

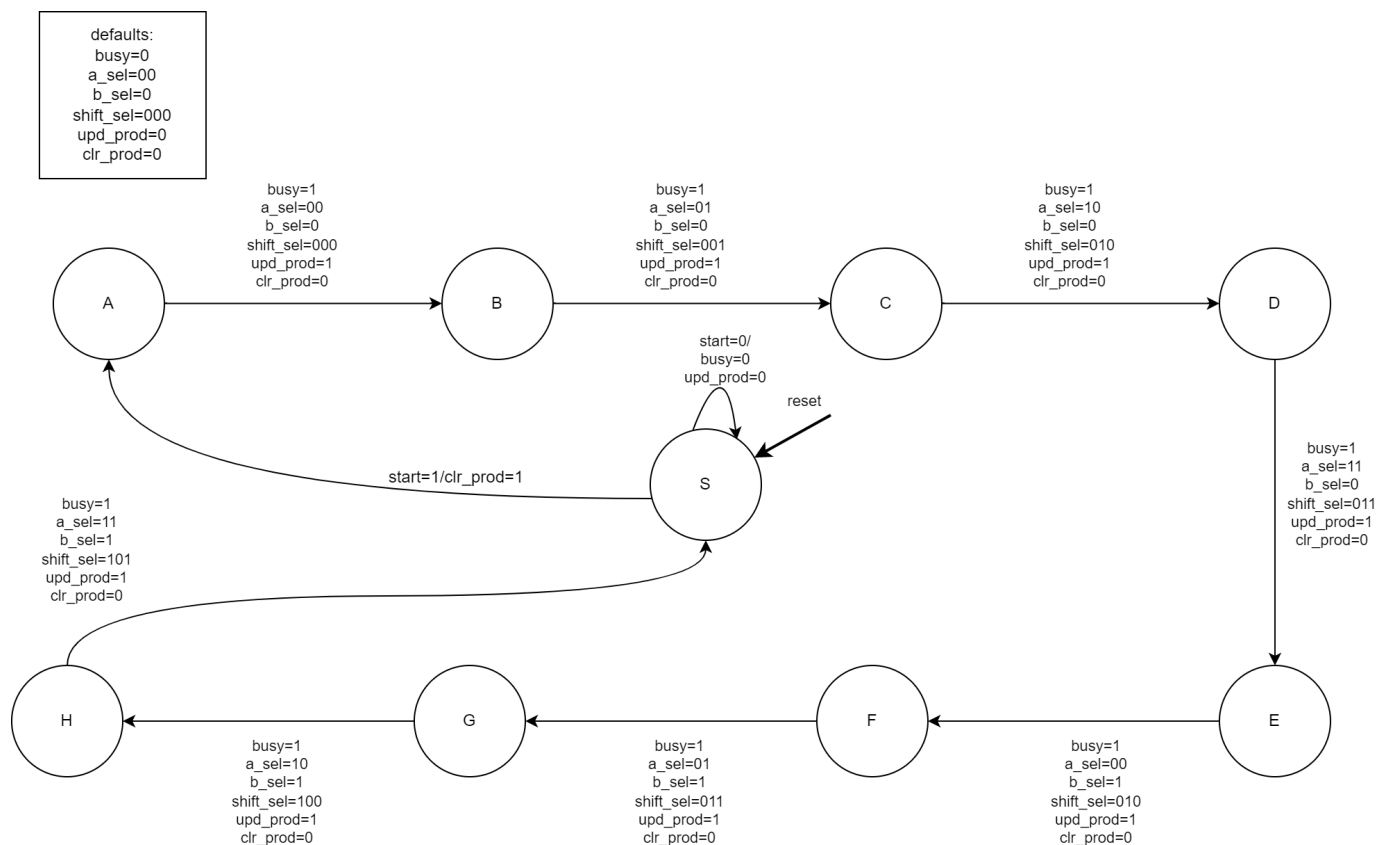
## מערכות ספרתיות - סימולציה 2

מגשים:

עידו טאוס	214008997
נועם ביטון	213745953

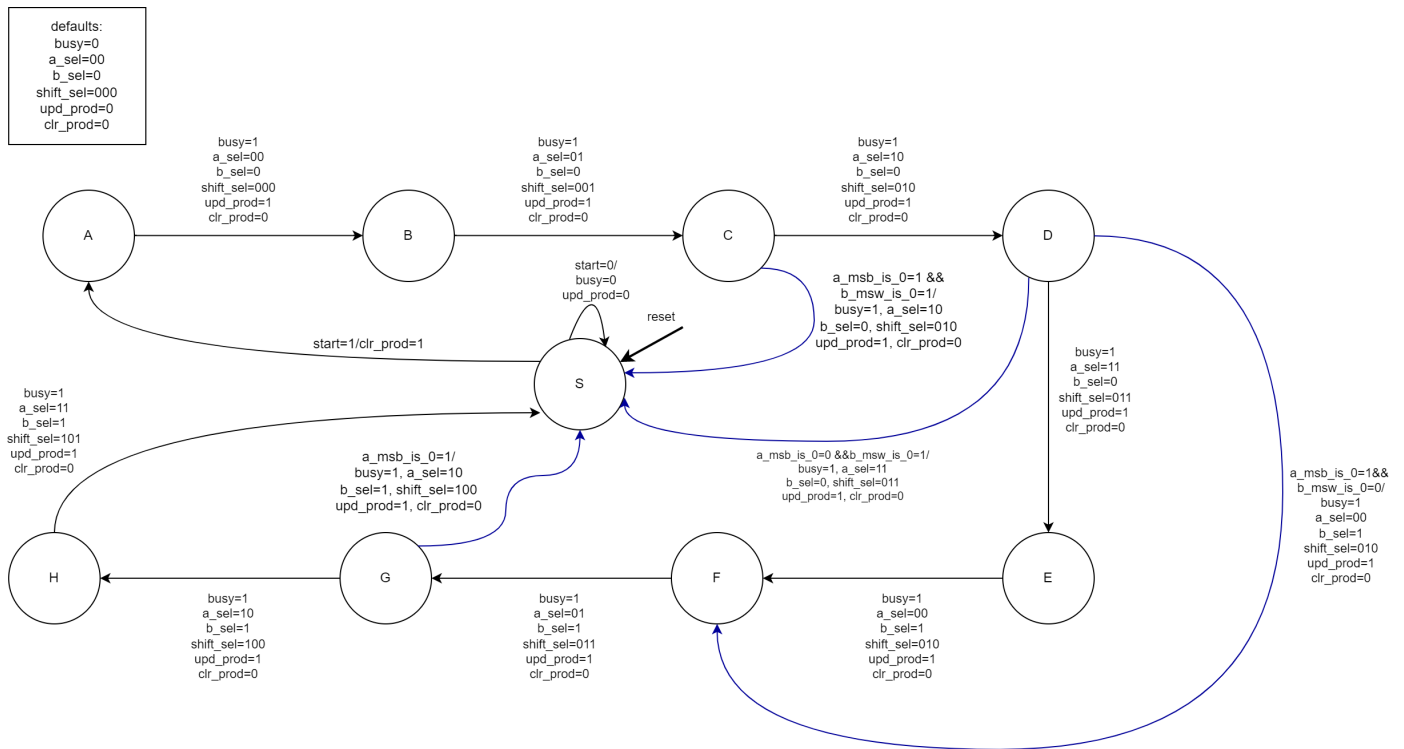
חלק יבש:

סעיף 2.1:



במכונת המצבים שציירו, פעולת הכפל הנדרשת שמתבצעת לוקחת 9 מחזורי שעות.

## סעיף 2.2:



כאשר  $a\_msb\_is\_0=0 \ \&\& \ b\_msw\_is\_0=0$ : ייקח לנו 9 מחזורי שעון, כמו בסעיף א'.  
 כאשר  $a\_msb\_is\_0=0 \ \&\& \ b\_msw\_is\_0=1$ : ייקח לנו 5 מחזורי שעון.  
 כאשר  $a\_msb\_is\_0=1 \ \&\& \ b\_msw\_is\_0=0$ : ייקח לנו 7 מחזורי שעון.  
 כאשר  $a\_msb\_is\_0=1 \ \&\& \ b\_msw\_is\_0=1$ : ייקח לנו 4 מחזורי שעון.

### סעיף 2.3:

#### האלגוריתם:

נסמן מספר אחד ב- $a$  ואת המספר השני נסמן ב- $b$ .

נתחיל בביצוע האלגוריתם, נאתחל משתנים  $sum = 0, i = 0$ .

בכל איטרציה כאשר  $i = 0$  עד  $i = \frac{N}{2} - 1$ , ניקח את 16 הספרות של  $b$  החל מהמיקום ה- $16i$  עד המיקום ה-

$16i + 15$ .

כעת נאתחל משתנה  $j = 0$  ובכל איטרציה כאשר  $j = 0$  עד  $j = N - 1$ , ניקח את 8 הספרות של  $a$  החל מהמיקום ה- $8j$  עד המיקום ה- $8j + 7$ . בכל איטרציה נבצע כפל של 16 הספרות הנוכחיות של  $b$  ב-8 הספרות הנוכחיות של  $a$ , נוכל לבצע זאת בעזרת המעבד שיודע לכפול מספרים בעלי 8 סיביות במספרים בעלי 16 סיביות. כעת, יש צורך לעשות shift left לתוצאה כך שנוכל לחבר אותה בצורה נכונה לסכום המכפלות עד כה, לכן נבצע shift left ב- $8j + 16i$ . נכון לבצע הזזה זו מפני שיש לנו  $8j + 16i$  ספרות לפני הספרות הנוכחיות שלנו שלא התייחסנו אליהן, לכן צריך לבצע הזזה שמאלה על מנת לקבל תשובה נכונה. כעת, נוסיף את המכפלה הנוכחית שבוצע עליה shift left באיטרציה זו, לסכום כל המכפלות עד כה  $sum$ .

בסיום כל האיטרציות, החיצוניות והפנימיות, נקבל כי במשתנה  $sum$  נמצאת התוצאה של המכפלה הרצויה, לכן נחזיר אותה.

נשים לב כי ביצענו  $\frac{N}{2}$  איטרציות, שבכל אחת מהן ביצענו  $N$  איטרציות, לכן בסך הכול ביצענו  $\frac{N^2}{2}$  פעולות, לכן סיבוכיות זמן הריצה של האלגוריתם הוא  $O(N^2)$ .

#### ובפסודו קוד:

```
sum = 0;
current;
for(int i=0; i<N/2; i++)
{
    for(int j=0; j<N; j++)
    {
        current = a[8j:8j+7] * b[16i:16i+15];
        doShiftLeft(current, 8j+16i);
        sum+= current;
    }
}
```

## סעיף 2.4:

צילום מסך של הסימולטור לאחר ההרצה:

RunStepPrevResetDump

Machine Code	Basic Code	Original Code
0x10000e17	auipc x28 65536	la t3, a
0x000e0e13	addi x28 x28 0	la t3, a
0x000e2e03	lw x28 0(x28)	lw t3, 0(t3)
0x10000e97	auipc x29 65536	la t4, b
0xff8e8e93	addi x29 x29 -8	la t4, b
0x000eae83	lw x29 0(x29)	lw t4, 0(t4)
0x00000fb3	add x31 x0 x0	add t6, x0, x0
0x0ff06293	ori x5 x0 255	ori t0, x0, 0xff
0x00829293	slli x5 x5 8	slli t0, t0, 8
0x0ff06293	ori x5 x5 255	ori t0, t0, 0xff

195065129

(x23)

s80x00000000(x24)

s90x00000000(x25)

s100x00000000(x26)

s110x0000ff00(x27)

t30x0000bad(x28)

t40x000feed(x29)

t50x00000000(x30)

t60x0ba07529(x31)

DisplayHex

ניתן לראות שאכן יצא לנו  $t6 = 0x0BA07529$  כמצופה מחישוב הכפל:  $0xBAD * 0xFEED$

ביצענו את הפעולה ב-10 שורות כאשר כל שורה היא פקודה שזמן ביצועה הוא זמן מחזור אחד ולכן זמן ביצוע הפעולה הוא 10 מחזורי שעון.

## סעיף 2.5:

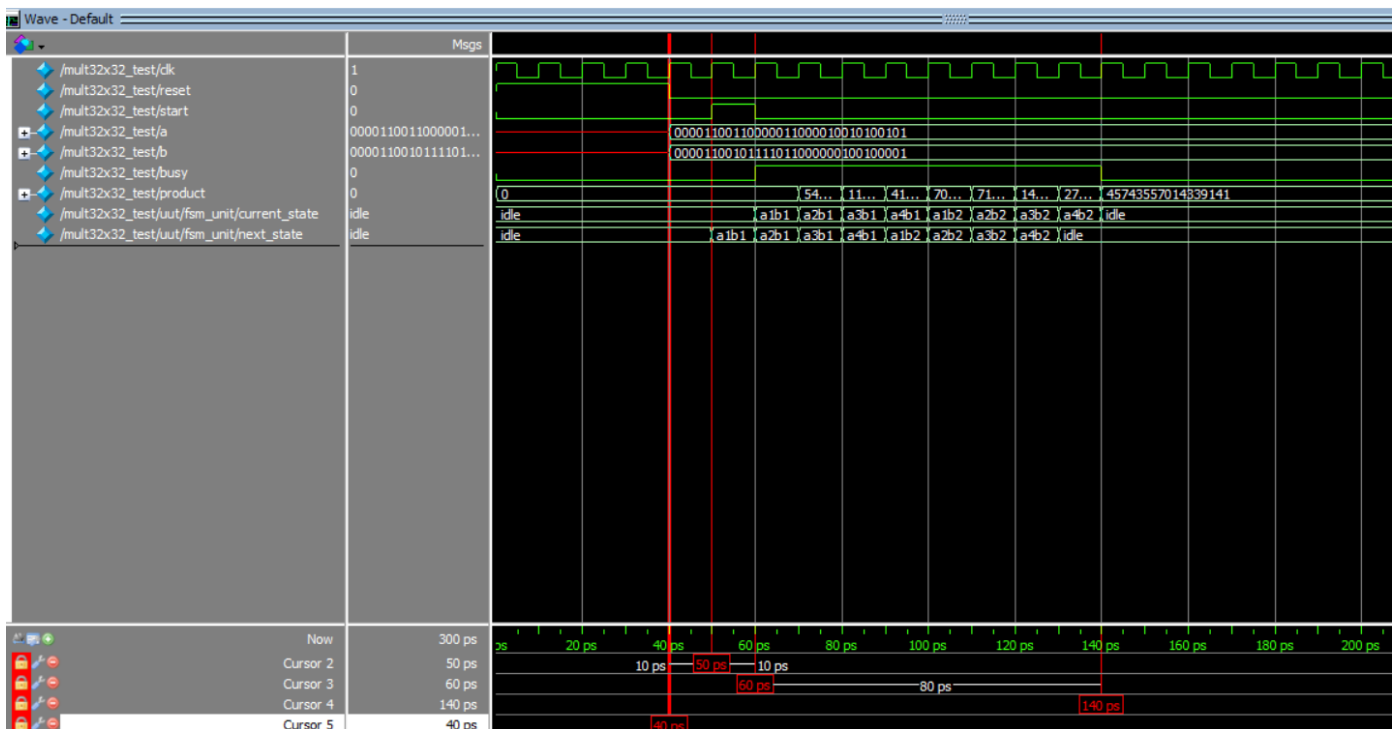
נוסיף בדיקה על הבית הראשון של  $a$  ועל הבית הראשון של  $b$ , נוכל לבצע זאת בעזרת בנייה של mask שיכיל  $0xFF00$ , ואז ביצוע bitwise AND, וכך נקבל את הבית העליון ב- $a$  וב- $b$ .  
אם הבית העליון של אחד מהם שווה ל-0, נתייחס אליו בתור מספר של 8 ביטים, מפני ש-8 הביטים העליונים שלו הם 0, וכעת נוכל לחשב ישירות את המכפלה שלהם, מפני שאנו יודעים לבצע מכפלה של  $8 \times 16$  באופן ישיר.

ניתן לבנות את ה-mask הרלוונטי ב-2 שורות, ואז נבצע AND עם  $a$ , ונבצע branch אם הוא שווה ל-0, לשורה בה יתבצע הכפל באופן ישיר של  $8 \times 16$  כאשר  $a$  הוא המספר בעל 8 הסיביות, ואחריה נקפוץ ל-finish, סך הכול 5 שורות, לכן 5 מחזורי שעון. אם תוצאת ה-AND עם  $a$  לא שווה ל-0, נבצע AND של ה-mask עם  $b$  ונבצע branch אם התוצאה שווה ל-0, נקפוץ לשורה בה יתבצע הכפל באופן ישיר של  $8 \times 16$  כאשר  $b$  הוא המספר בעל 8 הסיביות, ואחריה נקפוץ ל-finish, סך הכול 8 שורות, לכן 8 מחזורי שעון.

נשים לב כי בניית ה-mask וביצוע ה-AND ובדיקת השוואה ל-0, כולן יקרו לפני הקוד של הסעיף הקודם, ולכן יאריכו את זמן הריצה שלו, במקרה שבו הבית העליון של  $a$  וגם הבית העליון של  $b$ , שניהם שונים מ-0. שינוי זה יהיה משתלם רק במקרים בהם הבית העליון של  $a$  או של  $b$  הוא 0, במקרים אחרים הוא לא יהיה משתלם ואף יגרום להאטה.

## חלק רטוב:

### סעיף 3.4:



הגדרנו מחזור שעון להיות  $10ps$ .

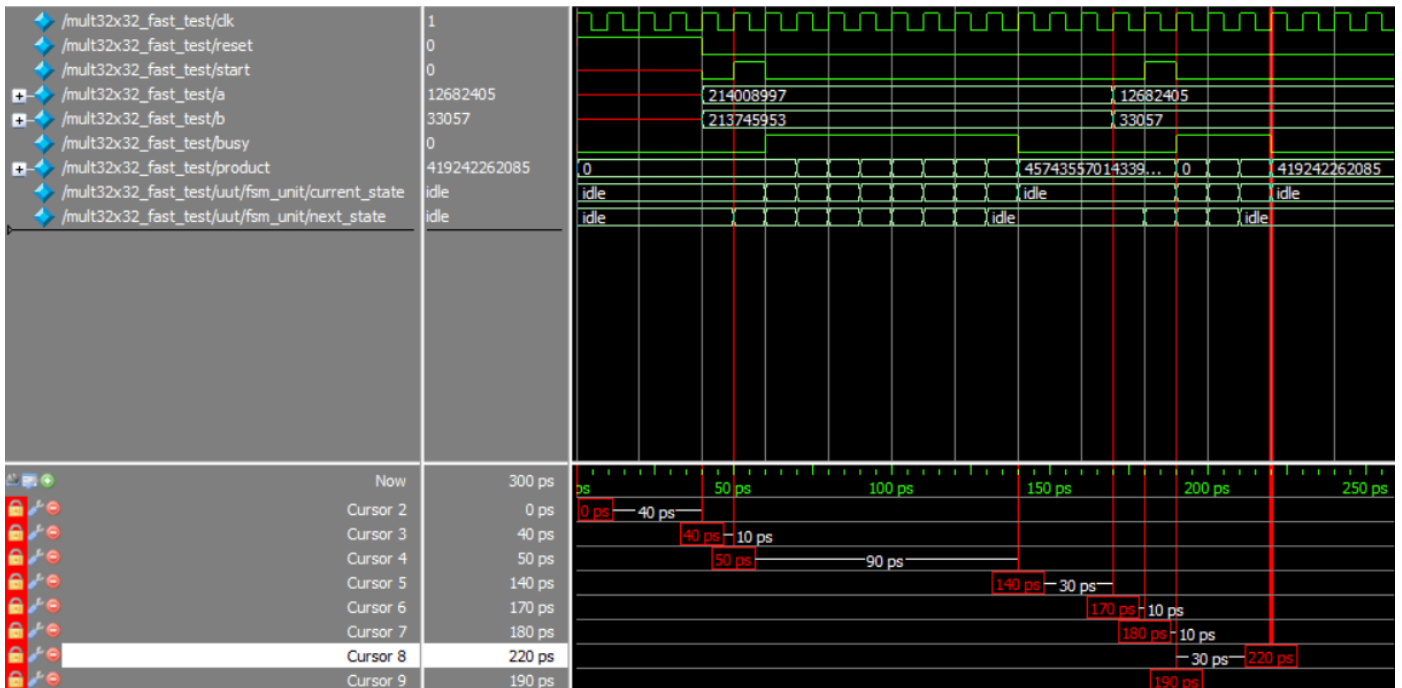
ניתן לראות כי אות ה-*reset* מאותחל ב-0 ודלוק 4 מחזורי שעון ( $40ps$  שניות) ולאחר מכן יורד עד סוף הסימולציה. עם ירידת אות ה-*reset* הוצבו בכניסות *a* ו-*b* ערכי תעודות זהות שלנו.

בנוסף, לאחר מחזור שעון נוסף ( $50ps$  מתחילת הריצה) אות ה-*start* עולה למשך מחזור שעון אחד. אות ה-*busy* עולה מיד עד ירידת אות ה-*start* והמכונה מתחילה את החישוב. לאחר 8 מחזורי שעון מעליית *busy* נראה כי החישוב הסתיים והאות *busy* יורד כאשר ב-*product* מופיעה התוצאה הסופית של המכפלה.

תעודות הזהות שלנו הן: 213745953, 214008997.

ומכפלתן היא 45743557014339141, וזאת אכן התוצאה שקיבלנו ב-*product* בסוף הריצה כאשר *busy* ירד ל-0.

### סעיף 3.7:



גם במקרה זה, הגדרנו את מחזור השעון להיות  $10ps$ . ניתן לראות כי אות ה-*reset* מאותחל ב-0 ודלוק 4 מחזורי שעון ( $40ps$  שניות) ולאחר מכן יורד עד סוף הסימולציה. עם ירידת אות ה-*reset* הוצבו בכניסות *a* ו-*b* ערכי תעודות זהות שלנו. בנוסף, לאחר מחזור שעון נוסף ( $50ps$  מתחילת הריצה) אות ה-*start* עולה למשך מחזור שעון אחד. אות ה-*busy* עולה מיד עם ירידת אות ה-*start* והמכונה מתחילה את החישוב. לאחר 8 מחזורי שעון מעליית *busy* נראה כי החישוב הסתיים והאות *busy* יורד כאשר ב-*product* מופיעה התוצאה הסופית של המכפלה, כמו בסעיף הקודם, תעודות זהות שלנו הן: 213745953, 214008997. ומכפלתן היא 45743557014339141, וזאת אכן התוצאה שקיבלנו ב-*product* בסוף הריצה כאשר *busy* ירד ל-0.

לאחר מכן, ממתינים למשך זמן מחזור אחד ואז לאחר מחזור שעון נוסף אות ה-*start* עולה למשך מחזור שעון אחד ואז ממתינים עוד 3 מחזורי שעון לסיום חישוב המכפלה הקודמת. כעת, מציבים את ערכי *a*, *b* כאשר הבתים העליונים שלהם מאופסים, אות ה-*busy* עולה מיד עם ירידת אות ה-*start* והמכונה מתחילה את החישוב החדש. לאחר 3 מחזורי שעות מעליית *busy* נראה כי החישוב הסתיים והאות *busy* יורד כאשר ב-*product* מופיעה התוצאה הסופית של המכפלה. תעודות זהות שלנו הן: 213745953, 214008997, ולאחר איפוס הבית האחרון של תעודת הזהות הראשונה והמילה האחרונה של תעודת הזהות השנייה נקבל: 12682405, 33057. ומכפלתן היא 419242262085 וזאת אכן התוצאה שקיבלנו ב-*product* בסוף הריצה כאשר *busy* ירד ל-0.

