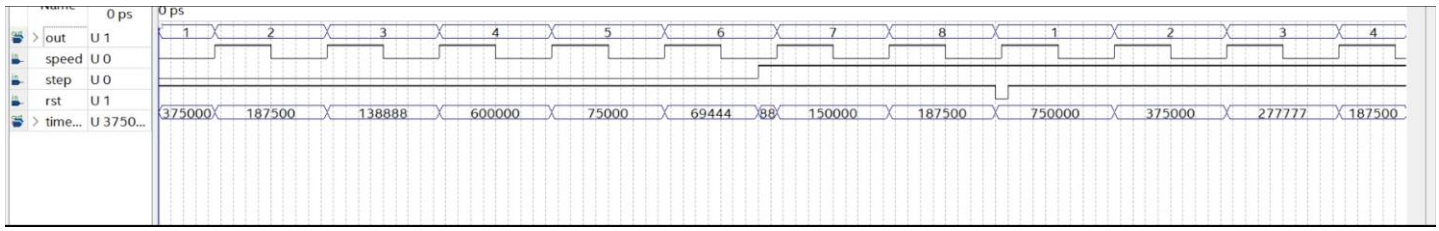
$$New\ freq = \frac{freq\ of\ the\ Fpga}{60\ sec} * number\ of\ step\ in\ a\ circle$$

בקוד השתמשנו במכונת מצבים המשקללת את גודל הצעד והמהירות
הכנסנו 9 מצבים – יש לנו מצבים מקבילים ולכן השתמשנו רק ב-9 מצבים
הגדרנו משתנים בהם לכל מצב יהיה ערך מתאים .

הקוד מוציא לנו את ערך ה - ttc .

סימולציה :

לצורך הסימולציה בלבד הכנסנו משתנים ע"מ לעקוב אחרי התוצאות :
ניתן לראות כי אנו עוברים ממצב למצב כפי שצריך



עבור כל לחיצה על המקש המהירות נקבל החלפת ערך ביציאה וכאשר נלחץ על ריסט נחזור למצב הראשון שלנו שיסווג את היציאה לפי צעד או חצי צעד (מהירות של 10 סיבובים לדקה).

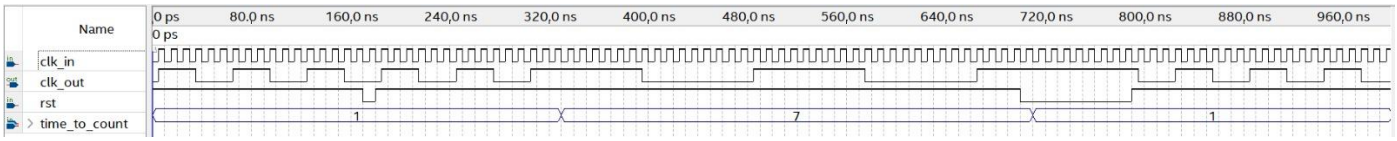
בלוק: freq_div

בבלוק זה
כניסות :
Rst,clk_in,time_to_count
יציאה:
clk_out

הבלוק מקבל ערך לספירה מהבלוק "sm_v" , סופר עד הערך הזה וכל פעם שמגיע לערך זה ,ערך היציאה מתחלף ,וכך אנו יוצרים שעון בתדר המבוקש.
ע"י לוגיקה זו אנו מוודאים שרוחב ה-"0 לוגי" הוא כרוחב ה-"1 לוגי" בשעון החדש שנוצר .
כמו"כ בחישוב ה-"$time\ to\ count$ " התחשב בלוגיקה זו (הכפלה בחצי).

בלחיצה על "rst" נאפס את הספירה והשעון והקאונטר יספור מחדש
הוספנו הגנה – יצרנו פרמטר שגבוה מכל הערכים שנקבל מהבלוק הקודם, כך שאם נגיע לערך זה הספירה תתחיל מחדש .
הגנה זו באה למנוע מצב שכאשר ה- "time_to_count " יתחלף לערך קטן יותר בעקבות שינוי מהירות, הקאונטר לא יספור עד אינסוף.

<u>סימולציה :</u>
לצורך הסימולציה הכנסנו ערכים שונים לתוך ה ttc וזאת ע"מ להתגבר על השעון הגדול.
בסימולציה ניתן לראות כי השעון החדש משתנה כתלות ב "ttc" וכאשר אנו משלבים את כפתור הריסט הוא משמר מצב קיים ולא גורם לפעילויות בלתי צפויות כלומר הוא יתחיל לספור מחדש ישר כאשר נעזוב את כפתור הריסט.



<u>הבלוק: seven segment</u>

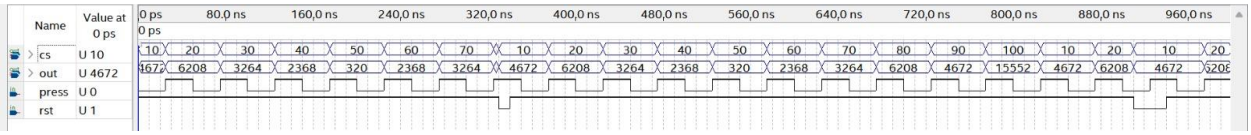
בבלוק זה
כניסות : ,press rst(שינוי המהירות)
יציאה:
רגיסטר של 14 ביטים (7 ביטים ראשונים ספרת האחדות ו7 לאחר מכן ספרת העשרות - Out

יצרנו מכונת מצבים בעלת 10 מצבים הכוללים את המהירויות של עליה 10 ל60 (6 מצבים)
ו- ירידה מ60 ל- 20 (4 מצבים)

כל לחיצה על "press" נעבור למצב הבא .

נבחין כי בלוק זה איננו עובר דרך "sm_steper_mottor" אלא מחובר ישירות ליציאה לכרטיס

סימולציה :
לצורך הסימולציה נוציא גם את פלט המצבים כלומר הבלוק יפלוט בנוסף ל "out" את "cs"
כמו"כ ניתן לראות בסימולציה כי הערכים של "out" תואמים את "cs"



ע"מ להבין את המצבים ("cs") נצרף את המילון:

```
A0 = 14'd10,
A1 = 14'd20,
A2 = 14'd30,
A3 = 14'd40,
A4 = 14'd50,
A5 = 14'd60,
A6 = 14'd70,//= 50
A7 = 14'd80,//=40
A8 = 14'd90,//=30
A9 = 14'd100;//=20
```

בלוק מכונת המצבים של המערכת :

שם הבלוק:
Sm_steper_controller

כניסות :
Rst ,clk ,step ,dir ,turn_on
יציאה:
Out

בלוק זה הוא מכונת מצבים שתפקידה לשקלל את הנתונים (צעד שלם או חצי צעד, כיוון ,והאם "turn_on " דלוק), ובהתאם לכך לעבור למצב הנכון ששולח את הפולס שמסובב את המנוע .

לבלוק זה נכנס אות בקרה שמשקלל את התנאים לסיבוב או אי סיבוב המנוע (ראה הסבר על מערכת " turn_on " בעמוד 13).

לוגיקת מכונת המצבים ממומשת ע"י אופרטור טרינארי, ראשית המכונה בודקת את כיוון התנועה ולאחר מכן את גודל הצעד ובהתאם בוחרת את המצב הבא.
המצבים במכונה שלנו :
S[1,2…8]
לכל מצב נשלח פולסים בהתאם ע"פ התרשים הנ"ל:

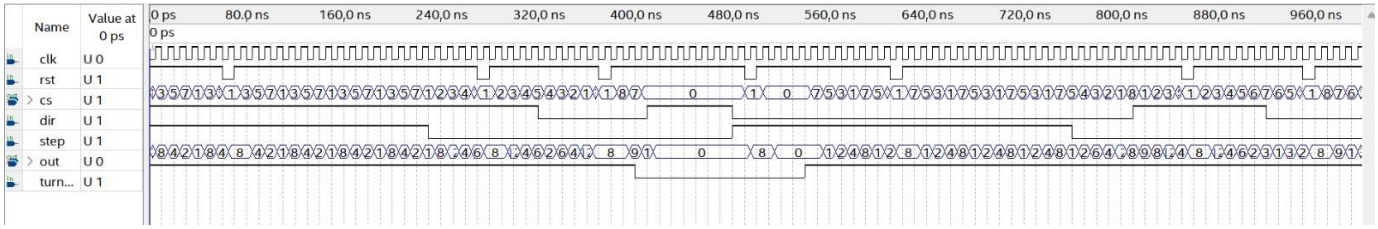
Step	Phase A	Phase B	Phase A'	Phase B'
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

כאשר מכונת המצבים לא פעילה (turn_on == 0) נעבור למצב s9 שבו ערך הפלט הוא "0000" כלומר המנוע לא יקבל מתח וכך נמנע חימום יתר של המנוע .

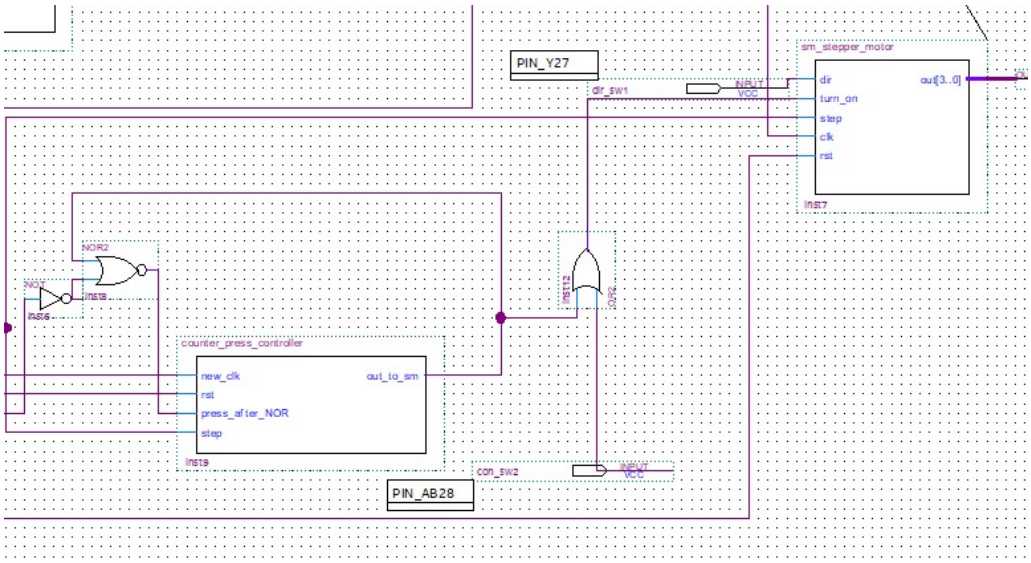
כמו"כ יצרנו מכונת מצבים עבור הפלט של הבלוק כך שערך רגיסטר היציאה משתנה בהתאם למצב מכונת המצבים שמשקללת את הנתונים .

סימולציה:

לצורך הסימולציה שינינו את "cs" של מכונת המצבים להיות פלט וכך נוכל בקלות גדולה יותר לעקוב אחר המצבים במכונה .
נשים לב כי "cs" תואם לפלט הבלוק "out"
נשים לב כי נבדקו כל האופציות האפשריות בבלוק הנ"ל
כלומר לכל מצב פעולה שינינו את כל הכפתורים, כלומר כל הפרמוטציות.



הסבר על מערכת "turn_on " :



מטרת החומרה היא להוציא לנו פולס למכונת המצבים רק כאשר המנוע פועל על מצב קבוע או המנוע פועל על מצב "רבע-סיבוב"
כלומר השער הלוגי "or" מקבל מהסוויץ הבורר הפעלה קבועה ומהבקרות של רבע הסיבוב
ולפי הלוגיקה של הפרוייקט שלנו:
כאשר הסוויץ "דלוק" אנו נצפה כי המעגל ינוע ללא תלות בלחיצה על לחצן רבע הסיבוב.
כאשר שניהם כבויים נצפה שהמנוע לא יסתובב .
כאשר הסוויץ "כבוי" אנו נצפה כי המעגל ינוע כתלות בלחיצה על לחצן רבע הסיבוב ובבקרות שלו.

לסיכום :

מאוד נהנו לתכנן מערכת שרואים בפועל את התוצאות שלה באופן חומרתי .

גילינו גם כי למנוע הצעד יש המון פונקציונליות בשוק למשל : מדפסת תלת מימד ובמכשור רפואי, ובנוסף הבלאי שלו מאוד נמוך.
והסרטון ביוטיוב ישמר לזיכרון לעולמי-עד.