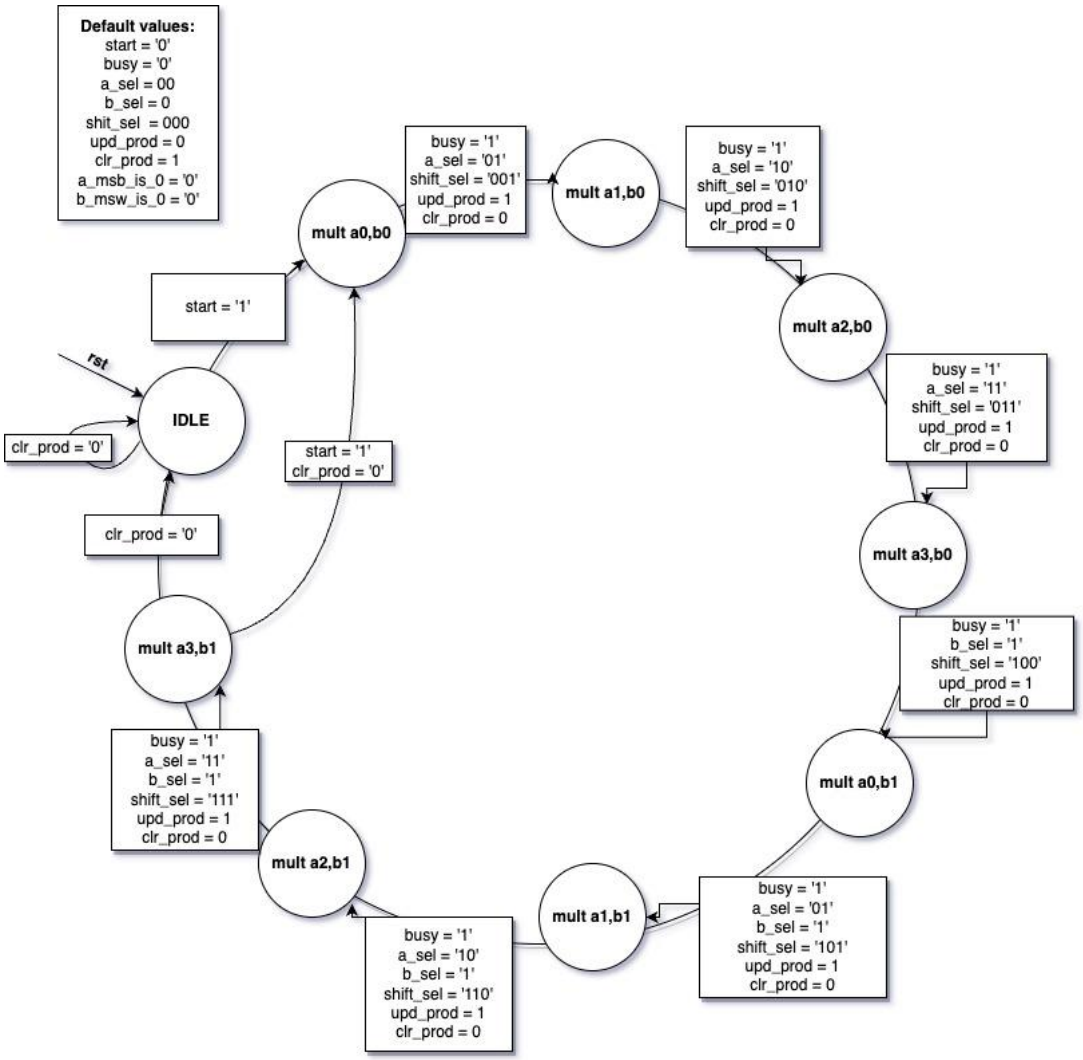
דנה ניפדוב	209381649
אווה פולוליאחוב	321882649

ספרתיות - רטוב 2

2.1

תכננו מכונת מצבים (FSM) מסוג Mealy השולטת על פעולת הכפל בתכן הנתון בשרטוט. שימו לב שכדי שמכונת מצבים תהיה מסוג Mealy, מספיק שלפחות במצב אחד, אחת היציאות תהיה תלויה צירופית באחת הכניסות.

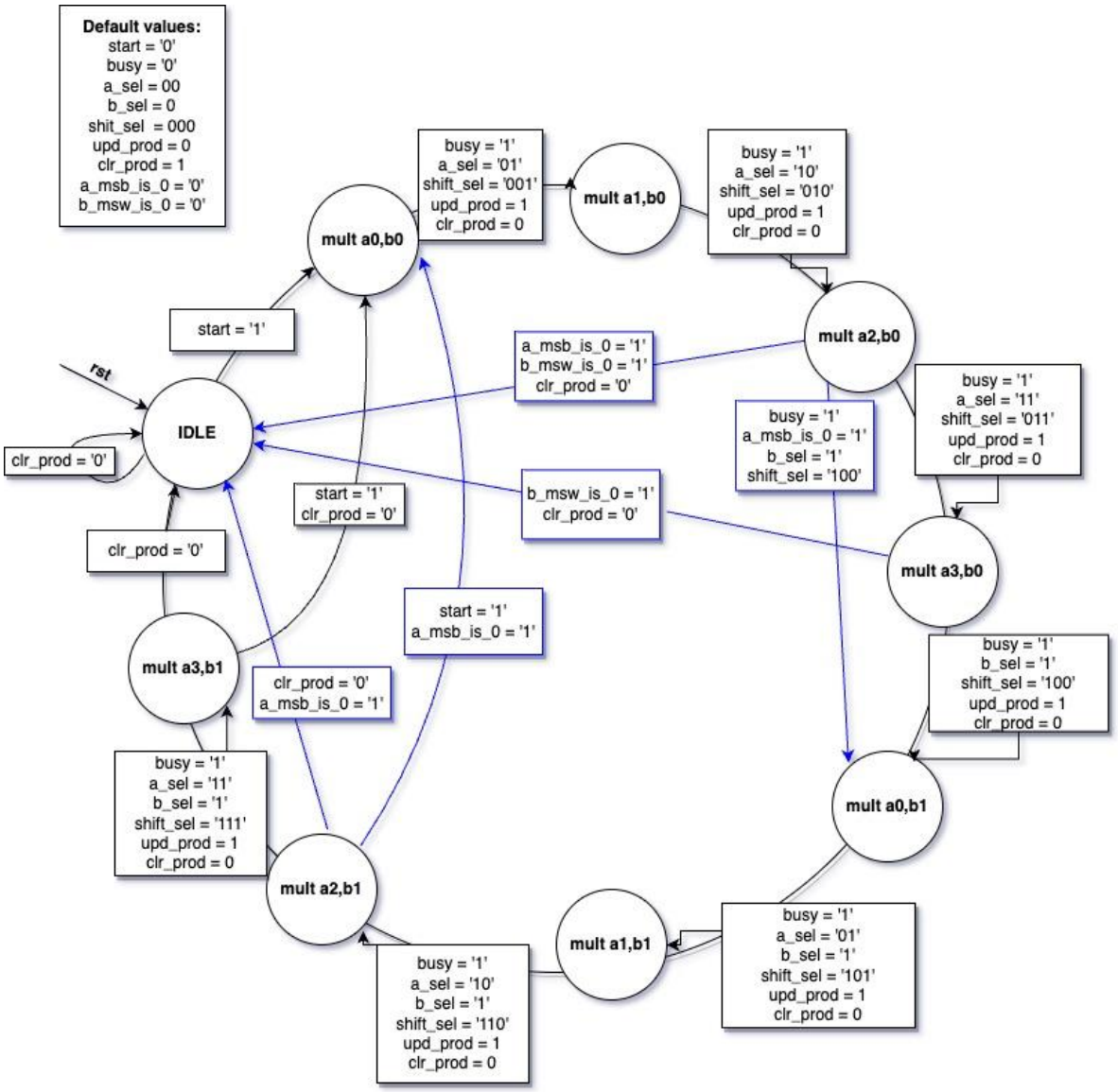
מצ"ב תכנון מכונת מצבים בעזרת דיאגרמה המתארת מכפל 32x32 בעזרת מכפל 16x8 ותשעה מצבים, התומכת בביצוע מספר לא מוגבל של פעולות כפל אחת אחרי השנייה ללא reset ביניהן, כך שתוצאת המכפל נשמרת עד לקבלת start נוסף.
כל אחד מהמצבים לוקח מחזור שעון שלם ולכן סך הכל פעולת מכפל שלמה לוקחת תשעה מחזורי שעון מרגע קבלת start = 1.



2.2

כאשר בית מסוים (8 סיביות) של אחד הגורמים במכפלה שווה ל-0, תוצאת כפל של הבית הזה בכל מספר אחר היא גם 0. ניתן לנצל את התכונה הזו כדי לייעל את זיהוי של הבתים האלה מראש ודילוג על שלבי החישוב תהליך ההכפלה על ידי הרלוונטיים במכונה.
הציעו שינוי במכונת המצבים המקורית כך שהפעולה תהיה מהירה יותר אם ה- Most Significant Byte (סיביות 24 עד 31 כולל) בכניסת A ו/או ה-Most Significant Word (סיביות 16 עד 31 כולל) בכניסה B שווים ל-0.

השינוי המוצע במערכת כתלות בשני סיגנלים נוספים נתונים :



השפעת השינויים על המערכת:
מספר מחזורי שעון שדורשת פעולת הכפל בהינתן השינויים:

	a_msb_is_0 = '0' b_msb_is_0 = '0'	a_msb_is_0 = '0' b_msb_is_0 = '1'	a_msb_is_0 = '1' b_msb_is_0 = '0'	a_msb_is_0 = '1' b_msb_is_0 = '1'
מספר מחזורים	מערכת נותרת ללא שינוי, 9	5	7	4

לכן ניתן לראות שפעילות המערכת המהירה ביותר קוראת במצב בו a_msb_is_0 = '1' וגם b_msb_is_0 = '1'

2.3

מימוש פעולת כפל 16x16 באמצעות פקודת כפל 16x8 בתוכנה

בדומה לנעשה בשאלות הקודמות, גם בתוכנה משתמשים במשאבים מוגבלים כדי לבצע משימות מורכבות.

לצורך תרגיל זה, הניחו שברשותכם מעבד שיודע לכפול מספר בגודל 8 סיביות במספר בגודל 16 סיביות ולהוציא תוצאה בגודל 24 סיביות.

עליכם לתכנן אלגוריתם תוכנה שכופל שני מספרים בגודל 8N סיביות, כאשר N מספר טבעי וזוגי. ניתן להניח כי המספרים בייצוג unsigned. כמו כן, ניתן להניח כי המשתנים שמשתמשים בהם באלגוריתם גדולים ככל שתרצו.

תארו בפירוט את האלגוריתם ואת אופן פעולתו. ניתן, אך לא חובה, להשתמש גם בתרשים זרימה ו/או pseudo code לצורך תיאור האלגוריתם. ציינו את הקשר בין N לבין זמן הביצוע של האלגוריתם (סיבוכיות זמן ריצה).

נסמן את שני המספרים בגודל $8N$ ב- A ו- B. נגדיר משתנה סכימה שישמור את התוצאה הסופית של פעולות המכפל.

נרוץ בלולאה על המספר A ונחלק אותו למקטעים בגודל 16 סיביות $(i = 0,\ i < N/2)$, נסמן את המקטע הראשון ב-a_0, את המקטע השני ב- a_1, וכך הלאה עד למקטע ה-$N/2$ ב-a_{N-1}.

בלולאה מקוננת, נרוץ על המספר B ונחלק אותו למקטעים בגודל 8 סיביות $(j = 0,\ j < N)$, נסמן את המקטע הראשון ב-b_0, את המקטע השני ב-b_1, וכך הלאה עד למקטע ה-N ב-b_{N-1}.

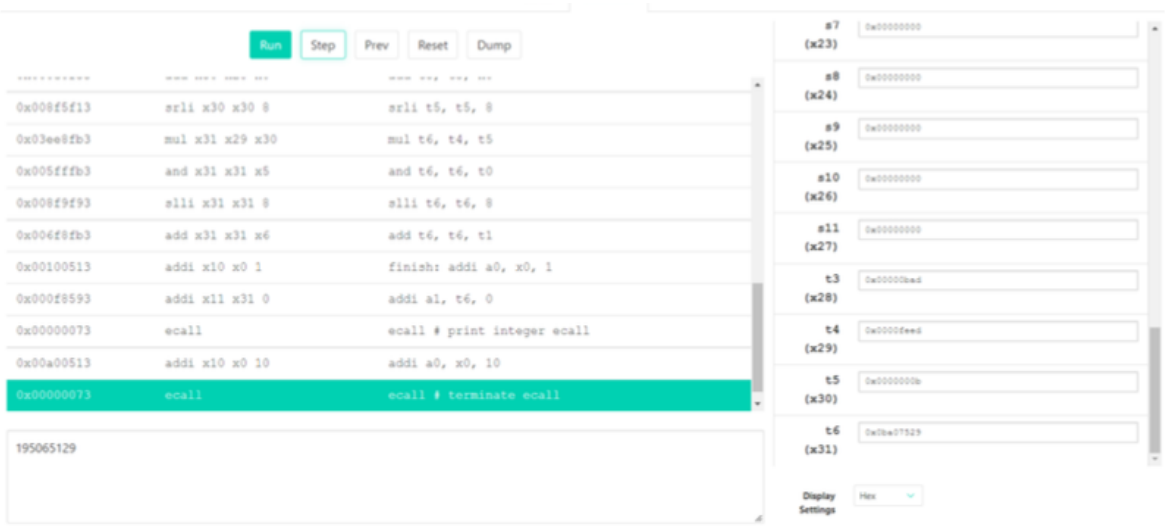
לכל $i < N/2$ ו- $j < N - 1$: נכניס למכפל את a_i, ו- b_i, לתוצאה נעשה *shift* של $j + i \cdot 8$,
ונסכום אותה לתוך משתנה הסכימה שמכיל את התוצאות הקודמות.
בסוף הלולאה משתנה הסכימה יכיל את תוצאת המכפלה של A ב- B.

הקשר בין זמן הביצוע ל- N הוא קשר ריבועי:
ככל ש- N גדל פי 2 זמן הריצה של האלגוריתם גדל פי 4.

2.4

בקובץ mult16x16.s כתבו קוד אסמבלי של RISC-V שכופל שני מספרים בגודל 16 סיביות תוך שימוש בפעולת כפל של 16x8 סיביות ופעולות נוספות. ניתן לפתוח ולערוך את הקובץ עם notepad++.

הריצו את הקוד בסימולטור שבאתר: http://www.kvakil.me/venus
הוסיפו לחלק היבש צילום מסך של הסימולטור לאחר הרצת הסימולציה. ודאו שערך האוגר t6 מופיע בצילום.



פעולת הכפל לוקחת 10 מחזורי שעון.

2.5

תארו מהו השינוי הנדרש בקוד מסעיף 0 כדי לממש דילוגים על אפסים בבית העליון של a ו/או b (בדומה לסעיף 0), כלומר כאשר כל הבית העליון של a מאופס, כל הבית העליון של b מאופס, או שניהם. ניתן לצרף את הקוד החדש או להסביר את השינויים בלבד. הבדיקה האם בתים אלו מאופסים צריכה להיות כלולה באלגוריתם. איך ישתנה זמן הריצה של התוכנה? האם השינוי הזה משתלם?

```
add t5, t3, x0
andi t5, t5, 0xff
mul t6, t4, t5
and t6, t6, t0

add t5, t3, x0
srli t5, t5, 8

beq t5, x0, finish

mul t1, t4, t5
and t1, t1, t0

slli t1, t1, 8

add t6, t6, t1
```

נכפיל את b בבית התחתון של a,נבדוק אם הבית העליון של a שווה לאפס, אם כן נקפוץ ל- finish, אחרת נמשיך להכפיל את הבית התחתון של a ב-b ולחבר בין התוצאות.

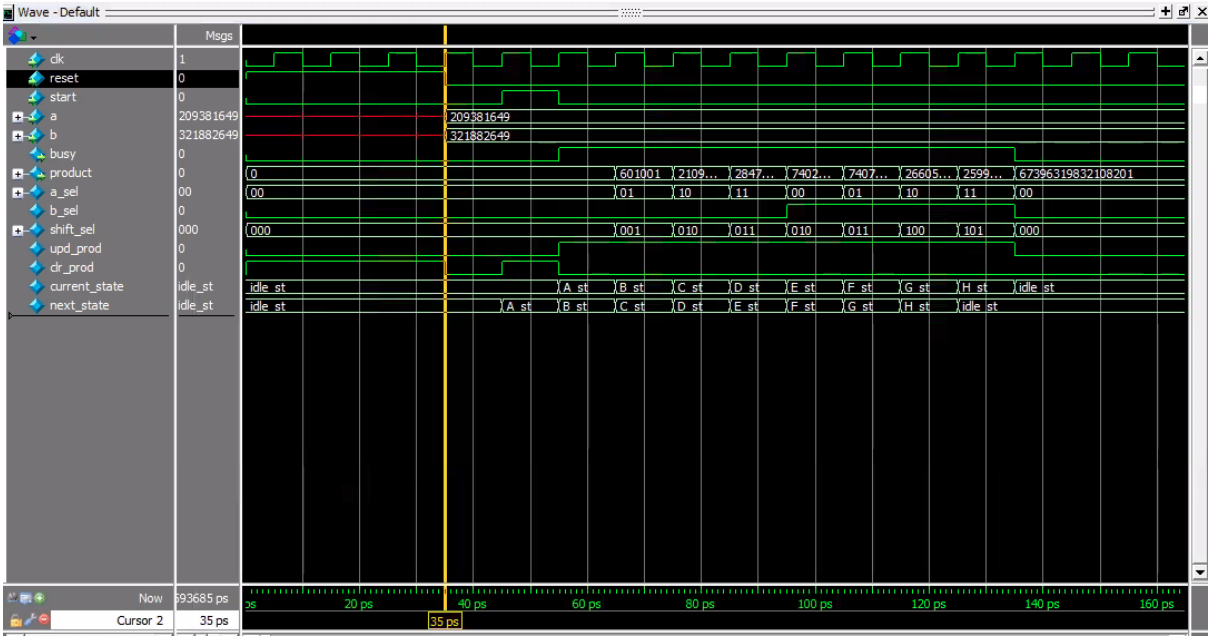
אם הבית העליון של a שונה לאפס, אז נכפיל את הבית התחתון של a ב-b, נכפיל את הבית העליון של a ב-b, נעשה לתוצאה זו shift 8 בתים שמאלה ונחבר עם התוצאה הקודמת, ונעבור ל-finish.

את b לא מחלקים לבתים כי פעולת ההכפלה שסיפקו לנו מכפילה בכל מקרה 16 ביטים ב-8 ביטים, ולכן אין השפעה על זמן הפעולה כי פעולת ההכפלה לוקחת מחזור שעון אחד.
זמן הריצה של התוכנית ישתנה בהתאם לבית העליון של a:

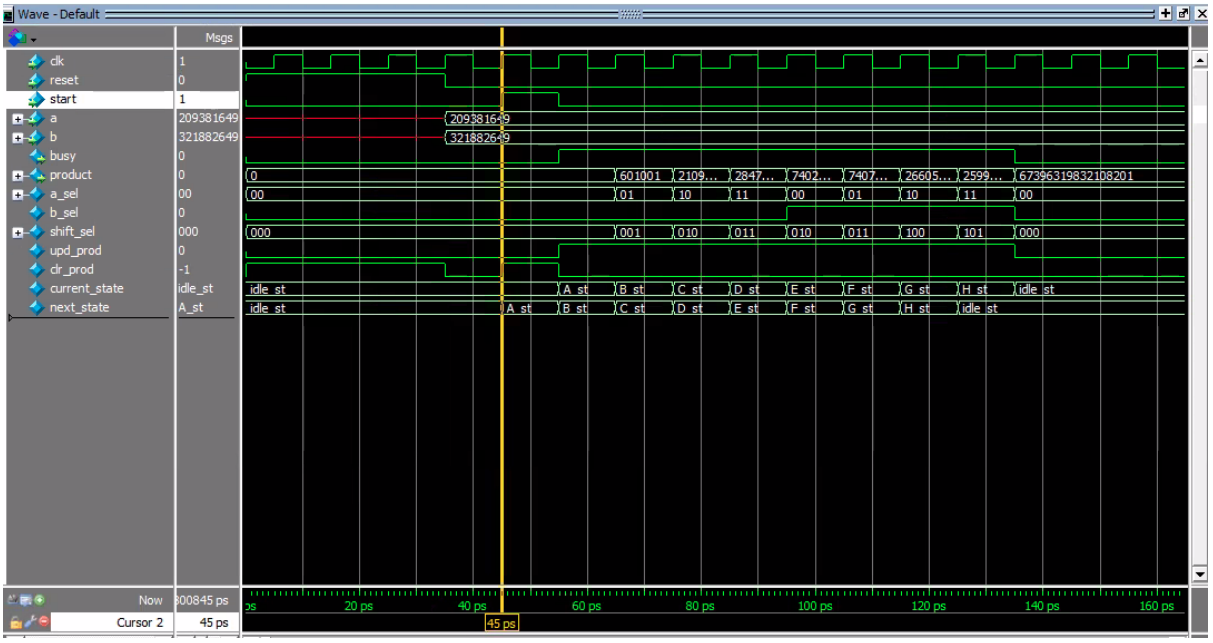
	MSB a ≠ 0	MSB a = 0
כמות מחזורי שעון להשלמת פעולת כפל	11	7

3.1

נבחין כי לאחר ארבעה מחזורי שעון התקבל סיגנל של reset שמיד לאחריו התעדכנו ערכי a , b.



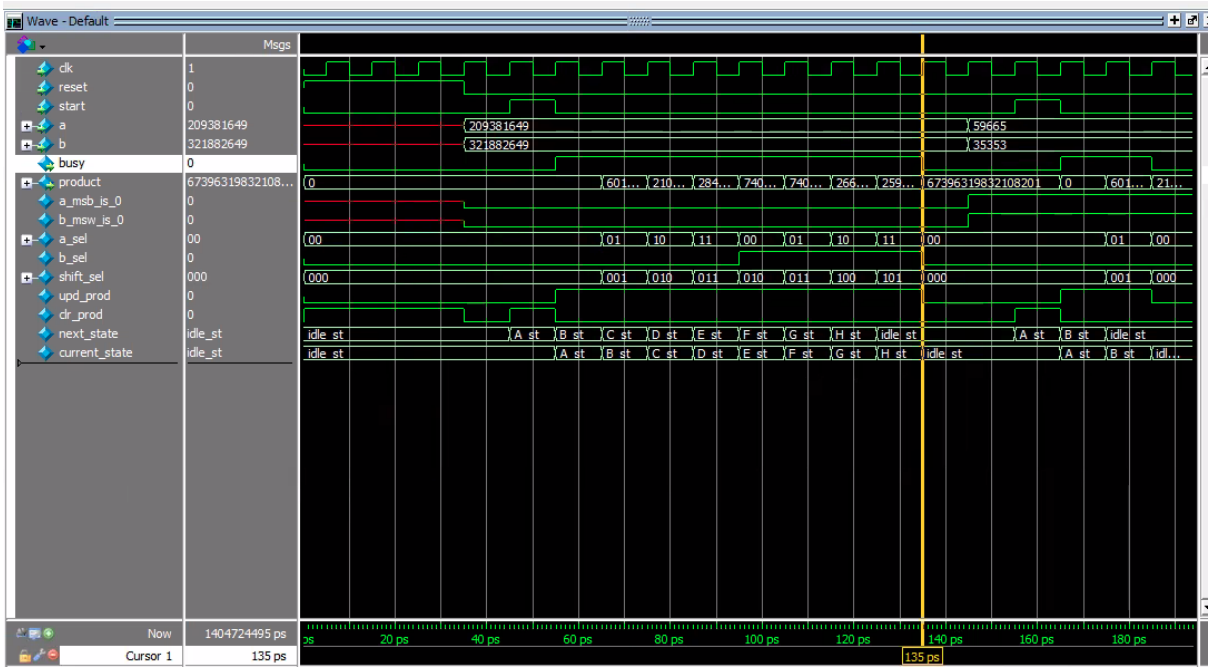
לאחר מחזור שעון אחד התעדכן start = 1 למשך מחזור שעון אחד וברגע שערכו התעדכן חזרה ל0 , משתנה הסיגנל busy = 1 ואנחנו עוברים לשלב הראשון של ה FSM שבו מתחילה פעולת המכפל.



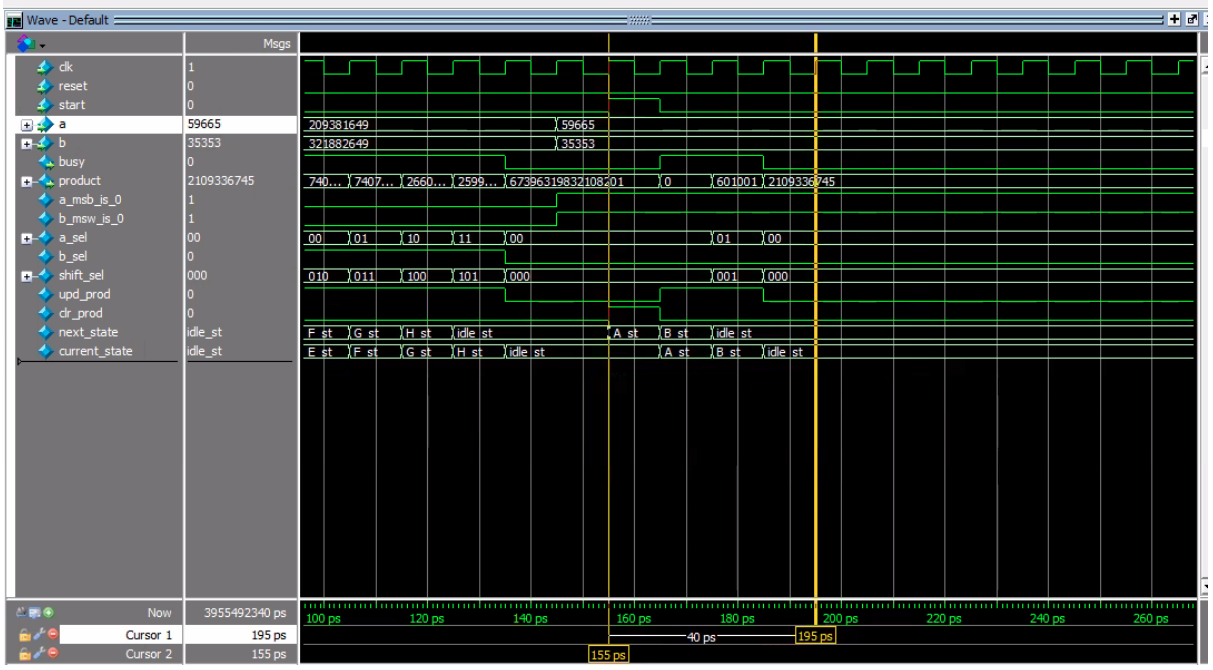
נבחין כי busy =1 למשך של 8 מחזורי שעון שבו מסתיימת פעולת המכפל ואנחנו חוזרים למצב idle וגם ערך busy = 0 מתייצב. לכן בסך הכל מרגע קבלת סיגנל start=1 עד לסוף פעולת המכפל יתקיימו 9 מחזורי שעון.
בנוסף נבחין כי תוצאת המכפל הנשמרת במשתנה product נשמרת קבועה לאחר סיום הפעולה מאחר ולא התקבל סיגנל start = 1.

3.2

נבחין כי בהינתן התנאים כמו בסעיף הקודם, ולאחר המתנה להתייצבות הערך busy = 0, קיבלנו את אותו הערך במשתנה product שנשאר יציב למשך שני מחזורי שעון שלמים ללא שינוי, כל עוד המשתנה start = 0.



כעת לאחר המתנה, הצבנו start = 1 וניתן לראות כי אכן לאחר עליית השעון, משתנה product התאפס וגם עברנו במצב idle כמצופה , זאת מאחר ולא ביצענו חישוב חדש מיד לאחר סיום הקודם.



מאחר ובמכפל זה הצבנו a_msb_is_0 = 1 וגם b_msb_is_0 = 1, נצפה לבצע את המסלול הקצר ביותר שחישבנו שאורכו 4 מחזורי שעון מרגע שבו start = 1 ועד לרגע שבו busy = 0.
ואכן ניתן לראות כי עד לרגע שבו מתייצב הסיגנל product עוברים שלושה מחזורי שעון ועוד מחזור שעון נוסף עד להתייצבות של busy = 0, לכן סך הכל זמן פעולת מכפל במצב הנוכחי הוא 4 מחזורי שעון כמצופה.