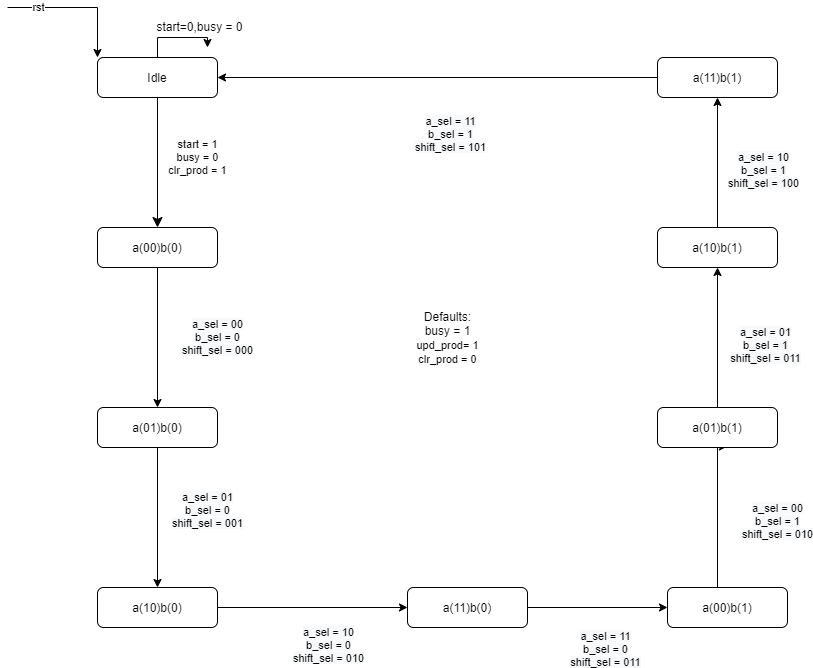
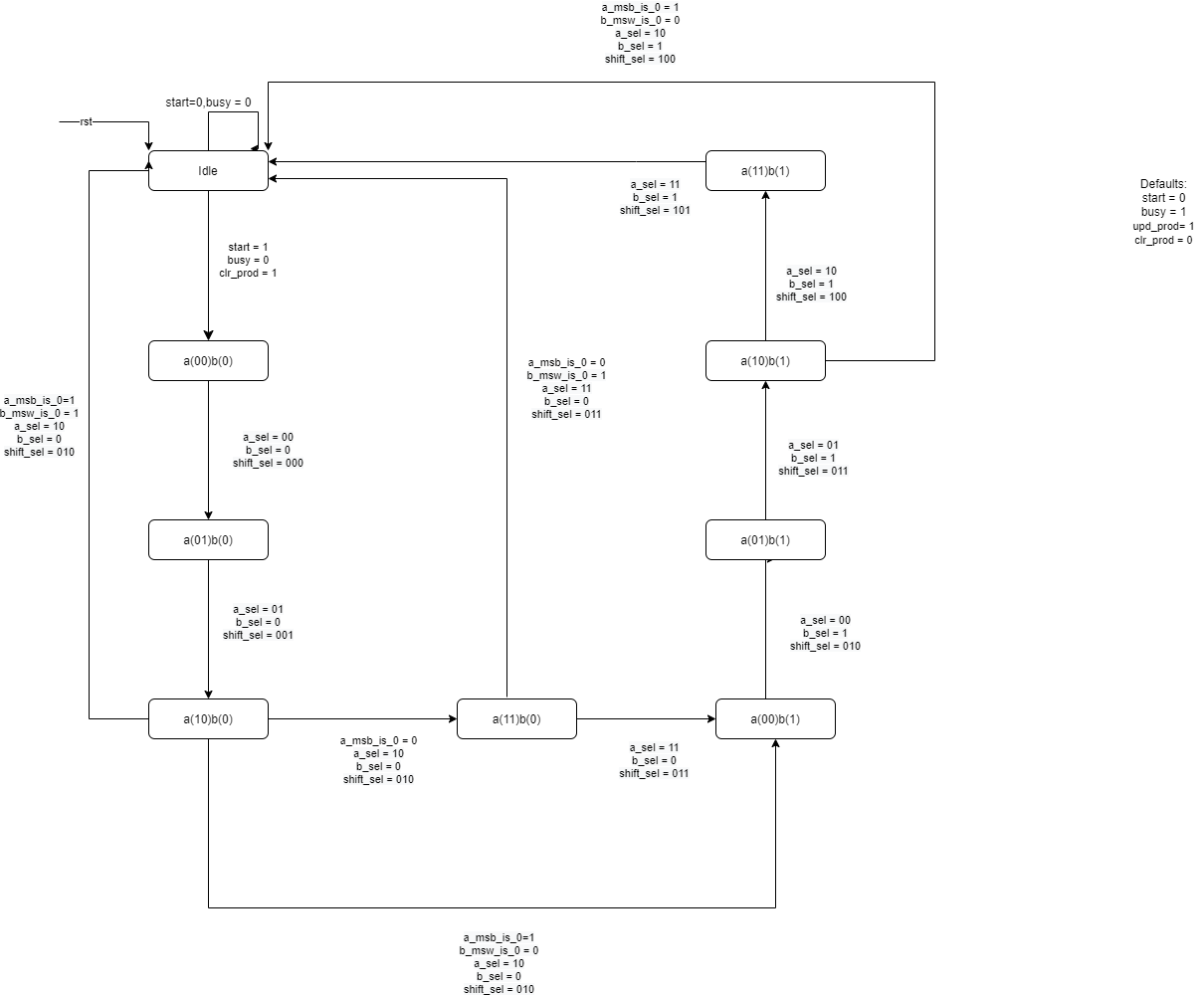
2.1:



פעולת הכפל לוקחת 9 מחזורי שעון, מרגע עליית הstart עד הרגע שבו busy יורד לאפס שזהו הרגע שבו product מעודכן בתוצאת הכפל המדויקת.

2.2:

כעת משך זמן פעולת הכפל תלוי בקלט:

* אם a\_msb\_is\_0 = 0 וגם b\_msw\_is\_0 = 0, המכונה תפעל באופן זהה לסעיף הקודם, והפעולה תיקח 9 מחזורי שעון, כולל המחזור שמתחיל את הפעולה ומדליק את start.
* אם a\_msb\_is\_0 = 1 וגם b\_msw\_is\_0 = 1, פעולת הכפל תיקח 4 מחזורי שעון כולל start.
* אם a\_msb\_is\_0 = 0 ו b\_msw\_is\_0 = 1, פעולת הכפל תיקח 5 מחזורי שעון כולל start.
* אם a\_msb\_is\_0 = 1 ו b\_msw\_is\_0 = 0, פעולת הכפל תיקח 7 מחזורי שעון כולל start.

2.3:

1. נתונים 2 מספרים a b באורך 8N סיביות כל אחד.

2. נגדיר משתנה result שישמור את תוצאת הכפל בינהם והוא יקבל ערך התחלתי result=0.

3. נחלק את a לN מספרים שונים באורך 8 סיביות כל אחד, נמספר אותם (a0, a1,a2,….,a(N-1))- שמונה הסיביות הראשונות (מימין לשמאל( ישמרו בa0, השמונה הבאות ישמרו בa1 וכן הלאה..

4. נחלק את b לN/2 מספרים שונים באורך 16 סיביות כל אחד, נמספר אותם (b0,b1,b2,…,b(N/2-1))

5. נגדיר משתנה זמני mul (משתנה שישמור פעולת כפל בודדת)

6. נגדיר 2 אינדקסים רצים i=0 וj=0.

7. כל עוד i<N:

7.1 כל עוד j<N/2:

7.1.1 באמצעות פקודת כפל 16x8: mul= a(i)\*b(j)

7.1.2 נוסיף לmul (8i+16j)- אפסים מימין (shiftleft) ונשמור את התוצאה בmul.

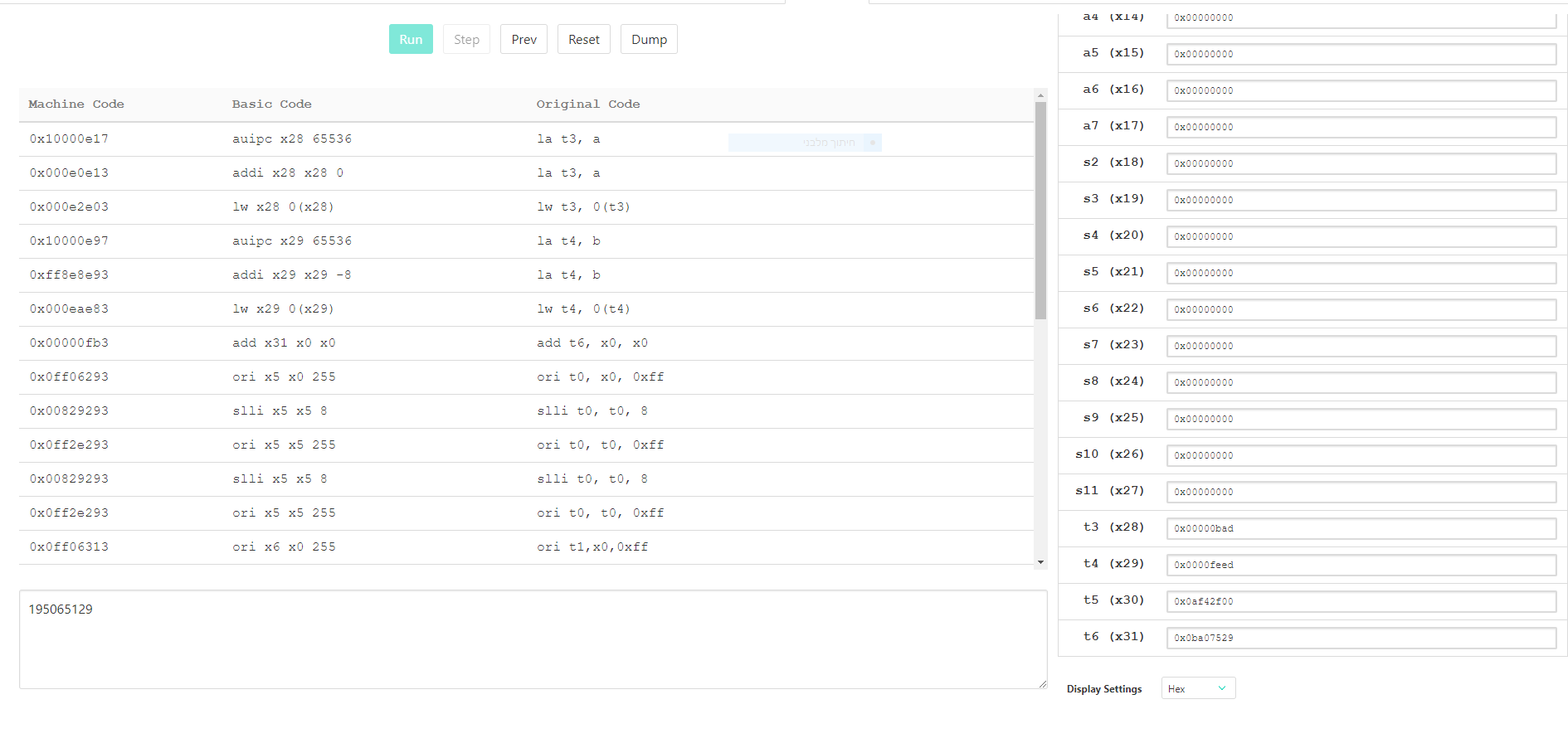
7.1.3 result= result+mul

7.1.4 j=j+1

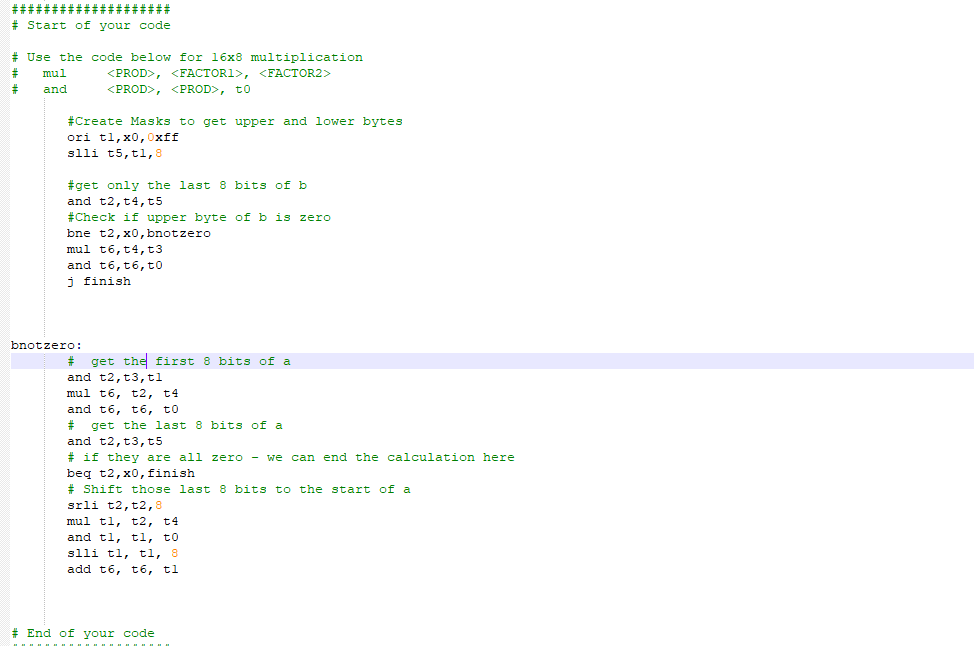
7.2 i=i+1

8. תוצאת הכפל שמורה בresult.

סיבוכיות זמן הריצה של האלגוריתם היא o (). באלגוריתם ישנה לולאה מקוננת, הלולאה הפנימית מתבצעת N/2 פעמים, החיצונית מתבצעת N פעמים. כלומר סה"כ זמן הריצה יהיה o(N\*N/2)=o().

2.4

החישוב מתבצע בתוך 11 פקודות, לא כולל אתחול וfinish. בהנחה שפקודה לוקחת מחזור שעון אחד, החישוב יתבצע תוך 11 מחזורי שעון.



2.5

הגרסה המעודכנת של הקוד בודקת בתחילת התוכנית האם הבית העליון של b מאופס. במקרה וכן, ניתן לבצע כפל 16x8 רגיל באמצעות mul, שכן אפקטיבית לb יש רק 8 ביטים משמעותיים.  
אחרת, מתבצע הכפל כמו בסעיף הקודם – קודם כופלים את הבית התחתון של a בb, ולאחר מכן מבודדים את הבית העליון של a. כעת בודקים האם הבית העליון של a מאופס. במקרה וכן, ניתן לדלג לסוף, שכן כפל הבית העליון של a בb יהיה שווה ל-0.

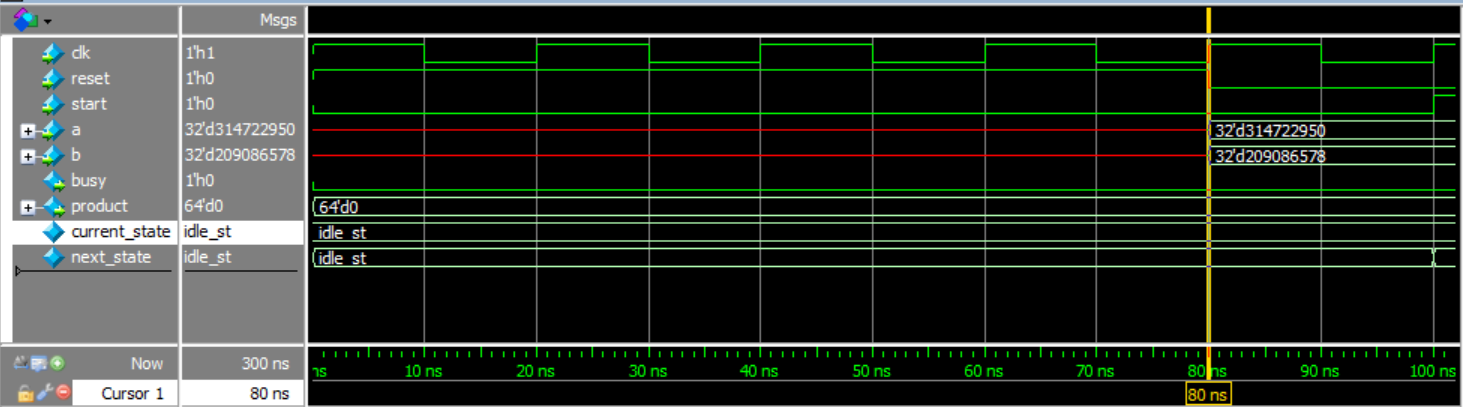
נבדוק את ההשפעות על זמן הריצה:

* כאשר הבית העליון של b מאופס: ללא תלות בערך של , a מבצעים 7 פקודות, כלומר זמן הריצה הוא 7 מחזורי שעון – 4 פחות מזמן הריצה בסעיף הקודם.
* כאשר הבית העליון של b לא מאופס אבל הבית העליון של a מאופס: מבצעים 9 פקודות, כלומר זמן הריצה הוא 9 מחזורי שעון – 2 פחות מזמן הריצה בסעיף הקודם.
* כאשר הבתים העליונים של a ו b שניהם לא מאופסים: מבצעים 14 פקודות, כלומר זמן הריצה הוא 14 מחזורי שעון – 3 יותר מזמן הריצה בסעיף הקודם.

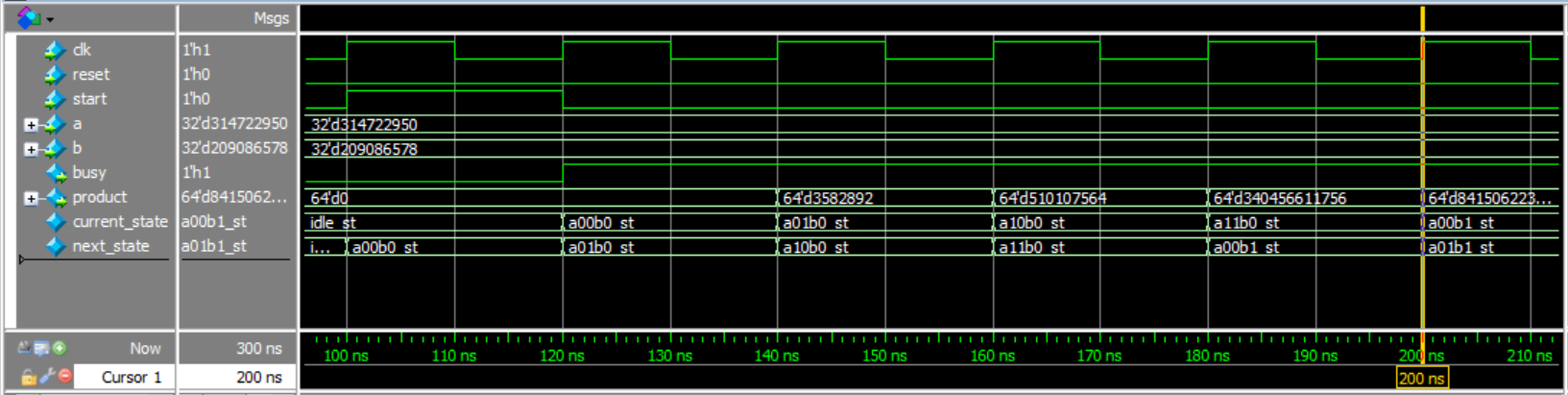
לסיכום, השינויים המוצעים בסעיף עשויים לשפר את פעולת התוכנית במקרה שהבית העליון של לפחות אחד מהמשתנים הוא 0, אבל בכל מקרה אחר יש פגיעה ביעילות המכפל.   
מטרת המכפל היא לכפול שני מספרים בני 16 סיביות כל אחד, ללא תלות בתנאים מיוחדים על המספרים עצמם. כלומר, למרות שבתנאים מסוימים הגרסה המעודכנת של המכפל עשויה להיות יותר יעילה, הפעילות השגרתית שלו נהיית פחות יעילה. יש יותר מספרים בהם הבית העליון לא מאופס מאשר מספרים בהם הבית העליון כן מאופס, ולכן ברוב המקרים תהיה פגיעה ביעילות התכנית.   
בהתחשב בכלל השיקולים נראה כי השינוי המוצע לא משתלם.

3.4

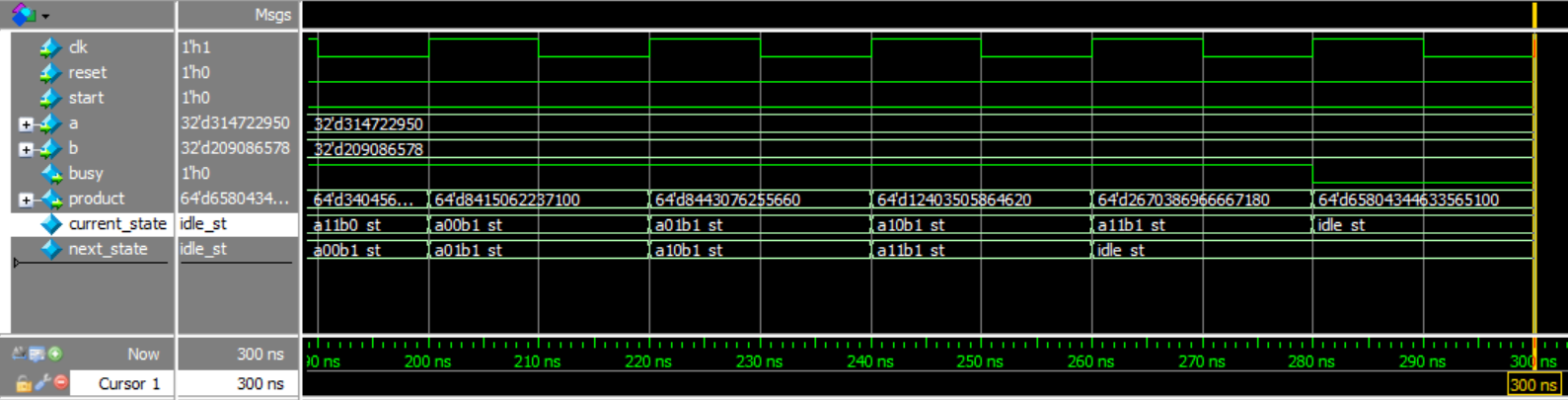
הגדרנו שכל 10ns השעון משנה את מצבו כלומר זמן מחזור השעון הוא 20ns.

הדיאגרמה מתחילת הסימולציה עד למעבר של 5 זמני מחזור שעון(0-100ns):

הדיאגרמה ממעבר של 5 זמני מחזור שעון עד מעבר של 10 זמני מחזור שעון (100ns-200ns):



הדיאגרמה ממעבר של 10 זמני מחזור שעון עד מעבר של 15 זמני מחזור שעון (200ns-300ns):



בדיארמת הגלים ניתן לראות כי הbusy עולה מחזור שעון אחד לאחר עליית start, וערך busy הוא 1 במשך 8 זמני מחזור שעון. כלומר כמו שצפינו בחלק התיאורטי פעולת הכפל לוקחת 9 מחזורי שעון. בנוסף ניתן לראות שבכל עליית שעון current\_state ו-next\_state, מתעדכנים בדיוק לפי מכונת המצבים שתכננו בסעיף 2.1.

לבסוף ניתן לראות שproduct לאחר השלמת פעולת הכפל אכן מקבל את התוצאה הצפויה:

314722950 X 209086578 = 65804344633565100