הקדמה:

המעבד שהתבקשנו לממש הוא מעבד מסוג multy cycle

כלומר כל פקודה מחולקת למספק שלבים כאשר כל שלב מתבצע במחזור שעון אחד.

המעבד שלנו יודע לבצא את הפקודות הבאות:

Mv – העתקת ערך מרגיסטר אחד לרגיסטר אחר.

Mvi – הכנסת ערך קבוע לרגיסטר המבוקש.

Add – חיבור של 2 רגיסטרים והכנסת ערך החיבור אחד מהם.

Sub - חיסור של 2 רגיסטרים והכנסת ערך החיסור אחד מהם.

Ones – ספירת כמות האחדות בערך הבינארי של ערך בתוך רגיסטר X והכנסת כמות האחדות לתוך רגיסטר Y.

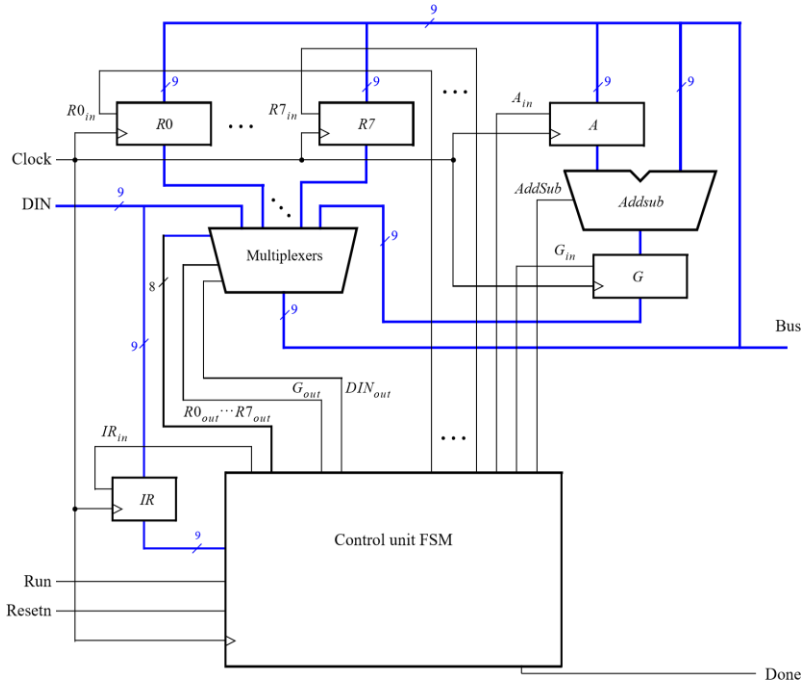
להלן פירוט מקור ויעד פעולות הרגיסטרים :

Operation	Function performed
mv *Rx,Ry*	$Rx \leftarrow [Ry]$
mvi *Rx,#D*	$Rx \leftarrow D$
add *Rx,Ry*	$Rx \leftarrow [Rx] + [Ry]$
sub Rx,Ry	$Rx \leftarrow [Rx] - [Ry]$
ones *Rx,Ry*	Rx , → [Ry]

חלוקת הפעולת למחזורי שעון :

	T_1	T_2	T_3
(mv): I_0	RY_{out}, RX_{in}, Done		
(mvi): I_1	DIN_{out}, RX_{in}, Done		
(add): I_2	RX_{out}, A_{in}	RY_{out}, G_{in} AddSub=01	G_{out}, RX_{in}, Done
(sub): I_3	RX_{out}, A_{in}	RY_{out}, G_{in}, AddSub=10	G_{out}, RX_{in}, §
I_4 (ones):	G_{in}, RX_{out}, AddSub=11	RY_{in}, Done	

להלן סכימת המעבד שאותו תכננו



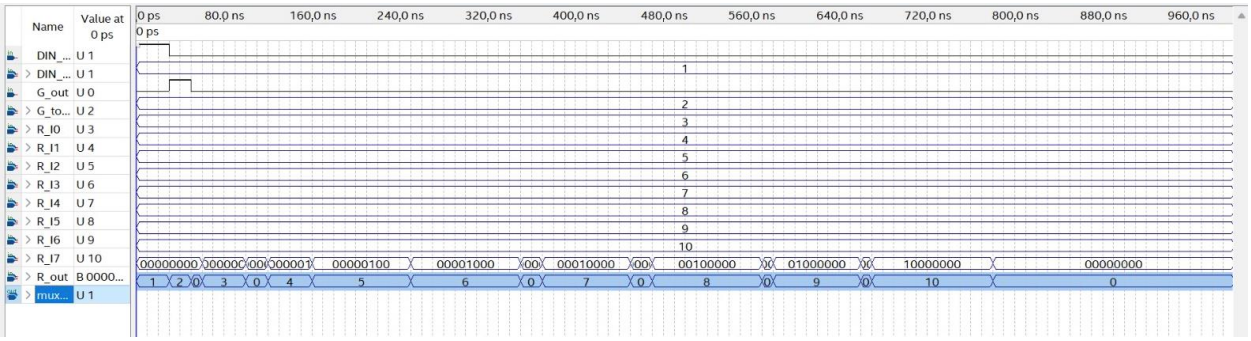
במימוש שלנו מימשנו את המוקס והמחבר בבלוקים שונים לצורך הנוחות.

נציין כי המימוש הודגם למנחה במעבדה .
לכן לא מצורף סרטון -באישורך

סימולציות על בלוקים משניים

Mux -test

בסימולציה זו אנו בודקים כי ה "mux"
מעביר לנו את החוטים הנכונים בהתאם לדרישותינו!
בדקנו עבור כל חיווט.
כמו"כ נציין כי ע"פ התכנון שלנו לא קיים מצב בו יש לנו מספר חוטים במקביל.



ALU-test

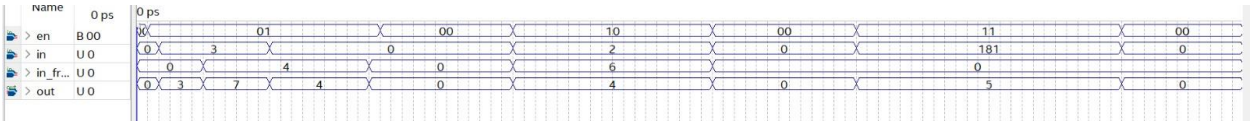
בסימולציה זו נבחן את תקינות ה- "ALU" נבדוק אם הוא מבצע כמו שצריך את הפעולות שהגדרנו לו :

00 - מחזיר אפסים

01 – חיבור

10 - חיסור

11 – פעולת ones



ניתן לראות מהסימולציה שכל הפעולות עובדות באופן תקין.

<u>הפעולה הראשונה חיבור</u>

נתנו ערכים לרגיסטר והכניסה השניה כך:

Reg{a}=,4 In =3

ונקבל ביציאה 7

בדקנו חיבור של הכניסה והרגיסטר עם 0 (כל אחד לחוד)

כאשר כל אחד מהם בעל ערך שונה התקבלה התוצאה כנדרש.

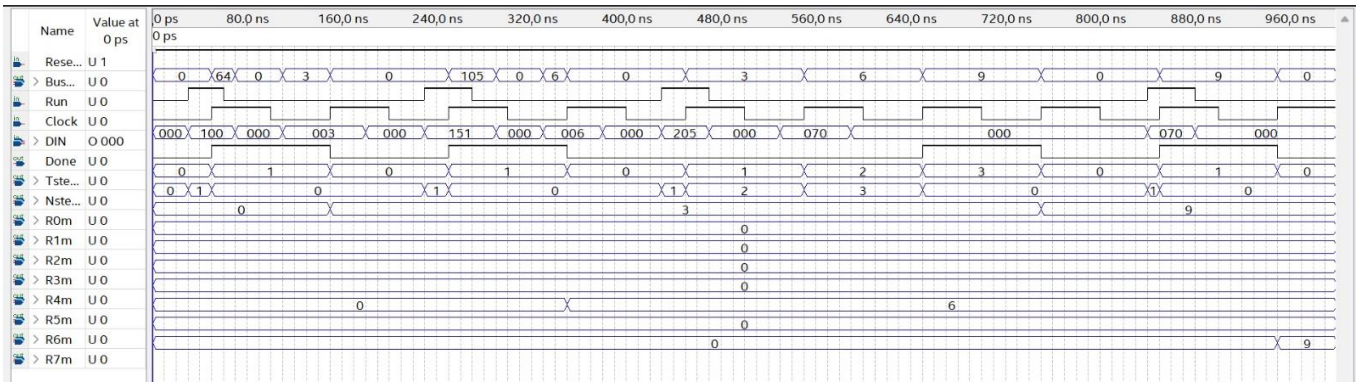
<u>פעולת חיסור</u>

Reg{a}=,6 In =2

ונקבל ביציאה 4=6-2

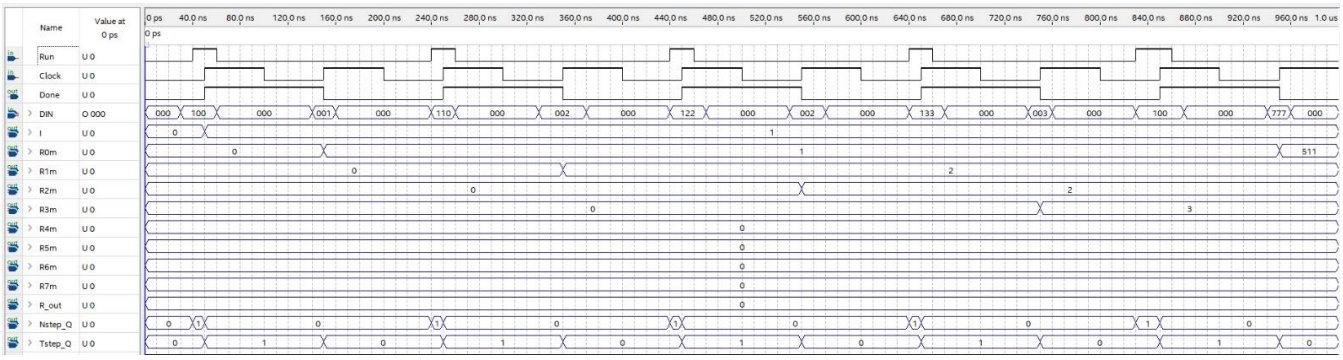
אח"כ בדקנו כי פעולת ספירת האפסים מתבצעת כמו שצריך על כניסה בעלת ערך 181 = 010110101 וכצפוי קיבלנו 5 ביציאה.

סימולציה כללית



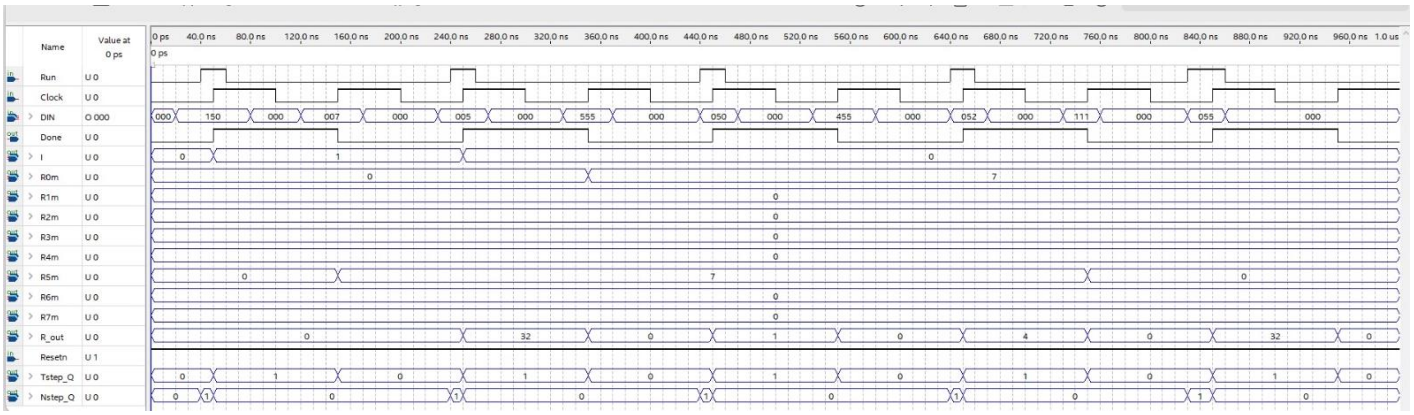
בדיקה כללית שכל הפעולות מתבצעות באופן תקין ומוציאות לנו ערכים תקינים
ניתן לראות כי מימשנו פעולות כמו : Mvi ,Add, Mv.
ניתן לראות כי Done עולה בזמן הנכון
הפעולה מתחילה כאשר run =1
ושה"Buswire"- מתעדכן כמו שצריך.
הסימולציה מציגה מספר פעולות כלליות,
בודקת האם הכנסת "סתם" מספרים בין לבין משפיע על התוכנית שלנו.
נדגיש כי בתחילת כל סימולציה ביצענו פעולת Mvi כדי למלא את הרגיסטר על מנת שנוכל לראות את השינויים המתבצעים במהלך הפקודות השונות.
בהמשך נפרט ביותר בפרטנות.

פעולת "mvi"



ביצוע פעולת העברה של קבוע בודקת שימוש באותו רגיסטר מספר פעמים (בהמשך נראה שימוש בריסט עבור פעולה זו)
נשים לב כי Done עולה בזמן הנכון ב -T1
והפעולה מתחילה כאשר run =1

פעולת "mv"



ביצוע פעולת העברה מרגיסטר לרגיסטר -בדיקת העברה מרגיסטר לעצמו, דחיפת פקודות לא רלוונטיות ,שימוש מספר פעמים באותו רגיסטר.

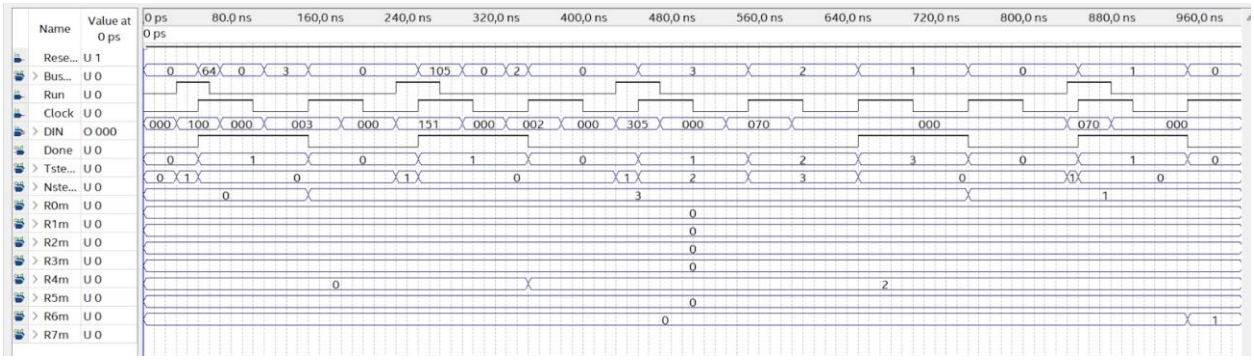
נשים לב כי Done עולה בזמן הנכון ב -T1

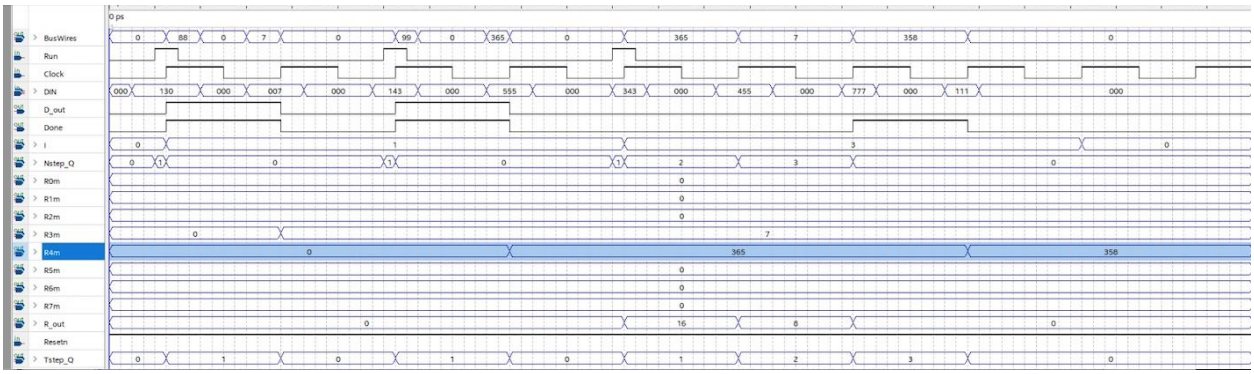
והפעולה מתחילה כאשר run =1

והפעולות מתרחשות בזמני המחזור שלהן.

פעולת חיסור

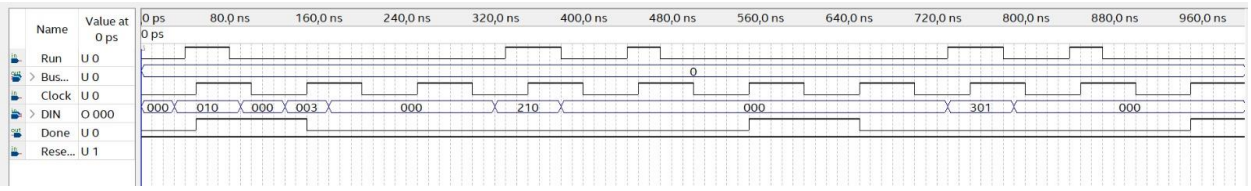
נשים לב כי Done עולה בזמן הנכון ב -T3
והפעולה מתחילה כאשר run =1
זמני מחזור נכונים .
ערכי התוצאות נכונים .
ניתן לראות כי ה- "Buswire" מתעדכן ע"פ פעולת המעבד כפי שציפינו.





ביצוע פעולת חיסור מרגיסטר לרגיסטר -בדיקה כללית, בדיקת חיסור מרגיסטר לעצמו(בהמשך), דחיפת פקודות לא רלוונטיות ,שימוש מספר פעמים באותו רגיסטר.

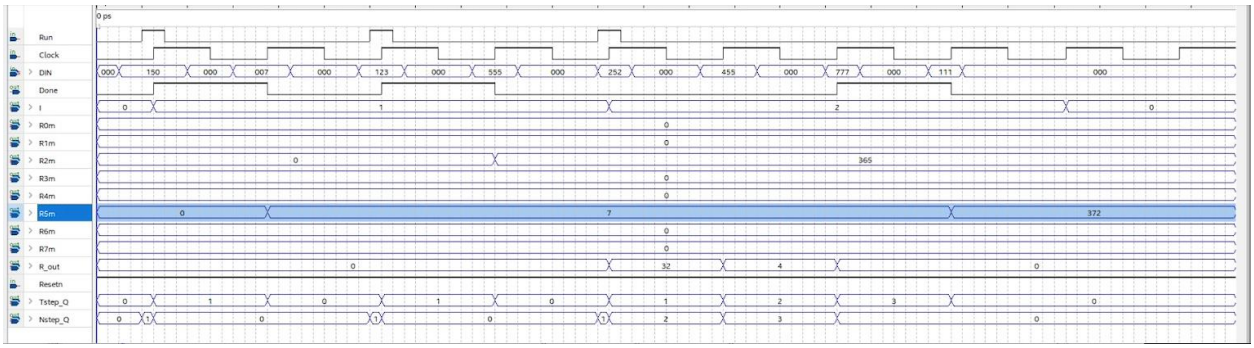
בדיקת run במהלך הרצה SUB

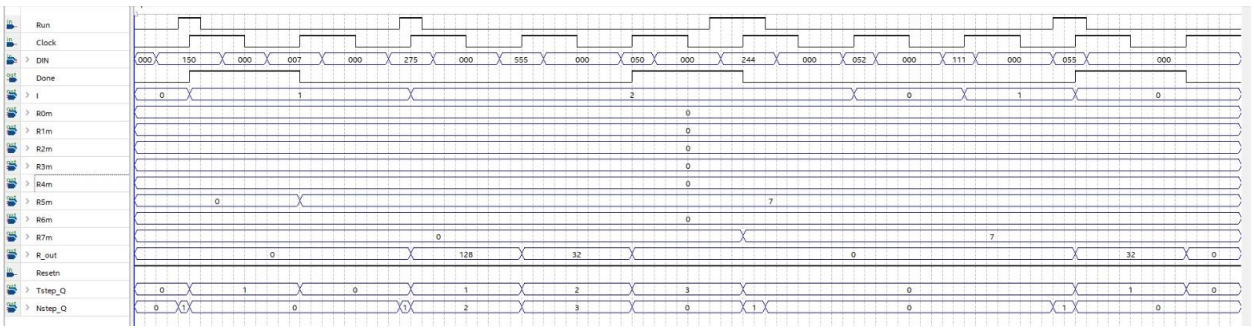


נוודא כי עליית "run" במהלך הרצת הפקודה לא תשפיע לנו על תוצאת הפעולה אותה ביקשנו לבצע

פעולת חיבור

ניתן לראות כי ה- "Buswire" מתעדכן ע"פ פעולת המעבד כפי שציפינו



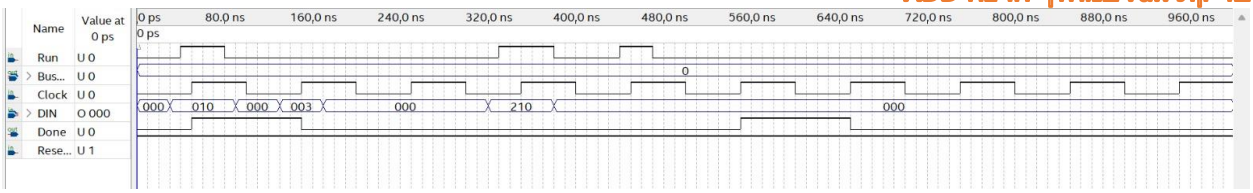


ביצוע פעולת חיבור מרגיסטר לרגיסטר

נבצע בדיקה כללית ונראה כי התוצאות שנקבל יהיו נכונות

בדיקת חיבור מרגיסטר לעצמו, דחיפת פקודות לא רלוונטיות ,שימוש מספר פעמים באותו רגיסטר.

בדיקת run במהלך הרצה ADD

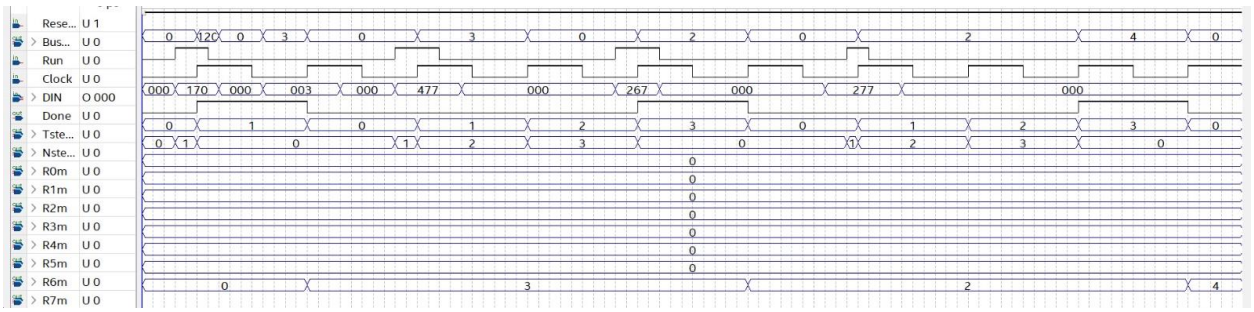


נוודא כי עליית "run" במהלך הרצת הפקודה לא תשפיע לנו על תוצאת הפעולה אותה ביקשנו לבצע.

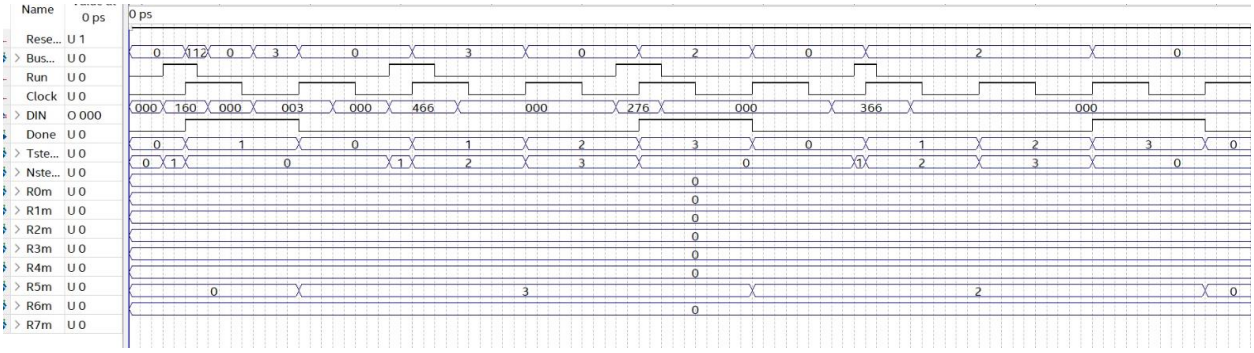
בדיקה פעולות על אותו רגיסטר

כעת נבדוק האם כאשר נבצע פעולות שונות על אותו רגיסטר ניתקל בבעיות מסויימות :

נבצע את כל הפעולות שקיימות

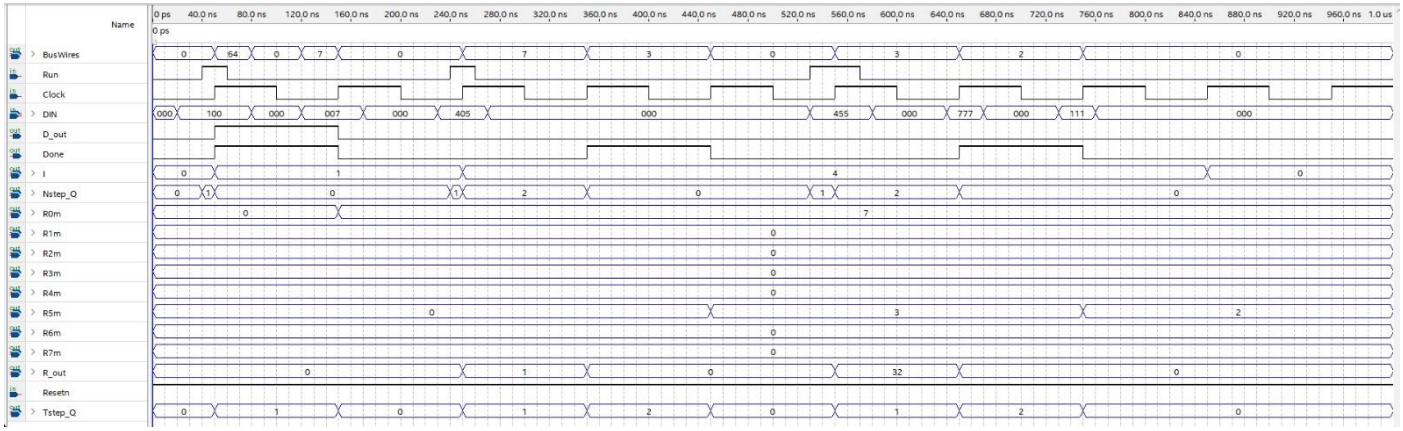


כוללת פעולת חיסור על רגיסטר שונה



סימולציה עבור פעולת ספירת כמות האחדים

ביצענו את פעולת ספירת ה 1 ע״י 2 שלבים (לא כולל זמן מחזור 0)
לא השתמשנו ברגיסטר A להכנסת הערך של רגיסטר X וזאת משום שחבל על זמן המחזור המיותר.
בסימולציה הכנסנו שימוש בסיסי בפקודה + פעולה על אותו הרגיסטר + הכנסת סתם ערכי פקודות.



אנו מימשנו את הפונקציה הנ״ל ע״י החישוב הבא:

out <= in [0] + in [1] + in [2] + in [3] + in [4] + in [5] + in [6] + in [7] + in [8];

כמו״כ ניתן לממש פעולה זו ע״י לולאת ״for״ ולא ע״י השמה כמו שאנחנו מימשנו.
עיקרון לולאת פור בורילוג הוא בעצם הכפלת החומרה כך שנשתמש במספר רכיבים ע״מ לממש את הלולאה.
כלומר כל אינדקס ממומש ברכיב אחר בחומרה, כך שהכל קורה בו זמנית!
פעולה זו היא מאוד יקרה לנו,לכן נשים לב כאשר אנו משתמשים בה!

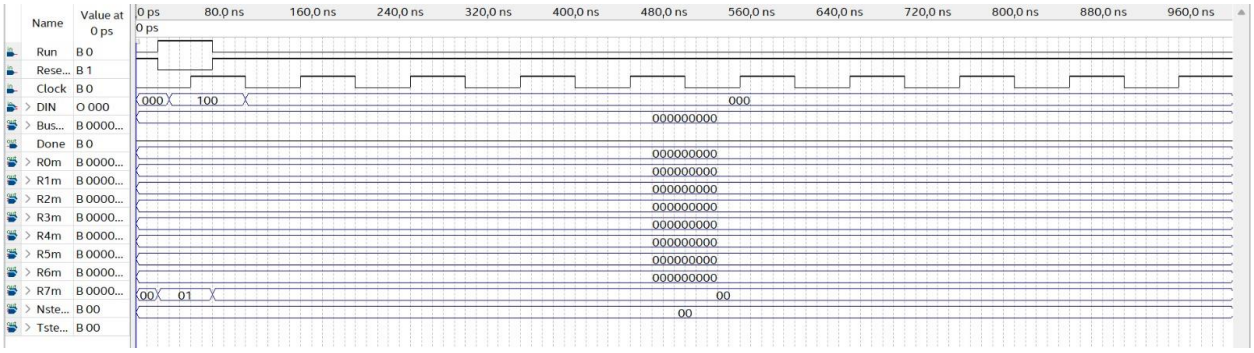
דוגמה למימוש ע״י לולאה:

```
output reg [8:0] out;
    out = 0;   //initialize count variable.
    for(i=0;i<8;i=i+1)    //check for all the bits.
        if(in[i] == 1'b1)     //check if the bit is '1'
           out = out + in[i];     //if its one, increment the count.
```

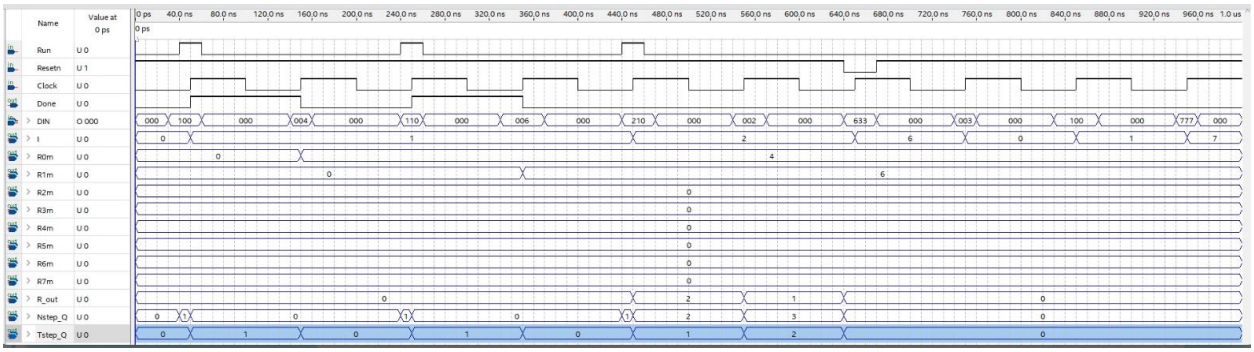
שימוש בריסט

נבדוק שביצוע פעולת ריסט מתבצעת כמו שצריך כך שתאפס לנו את זמני המחזור

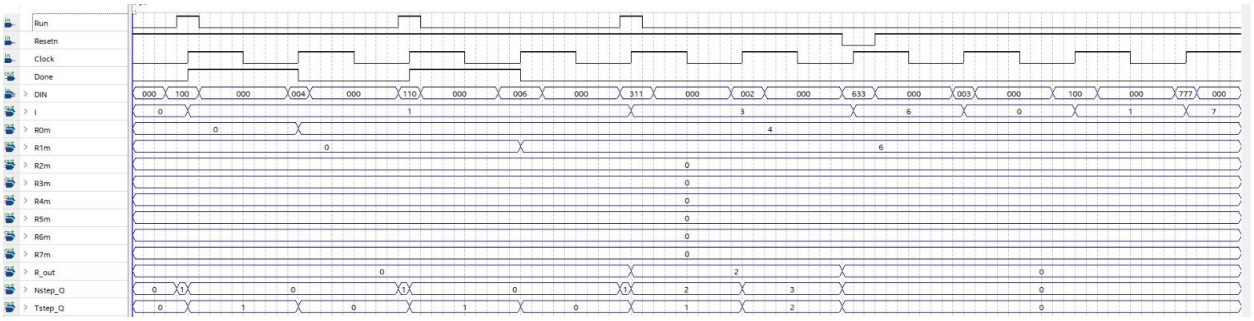
mvi



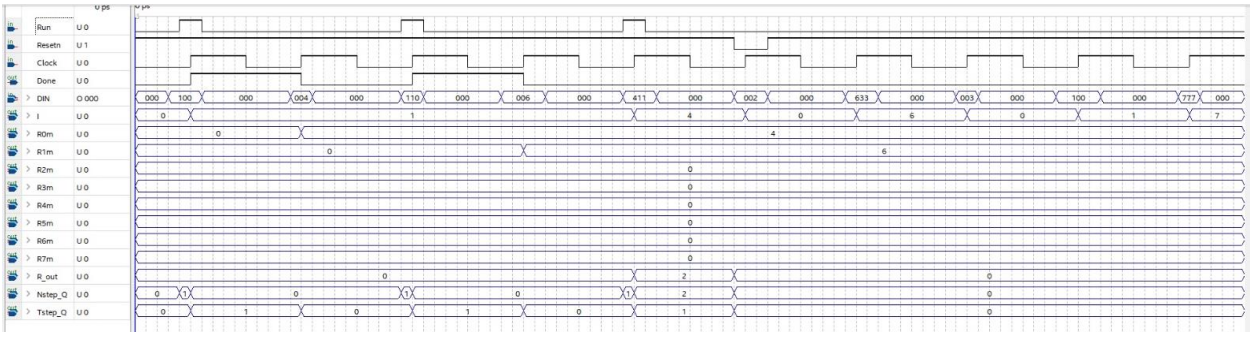
Add



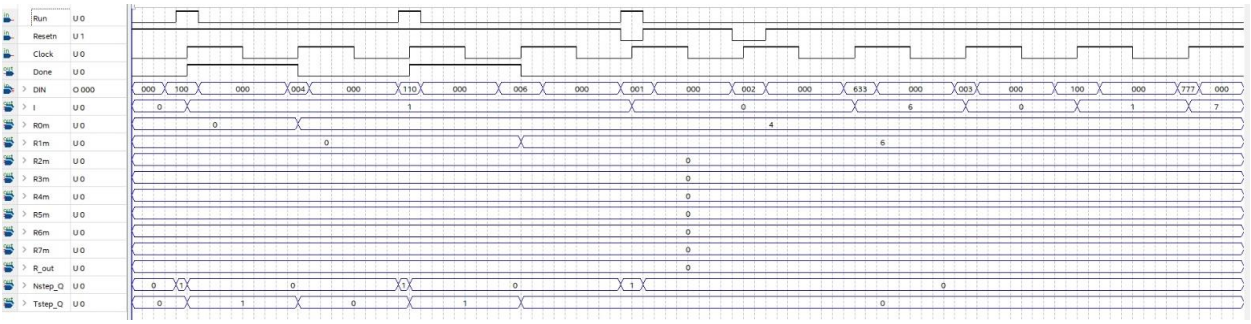
Sub



Ones



Mv



הרצת פקודות שלא קיימות :

נרצה לראות כי ערכי הרגיסטרים נשמרים ולא מתבצעות פעולות במעבד.

כמו״כ נצפה לא לראות את ״Done״ עולה.

נשים לב שמכונת המצבים רצה אך לא עושה כלום כמו שנצפה ע״פ איך שתכננו את המעבד.