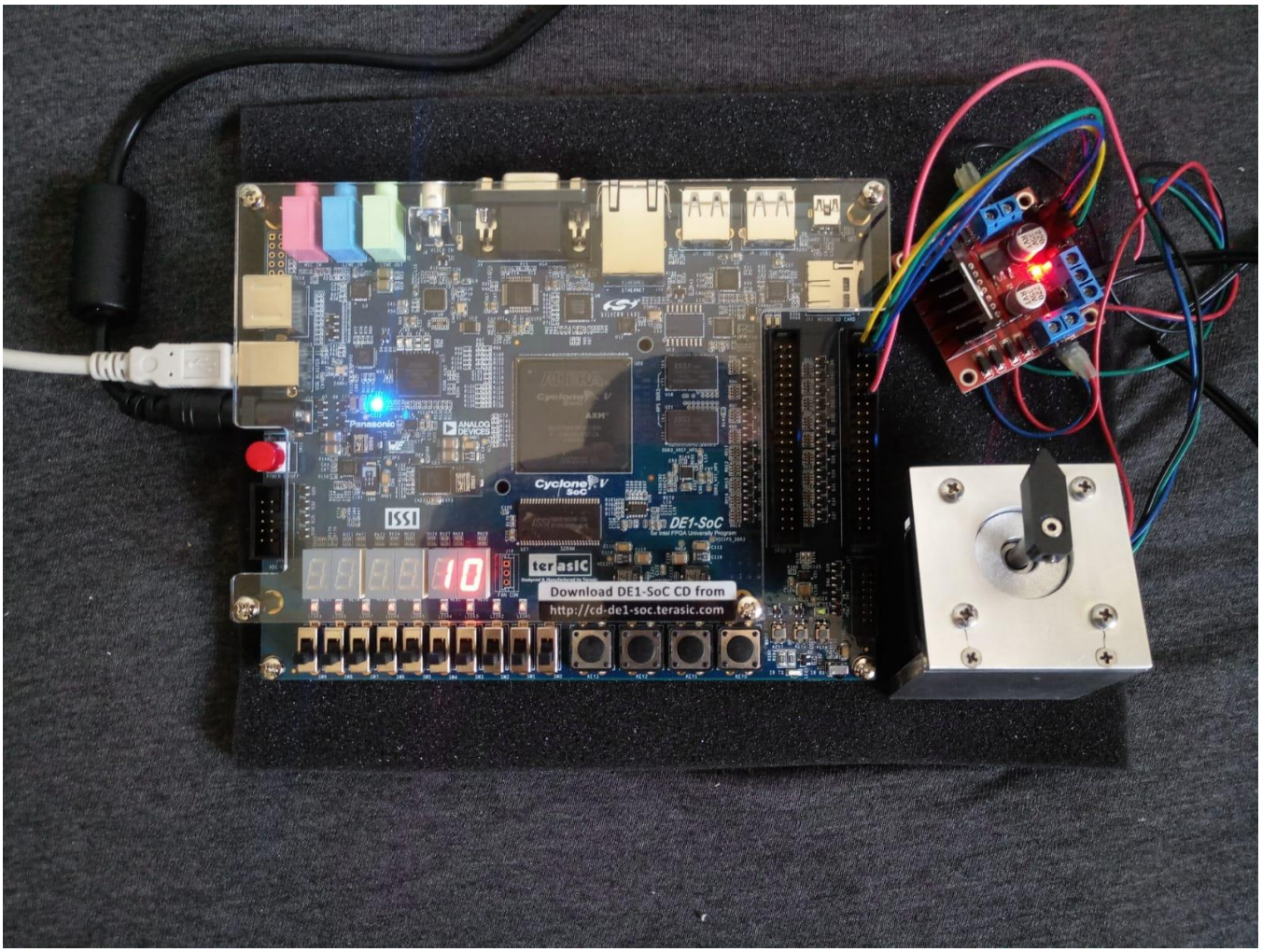
***Step Motor Controller* – 2 מעבדה**



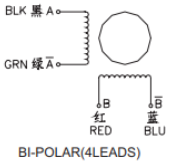
גלעד בינו 302252101

יואב אשד 305384869

מבוא:

מנוע צעד הינו מנוע DC שנע בצעדים בדידים קבועים. למנועים האלו מספר סלילים שמאוגדים בקבוצות הנקראות פאזות. על ידי הפעלת כל פאזה ברצף , המנוע מסתובב, צעד אחד בכל פעם.

המנוע בו אנו משתמשים הינו 17HS3430 והוא מסוג $BIPOLAR$, כלומר יש שני סלילים , אחד עבור כל קוטב, ולכל סליל יש שתי כניסות מתח, אחת בכל צד, המסודרים כפי שניתן לראות באיור.



במרכז המנוע נמצא ציר שבסיסו מגנטי , וכאשר מפעילים שדה מגנטי ,

המנוע מבצע סיבובים לפי כיוון המתח המופעל על הסלילים.

כיצד גורמים למנוע להסתובב בצעד מלא מול חצי צעד: בצעד מלא זורם זרם בסליל אחד בלבד, ועבור חצי צעד זורם זרם פעם בסליל אחד ופעם בשני סלילים. מנוע חצי צעד נחשב למדוייק יותר כיוון שניתן להזיז את המנוע בזוויות קטנות יותר, אך הוא מבזבז יותר אנרגיה עקב כך שהוא מצריך הזרמה בשני סלילים בו זמנית מפעם לפעם. עבור צעד מלא קיימים ארבע מצבי הזרמת מתח שונים , ועבור חצי צעד קיימים שמונה מצבים שונים,

כעת נדבר על גדלי צעדי המנוע. צעד אחד במנוע מסתובב ב-1.8 מעלות כך שסה"כ בסיבוב מלא של המנוע נגיע ל-200 צעדים. כפי שציינו, מאחר ובמנוע צד ישנה אפשרות לנוע בחצאי צעדים, במצב זה יהיו בסיבוב שלם 400 צעדים. ידיעת מספר הצעדים בסיבוב בכל מצב מסייעת לנו בבחירת גודל הסיבוב עצמו. לדוגמה במעבדה זו, כפי שנתאר גם בהמשך, התבקשנו ליישם סיבוב של המנוע בגודל 90 מעלות (רבע סיבוב) והדבר מושג על ידי הזזת המנוע ב-50 צעדים כאשר אנו במצב של צעד מלא או ב-100 צעדים כאשר נבחר לזוז בחצאי צעדים.

טבלאות אמת עבור תזוזה של המנוע, על פי סוג צעד וכיוון, כאשר הצבעים מייצגים את הסלילים שדרך זורם זרם וממופים לפי מפרט המנוע:

עבור צעד מלא בכיוון השעון:

כחול	אדום	ירוק	שחור	
0	0	0	1	A
0	1	0	0	B
0	0	1	0	C
1	0	0	0	D

עבור צעד מלא נגד כיוון השעון:

כחול	אדום	ירוק	שחור	
1	0	0	0	A
0	0	1	0	B
0	1	0	0	C
0	0	0	1	D

עבור חצי צעד בכיוון השעון:

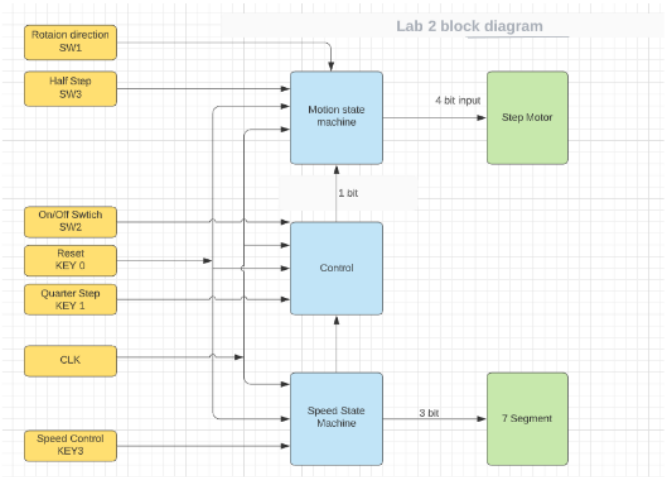
כחול	אדום	ירוק	שחור	
0	0	0	1	A
0	1	0	1	AB
0	1	0	0	B
0	1	1	0	BC
0	0	1	0	C
1	0	1	0	CD
1	0	0	0	D

עבור חצי צעד נגד כיוון השעון:

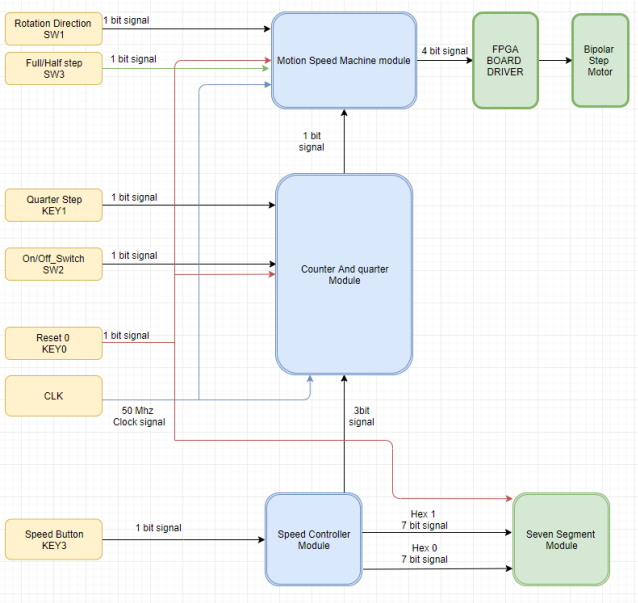
כחול	אדום	ירוק	שחור	
1	0	0	0	A
1	0	1	0	AB
0	0	1	0	B
0	1	1	0	BC
0	1	0	0	C
0	1	0	1	CD
0	0	0	1	D

תכנון המנוע -Design

דיאגרמת הבלוקים איתה התחלנו:



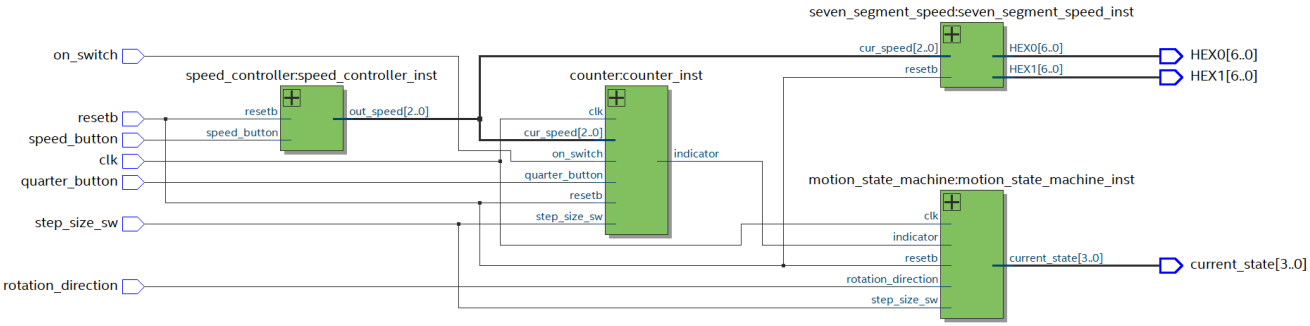
דיאגרמת הבלוקים של המערכת הסופית:



שינויים שנעשו בין תחילת התהליך לסופו:

בשרטוט ההתחלתי לא כללנו את מודול רבע הסיבוב, לאחר הבנת אופן פעולת מודול המונה שמוציא פולסים אל מכונת המצבים שאחראית על התנועה, וכן את פעולת רבע הסיבוב , העדפנו לממש את מודול רבע הצעד בתוך מודול הקאונטר על מנת להימנע מהתעסקות סנכרון יחידות המונה ויחידת רבע הסיבוב.

לטובת המחשה אנו מצרפים גם את שרטוט הRLT VIEW שנוצר על ידי הקווארטוס.



אותות הכניסה והיציאה:

כניסה

- on_switch - מתג הקובע אם המעגל פועל בתנועה רציפה או לא.
- $Speed\_button$ – כפתור השולט על מהירות הסיבוב. לחיצה מעלה או מורידה את המהירות בעשרה סיבובים לדקה, וכאשר מגיעים למהירות הגבוהה ביותר (60) או הנמוכה ביותר 10 (כיוון העלייה או הירידה מתהפך).
- $Quarter\_button$ – כפתור שגורם למנוע לבצע רבע סיבוב ולעצור, רק בתנאי שהמתג on_off_sw כבוי.
- $resetb$ - כפתור שאחראי על איפוס המערכת.
- clk – שעון חיצוני שמגיע מהFPGA ומתזמן את המעגל.
- $step\_size\_sw$ - כפתור שאומר למנוע להסתובב בצעד מלא או בחצי צעד.
- $rotation\_direction$– מתג למנוע לאיזה כיוון להסתובב – עם או נגד כיוון השעון.

יציאה

- $HEX0$ רגיסטר ש באורך שבעה ביטים המייצג את ספרת היחידות של המהירות בסיבובים לדקה , מאחר ואנו עובדים במהירויות עגולות שמשנות בגודל 10, היציאה מוחזקת על מצב 0 ($7b'100\_0000$)
- $HEX1$ רגיסטר באורך שבעה ביטים המייצג את ספרת העשרות של המהירות בסיבובים לדקה,
- $Current\ state$ רגיסטר באורך ארבעה ביטים המייצג את מתחי ההזנה למנוע, נקבעים על ידי המודול $motion\ state\ machine$

חוטים פנימיים:

- Indicator – חוט שמגיע למכונת התנועה או ממודול הקאונטר או ממודול הקוורטר ומכתיב למנוע מתי לבצע צעד.
- Speed state – חוט פנימי המעביר את המהירות באות בגודל שלושה ביטים.

חלוקת המערכת למודולים:

-מודול המהירות $speed\ controller$

מכונת המצבים ששולטת על המהירות, המהירות נקבעת על ידי לחיצת כפתור.

כניסות: $resetb, speed\ button$

יציאות : out_speed : יציאה פנימית במעגל שמעבירה לקאונטר ולקוורטר את המהירות.

-מודול התצוגה $seven\ segment$:

מודול התצוגה אחראי להציג את המהירות באמצעות התצוגה הקיימת על הכרטיס.

אותות כניסה : cur speed ,reset

אותות יציאה: $Hex0, Hex1$, אחת לכל ספרה את המהירות שצריכה להיות מוצגת.

-מודול השליטה והתזמון -counter:

הקאונטר אחראי לספור עליות השעון עבורן הוא צריך לבצע סיבוב ומוציא פולס למכונת התנועה עבור כל צעד של המנוע. בנוסף, הוא גם אחראי לספירת הצעדים במצב של רבע סיבוב.

אותות כניסה : $resetb, clk, step\ size, on\_switch, quatrer\ button\ , cur\ speed$

אותות יציאה: $indicator$.

-מודול התנועה motion state machine:
המודול הנ"ל אחראי על תנועת המנוע, מכתיב למנוע איך להסתובב באיזה גודל צעד ובאיזה כיוון ומוציא אות במוצא שעובר דרך הFPGA לדרייבר ומהדרייבר למנוע.

אותות כניסה : $rotation\ direction, indicator\ resetb, clk, step\ size, step\_size\_sw,$

אותות יציאה: $current\ state$.

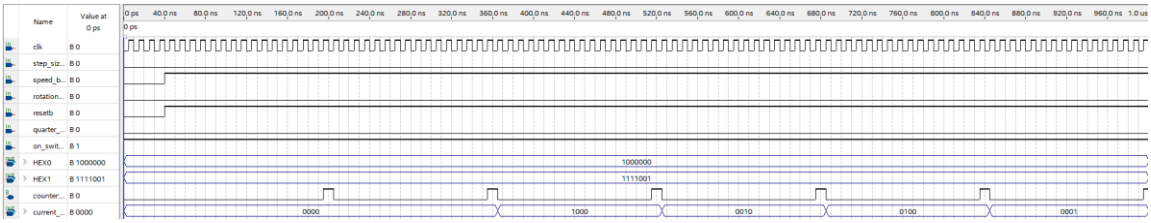
מודול הטופ step motor top:

- מודול שמחבר את המעגל ביחד, על ידי יצירת אינסטנסים של כל שאר המודולים במערכת וחיווט שלהם ביחד.
- כניסות :
$CLK, resetb, speed\ button, quater\ button, step\ size\ sw\ , rotation\ direction\ , on\ switch$
- יציאות : $HEX0, HEX1, Current\ state$

מימוש הtop module

```
module step_motor_top(clk, resetb, speed_button, quarter_button, step_size_sw,
                      rotation_direction, on_switch, current_state, HEX0, HEX1);

    input wire clk, resetb, step_size_sw, rotation_direction, on_switch,
              speed_button, quarter_button;
    wire indicator;
    wire [3:0] cur_speed;
    output wire [6:0] HEX0;
    output wire [6:0] HEX1;
    output wire[3:0] current_state;

    speed_controller speed_controller_inst(.resetb(resetb), .speed_button(speed_button), .out_speed(cur_speed));

    seven_segment_speed seven_segment_speed_inst(.resetb(resetb), .cur_speed(cur_speed), .HEX0(HEX0), .HEX1(HEX1));

    motion_state_machine motion_state_machine_inst(.clk(clk), .resetb(resetb), .rotation_direction(rotation_direction), .step_size_sw(step_size_sw), .current_state(current_state), .indicator(indicator));

    counter counter_inst(.clk(clk), .resetb(resetb), .cur_speed(cur_speed), .on_switch(on_switch), .step_size_sw(step_size_sw), .quarter_button(quarter_button), .indicator(indicator));

endmodule
```

- סימולציה עבור תקינות הטופTop:



ניתן לראות שמודול הטופ פועל, כאשר הוא במצב פעולה רציפה, סיבוב עם כיוון השעון, במהירות 10 סיבובים לדקה. ניתן לראות את הפולסים ששולח מודול הקאונטר לתוך מכונת המצבים שאחראית על התנועה, וזו משנה את המצג עבור כל פולס. כמו כן ניתן לראות את המהירות כפי שנשלחת לתצוגה בseven segment, שעומדת בבינארית על עשרה סיבובית לדקה.

כעת נציג את תתי היחידות המרכיבות את המערכת , וסימולציות שמראות את פעולתן:

מודול מספר 1: *Speed controller*

כניסות:

Resetb – איפוס המעגל , מיוצג על ידי ביט בודד שמתחלף בין 1 ל-0 בהתאם למצב הכפתור.
Speed button – הכפתור ששולט על המהירות.

יציאה :

out_speed: יציאה פנימית במעגל (רגיסטר בעל 3 ביטים) שמעבירה למודולי הקאונטר והצג את המהירות הנוכחית.

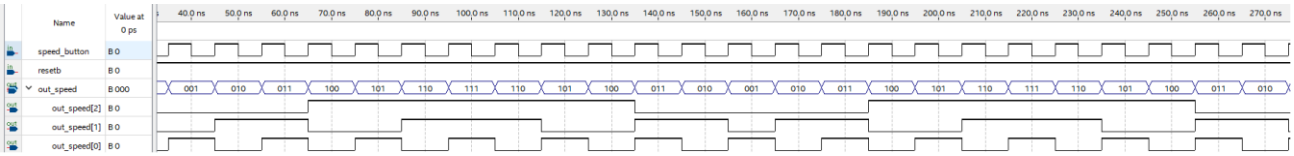
מימוש בקוד:

```
module speed_controller(resetb, speed_button, out_speed);

   //inputs for the speed conroller
   input wire resetb, speed_button;

   //this parameter will be an indicator for adding
   //speed or decreasing if needed. 1- add, 0 - decrease.
   reg speed_up;

   //the current speed will be defined by number
   output reg [2:0] out_speed;

   parameter ten_laps = 3'b001,
             laps_50 = 3'b101,
             laps_20 = 3'b010;


   always @(negedge speed_button or negedge resetb)
   begin
      if (~resetb)
      begin
         out_speed <= 3'b001;
         speed_up <= 1'b1;
      end
      else
      begin
         if (~speed_button && speed_up) //pressing the speed button will increase speed
         begin
            out_speed <= out_speed + ten_laps;
            if (out_speed == laps_50 )
               speed_up <= 1'b0; //we got to 60 laps per minute
         end

         else if (~speed_button && ~speed_up) //pressing the speed button will now decrease speed
         begin
            out_speed <= out_speed - ten_laps;
            if (out_speed == laps_20)
               speed_up <= 1'b1; //we got to 10 laps per minute
         end
      end
   end
endmodule
```

אופן הפעולה: מדובר במכונת מצבים , שמשנה את המהירות הסיבובים של המנוע. מתחילים ממצב התחלתי עם מהירות של עשרה סיבובים לדקה, ובכל לחיצת כפתור המהירות עולה בעוד עשרה סיבובים לדקה , עד שמגיעים לשישים סיבובים לדקה. כאשר מגיעים ל60 סיבובים לדקה, ישנו דגל *speed up* שמשתנה לאפס, וגורם לכל לחיצת כפתור לאחר מכן להוריד את המהירות בעשרה סיבובים לדקה. כאשר מגיעים שוב לעשרה סיבובים לדקה הדגל הנ"ל נדלק ולאחר מכן כל לחיצה תעלה את המהירות שוב. המודול מוציא רגיסטר באורך שלושה ביטים שמייצג את המהירות ומעביר אותו למודול ה– *seven segment* ומודול הקאונטר לצורך המשך פעולת המנוע ותצוגת המהירות על הלוח.

סימולציה:

נדמה לחיצות רבות על כפתור המהירות באמצעות הכנסת קלט של שעון במקום לחיצה פיזית (ניתן גם לדמות לחיצה פיזית ידנית עד ידי בחירת האות בצורה ידנית) , ונצפה לראות במוצא המייצג את ספרת העשרות של המהירות עולה ויורד בהתאמה.



ניתן לראות כי המהירות מתחילה מ10 סיבובים לדקה, מצב המיוצג במוצא על ידי המספר 001 (1 בבינארית) עבור כל לחיצה המהירות עולה , וכאשר מגיעים למהירות המקסימלית של 60 סיבובים לדקה, המיוצגת על ידי המספר 110 (6 בבינארית) המהירות מפסיקה לעבוד, ובלחיצה הבאה המהירות מתחילה לרדת, עד שהיא מגיעה לערך המינימלי שוב ושוב עולה בלחיצה לאחר מכן. מכונת המצבים הנ"ל היא אסינכרונית ,כלומר לא תלויה בשעון.

בלוק מספר 2 $Seven\ Segment$:

כניסות:

$Resetb$ – איפוס המעגל , מיוצג על ידי ביט בודד שמתחלף בין 1 ל0 בהתאם למצב הכפתור.

cur_speed - אות כניסה אשר מגיעה ממודול ה$speed_controller$, בעלת שלושה ביטים המייצג את המהירות הנוכחית אותה צריך להציג על הצג.

יציאות:

$HEX0, HEX1$: רגיסטרי שבעה ביטים שיוצאים ל$FPGA$ על מנת להראות את הספרות על התצוגה שנמצאת על הלוח.

מימוש בקוד:

```
module seven_segment_speed(resetb, cur_speed, HEX0, HEX1);

   input wire resetb;
   //the current speed we need to translate to 7-segment
   input [2:0] cur_speed;
   //the speed we will see on the display
   output [6:0] HEX0;
   output [6:0] HEX1;

   parameter laps_10 = 3'b001,
             laps_20 = 3'b010,
             laps_30 = 3'b011,
             laps_40 = 3'b100,
             laps_50 = 3'b101,
             laps_60 = 3'b110;

   assign HEX1 = (cur_speed == laps_10) ? 7'h79 :
                 (cur_speed == laps_20) ? 7'h24 :
                 (cur_speed == laps_30) ? 7'h30 :
                 (cur_speed == laps_40) ? 7'h19 :
                 (cur_speed == laps_50) ? 7'h12 :
                 (cur_speed == laps_60) ? 7'h02 :7'h40;
   assign HEX0 = 7'h40;
endmodule
```

אופן הפעולה: המודול האחראי להצגת המהירות על הלוח. גם כאן מדובר במכונת מצבים שמקבלת את המהירות ממודול

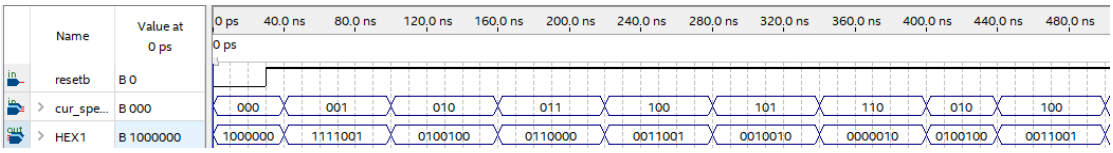
ה $speed\_contorller$ בתצורה של שלושה ביטים ובהתאם מוציא שני רגיסטרים באורך שבע ביטים המייצגים את המהירות , כאשר $HEX0$ מייצג את ספרת היחידות ומוגדר תמיד להיות אפס (מאחר ואנו עובדים עם מהירויות עגולות בלבד) , ו$HEX1$ מייצג את ספרת העשרות וערכו נע בין 1 ל6. את התוצאה ניתן לראות בתצוגה הנמצאת על כרטיס הFPGA.

סימולציה:

טבלת האמת אותה נצפה לקבל עבור התצוגה :

$digit$	1	2	3	4	5	6
$binary$	001	010	011	100	101	110
$Seven\ segment$	1111001	0100100	0110000	0011001	0010010	0000010

לטובת הסימולציה נדמה הכנסה של מהירויות שונות לתוך המודול מתוך מכונת המהירויות ונראה מהם האותות ביציאה שמועברים למודול הseven segment שנמצא על הכרטיס:



ניתן לראות כי עבור כל מהירות שונה בכניסה המיוצגת על ידי שלוש ספרות בינאריות, מופיעה אותה ספרה במוצא בתצורה שמתאימה לכניסות הבינאריות של מודול ה$seven\ segment$ הכלול בכרטיס. מודול ה$seven$ סגמנט הכלול בכרטיס שלנו הינו מסוג common anode , מאחר והוא נדלק כאשר המתח על הפין שמייצג את האות הוא אפס (מוארק) .

מאחר והערכים שקיבלנו במוצא תואמים לאלה שמופיעים בטבלה בתור הערכים שהולכים לכניסות של הseven segment המודול עובד כמצופה.

מודול 3: $Counter$

כניסות:

CLK -שעון המערכת $(50MHz)$
$Resetb$ – איפוס המעגל , מיוצג על ידי ביט בודד שמתחלף בין 1 ל0 בהתאם למצב הכפתור.
cur_speed - אות של שלושה ביטים המייצג את המהירות הנוכחית בבינארית, בה המנוע אמור להסתובב, והיא מגיעה ממודול ה$speed_controller$.
on_switch : מתג האחראי להפעיל את המעגל בתנועה רציפה.
$quarter\_button$ – כפתור המייצג הפעלה של רבע צעד , מיוצג על ידי ביט 1 ועובד רק כאשר on_switch נמצא במצב כבוי (0).
$step\_size\_sw$ – כפתור שאחראי על גודל הצעד, מיוצג על ידי ביט.

יציאות:

אינדיקטור: פולס היוצא מהמודול כאשר עברו מספר עליות השעון הרצוי, המתריע מתי יש לבצע צעד.
הפולס מועבר למודול ה$motion\ state\ machine$, מיוצג על ידי ביט 1.

מימוש בקוד:

המודול מחולק לשני חלקים, חלק שאחראי על תנועה רציפה וחלק שאחראי על תנועה בדידה, או בשם אחר רבע סיבוב.

```
module counter(clk, resetb, cur_speed, on_switch, step_size_sw, quarter_button, indicator);

   input wire clk, resetb, on_switch, step_size_sw, quarter_button;
   reg [26:0] count;
   wire [26:0] clock_cycles;

   reg [26:0] current_cycles; //our current clock cycles number

   input [2:0] cur_speed;

   reg [6:0] step_counter; //counts the number of steps we did
   reg finished_quarter; //indicator which tells us if we finished a quarter
   output reg indicator; //our pulse when we make 1 full/half step
                         //of a circle
   reg [26:0] cycles_counter; //sums the cycles we passed so far

   reg log_press_reg;
   wire long_press_wire;

   parameter laps_10 = 3'b001,
             laps_20 = 3'b010,
             laps_30 = 3'b011,
             laps_40 = 3'b100,
             laps_50 = 3'b101,
             laps_60 = 3'b110,
             full_step = 1'b0,
             half_step = 1'b1;

   //define the number of cycles needed according to the current speed/step size mode
   assign clock_cycles = (cur_speed == laps_10 && step_size_sw == full_step) ? 27'b001_0110_1110_0011_0110_0000://1,500,000
                         (cur_speed == laps_20 && step_size_sw == full_step) ? 27'b000_1011_0111_0001_1011_0000://  750,000
                         (cur_speed == laps_30 && step_size_sw == full_step) ? 27'b000_0111_1010_0001_0010_0000://  500,000
                         (cur_speed == laps_40 && step_size_sw == full_step) ? 27'b000_0101_1011_1000_1101_1000://  250,000
                         (cur_speed == laps_50 && step_size_sw == full_step) ? 27'b000_0100_1001_0011_1110_0000://  375,000
                         (cur_speed == laps_60 && step_size_sw == full_step) ? 27'b000_0011_1101_0000_1001_0000://  250,000

                         (cur_speed == laps_10 && step_size_sw == half_step) ? 27'b000_1011_0111_0001_1011_0000://  750,000
                         (cur_speed == laps_20 && step_size_sw == half_step) ? 27'b000_0101_1011_1000_1101_1000://  375,000
                         (cur_speed == laps_30 && step_size_sw == half_step) ? 27'b000_0011_1101_0000_1001_0000://  250,000
                         (cur_speed == laps_40 && step_size_sw == half_step) ? 27'b000_0010_1101_1100_0110_1100://  187,500
                         (cur_speed == laps_50 && step_size_sw == half_step) ? 27'b000_0010_0100_1001_1111_0000://  150,000
                         (cur_speed == laps_60 && step_size_sw == half_step) ? 27'b000_0001_1110_1000_0100_1000://  125,000
                         27'b000_0000_0000_0000_0000_0000;

   //Making sure that long press doesnt continuing to count after quarter turn
   always @ (posedge clk or negedge resetb) begin
      if(~resetb)
         log_press_reg <= 1'b0;

      else
         log_press_reg <= ~quarter_button;
   end

   assign long_press_wire = (~quarter_button & ((~quarter_button)^log_press_reg));

```

החלק שסופר תנועה רציפה:

```
   //continuous mode
   always @(posedge clk or negedge resetb)
   begin
      if(~resetb)
      begin
         current_cycles <= 27'b000_0000_0000_0000_0000_0000;
         indicator <= 1'b0;
         finished_quarter <= 1'b0;

      end

      else if (on_switch)
      begin
         if (current_cycles ==clock_cycles) //if we got to the clock cycles for the current mode, our indicator turns to logic 1
         begin
            indicator <= 1'b1; //pulse for making a step
            current_cycles <= 27'b000_0000_0000_0000_0000_0000;
         end
         else
         begin
            current_cycles <= current_cycles + 27'b000_0000_0000_0000_0000_0001;
            indicator<=1'b0;
         end
      end

```

החלק שאחראי על רבע הצעד:

```
86          //QUARTER MODE
87
88          else if (~on_switch && long_press_wire)
89          begin
90            finished_quarter <= 1'b1;
91          end
92
93          else if (finished_quarter && clk)
94          begin
95            if (step_size_sw == full_step) //FULL STEP
96            begin
97
98              if (cycles_counter == clock_cycles)//our indication for knowing when we got to the correct cycles number
99              begin
100               indicator <= 1'b1; //pulse for making a step
101               cycles_counter <= 27'b0;
102               step_counter <= step_counter + 7'b1;
103               if (step_counter == 7'b011_0010) //if we passed 50 steps it means we finished a quarter
104               begin
105                 finished_quarter <= 1'b0;
106                 step_counter <= 7'b0;
107                 indicator <= 1'b0;
108               end
109             end
110             else
111             begin
112               cycles_counter <= cycles_counter + 27'b1;
113               indicator <= 1'b0;
114             end
115           end
116
117
118
119           else if (step_size_sw == half_step) //HALF STEP
120           begin
121
122             if (cycles_counter == clock_cycles)//our indication for knowing when we got to the correct cycles number
123             begin
124               indicator <= 1'b1; //pulse for making a step
125               cycles_counter <= 27'b0;
126               step_counter <= step_counter + 7'b1;
127               if (step_counter == 7'b110_0100) //if we passed 50 steps it means we finished a quarter
128               begin
129                 finished_quarter <= 1'b0;
130                 step_counter <= 7'b0;
131                 indicator <= 1'b0;
132               end
133             end
134             else
135             begin
136               cycles_counter <= cycles_counter + 27'b000_0000_0000_0000_0000_0000_0001;
137               indicator <= 1'b0;
138             end
139           end
140         end
141
142
143
144         else
145         begin
146           indicator <= 1'b0;
147         end
148       end
149     endmodule
```

<u>אופן הפעולה:</u>

תפקיד מודול זה הוא הוצאת פולסים למערכת אשר מתריעים על ביצוע צעדים. המודול מחולק לשני תתי תפקידים- תנועה רציפה של המנוע וביצוע רבע סיבוב (לפי בחירת המשתמש כמובן).

לאחר הגדרת הכניסות והיציאות אותן תיארנו למעלה, אנו רוצים שעבור כל מצב יהיה מספר מסוים של עליות שעון השמור ברגיסטר CLOCK_CYCLES אשר יגרום להוצאת הפולס כאשר נגיע אליו. כל מצב כזה תלוי במהירות ובגודל הצעד אשר נכנסים למודול ככניסות. כאשר כפתור ה- RESET מופעל אנו מאפסים את ספירת עליות השעון וכמו כן גם את הפולס (אינדיקטור).

עבור המצב בו נרצה להפעיל את המנוע באופן רציף, קיים תנאי שרק אם המפסק המתאים הופעל ON_SWITCH ניכנס למצב זה ונתחיל בספרית עליות השעון באמצעות הרגיסטר CURRENT_CYCLES. כאשר נגיע למספר הרצוי של עליות השעון, שהוא CLOCK_CYCLES נוציא פולס ונאפס את מונה עליות השעון, ונמשיך כך עד אשר המפסק יכבה.

במידה והמפסק לא מופעל ובחרנו ללחוץ על הכפתור שמבצע רבע סיבוב QUARTER_BUTTON, הדגל FINISHED_QUARTER יוגדר להיות 1 ונתחיל בפעולה. פולסים יצאו כפי שתואר בפעולה הרציפה רק שהפעם נרצה לספור גם את מספר הצעדים אותם ביצענו. כאשר המונה STEP_COUNTER יגיע ל-50 במקרה של צעד מלא או 100 במקרה של חצי צעד, נדע שסיימנו רבע סיבוב, נאפס את המונה ונגדיר את הדגל להיות 0 וכך פעולת רבע הסיבוב תיעצר. רגיסטר דגל מסייע לנו בדרישה שהמנוע לא ימשיך להסתובב לאחר שביצע רבע סיבוב גם אם נלחץ על הכפתור באופן רציף.

במידה ומה שתואר לעיל לא קורה, האינדיקטור יהיה אפס ולא יצאו פולסים.

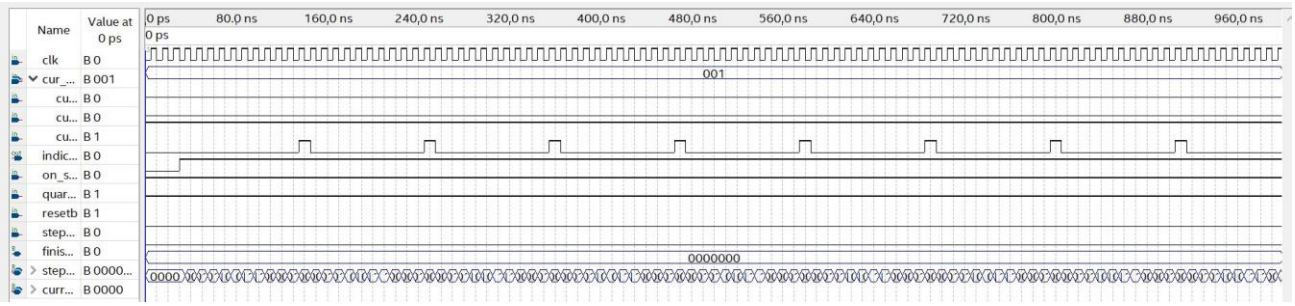
את חישוב מספר עליות השעון מחשבים ע"י הנוסחא הבאה:

```
128             begin
129               cycles_counter <= cycles_counter + 27'b000_0000_0000_0000_0000_0000_0001;
130               indicator <= 1'b0;
131             end
132           end
133         end
134
135         else
136         begin
137           indicator <= 1'b0;
138         end
139       end
140     endmodule
```

$$CLOCK\ CYCLES\ = \frac{50MHz\ [clock] \cdot 60[secondes\ per\ minute]}{(rounds\ per\ minute) \cdot \frac{360}{1.8}[motor\ step\ size] \cdot (step\ type)}$$

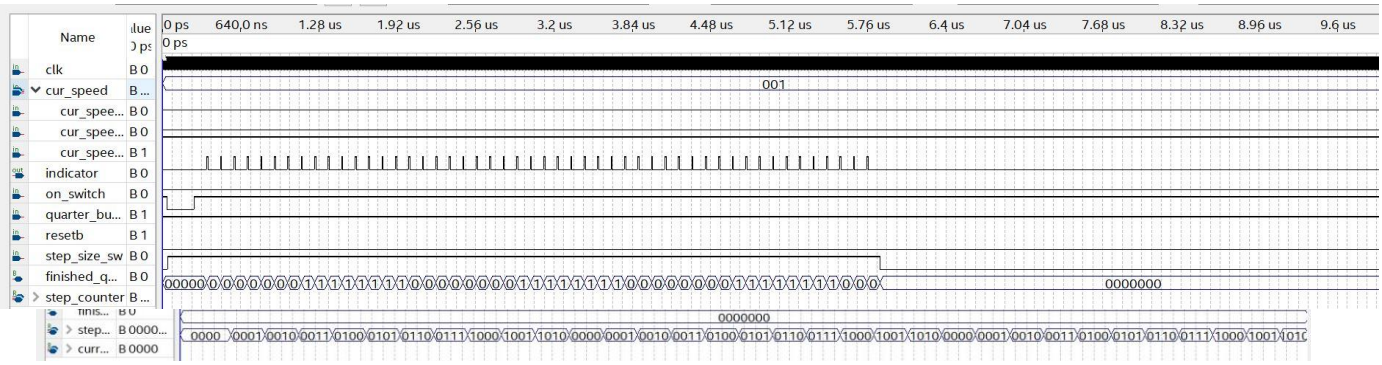
סימולציה:

עבור תנועה רציפה:



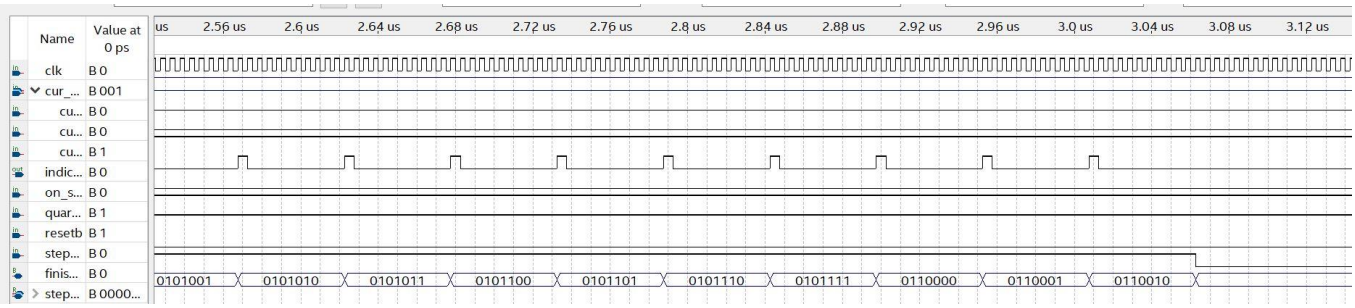
ממבט קרוב יותר:

בסימולציה של הפעולה הרציפה הגדרנו את הפולס לצאת כל 10 עליות שעון, ואכן ניתן לראות כי כאשר מונה עליות השעון (בשורה התחתונה) מגיע ל-10 עליות שעון האינדיקטור הופך ל-1 והמונה מתאפס. מאחר והפעולה עובדת כנדרש עבור 10 עליות שעון היא תעבוד גם עבור מספר עליות השעון בפועל. ניתן לראות גם שמונה עליות השעון לא פועל כאשר הסוויץ לא מופעל.



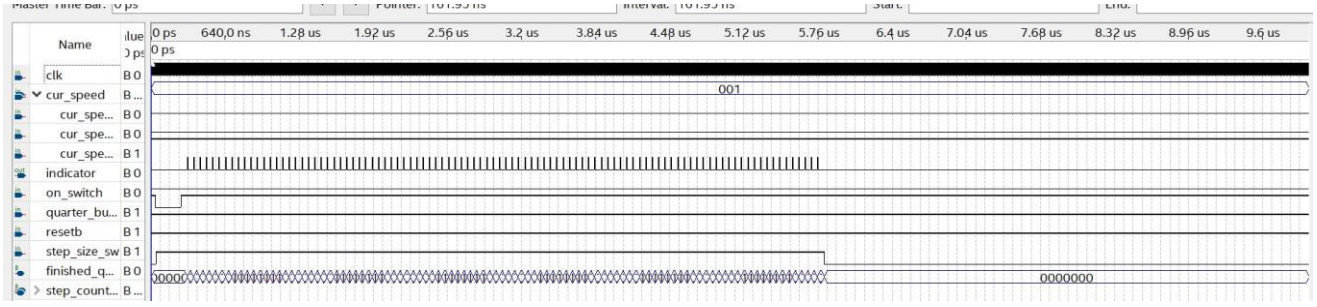
עבור רבע סיבוב:

ממבט קרוב יותר:

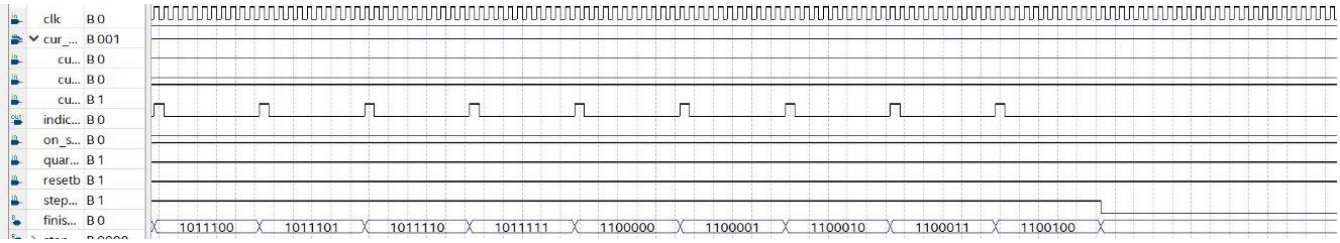


זוהי הסימלוציה עבור פעולת הרבע סיבוב כאשר אנו במצב של צעד מלא. אנו שמים לב להוצאת פולסים לאחר לחיצה בודדת על כפתור הרבע סיבוב וסוויץ הפעולה הרציפה כבוי. האינדיקטור עולה כל 10 עליות שעון כמו בסימולציה של הפעולה הרציפה רק שכאן הוספנו רגיסטר המונה את מספר הצעדים שעברנו וכאשר מגיע ל-50 צעדים, כפי שניתן לראות בסימולציה, אינדיקטור הדגל מתאפס ואנו מפסיקים להוציא פולסים.

סימלוצית רבע סיבוב כאשר אנו במצב של חצי צעד:



ממבט קרוב:



גם פה אנו מזהים פעולה תקינה של הכפתור. לאחר לחיצה בודדת אנו מתחילים להוציא פולסים כל 10 עליות שעון ותוך כדי לספור את מספר הצעדים שעברנו. כאשר אנו מגיעים ל-100 צעדים אנו מאפסים את גריסטר הדגל ומפסיקים את הפעולה.

מודול 4: *Motion State Machine*

כניסות:

CLK -שעון המערכת $(50MHz)$
$Resetb$ – איפוס המעגל , מיוצג על ידי ביט בודד שמתחלף בין 1 ל0 בהתאם למצב הכפתור.
$Rotation\ direction$ - מיוצג על ידי ביט בודד שמתחלף בין 1 ל0 בהתאם למצב הכפתור.
$indicator$: אות המסמן למכונה מתי להוציא אות יציאה לכיוון המנוע, מגיע ממודול הקאונטר או ממודול הקוורטר.
$step\_size\_sw$ – כפתור שאחראי על גודל הצעד, מיוצג על ידי ביט כאשר 0 מסמן צעד מלא ו1 חצי צעד.

יציאות:

Current state: יציאה בעלת ארבעה ביטים שמייצגת את המצב והאופן שבו המנוע אמור להסתובב, יוצא אל הכרטיס ומשם דרך הדרייבר אל המנוע.

מימוש בקוד:

```
module motion_state_machine(clk,resetb,rotation_direction, step_size_sw, current_state, indicator);

    input wire resetb;              // key
    input wire clk;                 // clk

    input wire rotation_direction;  // key

    input wire step_size_sw;        // key

    input wire indicator;

    output reg[3:0] current_state;

    reg [3:0]next_state;

            //rotation states
    parameter idle    = 4'b0000,

            fstep_1 = 4'b1000,
            fstep_2 = 4'b0010,
            fstep_3 = 4'b0100,
            fstep_4 = 4'b0001,

            hstep_1 = 4'b1010,
            hstep_2 = 4'b0110,
            hstep_3 = 4'b0101,
            hstep_4 = 4'b1001,

            // Roatation directions
            clockwise = 1'b0,
            counterclockwise = 1'b1,

            //step size
            full_step = 1'b0,
            half_step = 1'b1;


       // Rotation conditions.
    always @(posedge clk or negedge resetb)
        begin
                if (~resetb)
                  //If reset is pressed
                    next_state = idle;


               else if(indicator)
                 begin

                 current_state = next_state;
                 //Full step clockwise direction
                 if(rotation_direction == clockwise && step_size_sw == full_step)

                       case (current_state)
                       idle:
                             next_state <= fstep_1;
                       fstep_1:
                             next_state <= fstep_2;
                       fstep_2:
                             next_state <= fstep_3;
                       fstep_3:
                             next_state <= fstep_4;
                       fstep_4:
                             next_state <= fstep_1;
                       default :
                             next_state <= idle;
                       endcase

                  //Full step counterclockwise direction
                  else if(rotation_direction == counterclockwise && step_size_sw == full_step)

                        case (current_state)
                        idle:
                              next_state <= fstep_1;
                        fstep_1:
                              next_state <= fstep_4;
                        fstep_4:
                              next_state <= fstep_3;
                        fstep_3:
                              next_state <= fstep_2;
                        fstep_2:
                              next_state <= fstep_1;
                        default :
                              next_state <= idle;
                        endcase

                  //halt step clockwise direction
                  else if(rotation_direction == clockwise && step_size_sw == half_step)

                        case (current_state)
                        idle:
                              next_state <= fstep_1;
                        fstep_1:
                              next_state <= hstep_1;
                        hstep_1:
                              next_state <= fstep_2;
                        fstep_2:
                              next_state <= hstep_2;
                        hstep_2:
                              next_state <= fstep_3;
                        fstep_3:
                              next_state <= hstep_3;
                        hstep_3:
                              next_state <= fstep_4;
                        fstep_4:
                              next_state <= hstep_4;
                        hstep_4:
                              next_state <= fstep_1;
                        default :
                              next_state <= idle;
                        endcase

                  //halt step counterclockwise direction
                  else if(rotation_direction == counterclockwise && step_size_sw == half_step)

                        case (current_state)
                        idle:
                              next_state <= fstep_1;
                        fstep_1:
                              next_state <= hstep_4;
                        hstep_4:
                              next_state <= fstep_4;
                        fstep_4:
                              next_state <= hstep_3;
                        hstep_3:
                              next_state <= fstep_3;
                        fstep_3:
                              next_state <= hstep_2;
                        hstep_2:
                              next_state <= fstep_2;
                        fstep_2:
                              next_state <= hstep_1;
                        hstep_1:
                              next_state <= fstep_1;
                        default :
                              next_state <= idle;
                        endcase

                 end
              else
                current_state = current_state; // enable to the motor to finnish step
              end
        //end cases

endmodule
```

אופן הפעולה:

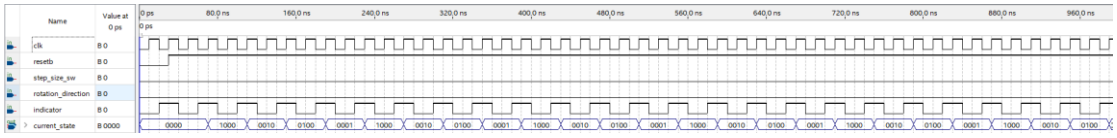
מגדירים למכונת מצבים את כל המצבים האפשריים לתנועה מבחינת כיוון, גודל צעד , מגדירים שבו לחיצה על כפתור הreset תגרום למכונה לחזור למצב סרק ($idle$) ולא לפעול, ולאחר מכן מגדירים למכונה כיצד להזיז את המנוע בהתאם להגדרות.

כאשר מגיע אות פולס (מופיע כאינדיקטור בקוד) ממודול הקאונטר , המכונה בודקת לאיזה כיוון עליה להסתובב ובאיזה גודל צעד, ומקצה את אות היציאה המתאים לתוך הרגיסטר $next\ state$, ולאחר שסיימה לבדוק את כל התנאים ולמצוא את המצב המתאים, היא מוציאה אותו מ$next\ state$ לתוך $current\ state$, היציאה של המעגל. אות היציאה הינו רגיסטר באורך ארבעה ביטים, שעובר אל הכרטיס ומשם אל המנוע. המנוע מסתובב כי המכונה ממשיכה להעביר לו מצב אחר מצב שהוגדר מראש עבור כל סוג תנועה אפשרי.

סימולציות:

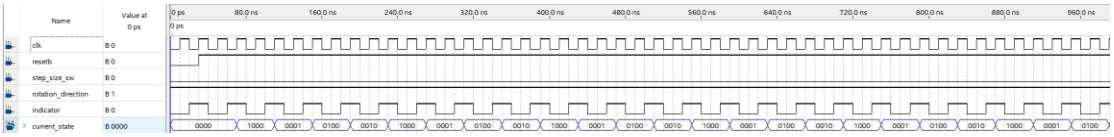
הדגמת תקינות פעולת מכונת המצבים, כאשר מראים את ארבעת המצבים המשתנים האפשריים – תנועה עם ונגד כיוון השעון, חצי צעד וצעד מלא.

א. עבור צעד מלא בכיוון השעון :



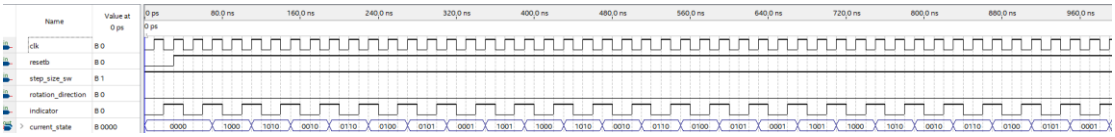
ניתן לראות את המעבר בין ארבעת המצבים $1000 \rightarrow 0001 \rightarrow 0100 \rightarrow 0010 \rightarrow 1000$ בהתאם לציפיות שלנו ובהתאם לטבלת האמת בחלק הראשון של הדוח כאשר המכונה מתחילה ממצב ברירת מחדל של $idle$.

ב. עבור צעד מלא נגד כיוון השעון :



ניתן לראות את המעבר ההפוך בין המצבים $1000 \rightarrow 0010 \rightarrow 0100 \rightarrow 0001 \rightarrow 1000$ בהתאם לציפיות שלנו ובהתאם לטבלת האמת בחלק הראשון של הדוח.

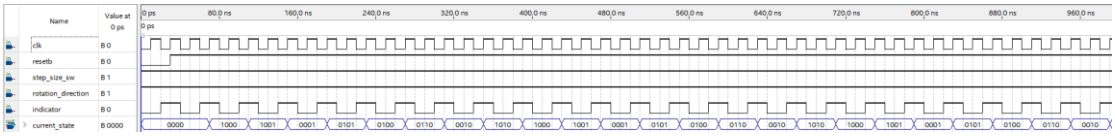
ג. עבור חצי צעד בכיוון השעון:



כעת ,בניגוד לסעיף הקודם ניתן להבחין בשמונה מצבים שונים , וניתן לראות כי הם חוזרים על עצמם בצורה מחזורית כפי שציפינו.

$$1000 \rightarrow 1010 \rightarrow 0010 \rightarrow 0110 \rightarrow 0100 \rightarrow 0101 \rightarrow 0001 \rightarrow 1001 \rightarrow 1000$$

ד. עבור חצי צעד נגד כיוון השעון:



גם עבור המצב הנ"ל ניתן לראות מעבר מחזורי בין שמונה מצבים כפי ששציפינו.

$$1000 \rightarrow 1001 \rightarrow 0001 \rightarrow 0101 \rightarrow 0100 \rightarrow 0110 \rightarrow 0010 \rightarrow 1010 \rightarrow 1000$$

לסיכום , בהתאם לטבלת האמת בחלק א' מכונת המצבים נעה בין המצבים על פי הסדר שהוגדר לה ובהתאם לגודל הצעד וכיוון הסיבוב, ולכן נסיק כי היא עובדת בצורה תקינה.

במעבדה זו מימשנו שליטה במנוע צעד , כאשר קבענו לו את כיוון הסיבוב, גודל המהירות בסיבובים לדקה, פעולה רציפה או בדידה, גודל הצעד , וכן מימשנו כפתור איפוס ותנועה רגעית.

מצורף קישור לצפייה בסרטון המדגים את פעול המנוע:
https://drive.google.com/file/d/1NwxbKDlQoBn6Jk_pFZ6RlmbNEn3EuyM0/view?fbclid=IwAR3YIvjUQtB7DnVRv8ZCO6UyajbzIacUjR6aRXobjp6A-A28n9o6t25c0TM

בהתחלה הפעלנו את המנוע על פעילות רציפה עם כיוון השעון על מהירות של 60 סיבובים לדקה במשך 60 שניות, לאחר מכן פעולה רציפה במהירות של 20 סיבובים לדקה נגד כיוון השעון וכך גם עבור מהירות של 60 סיבובים .