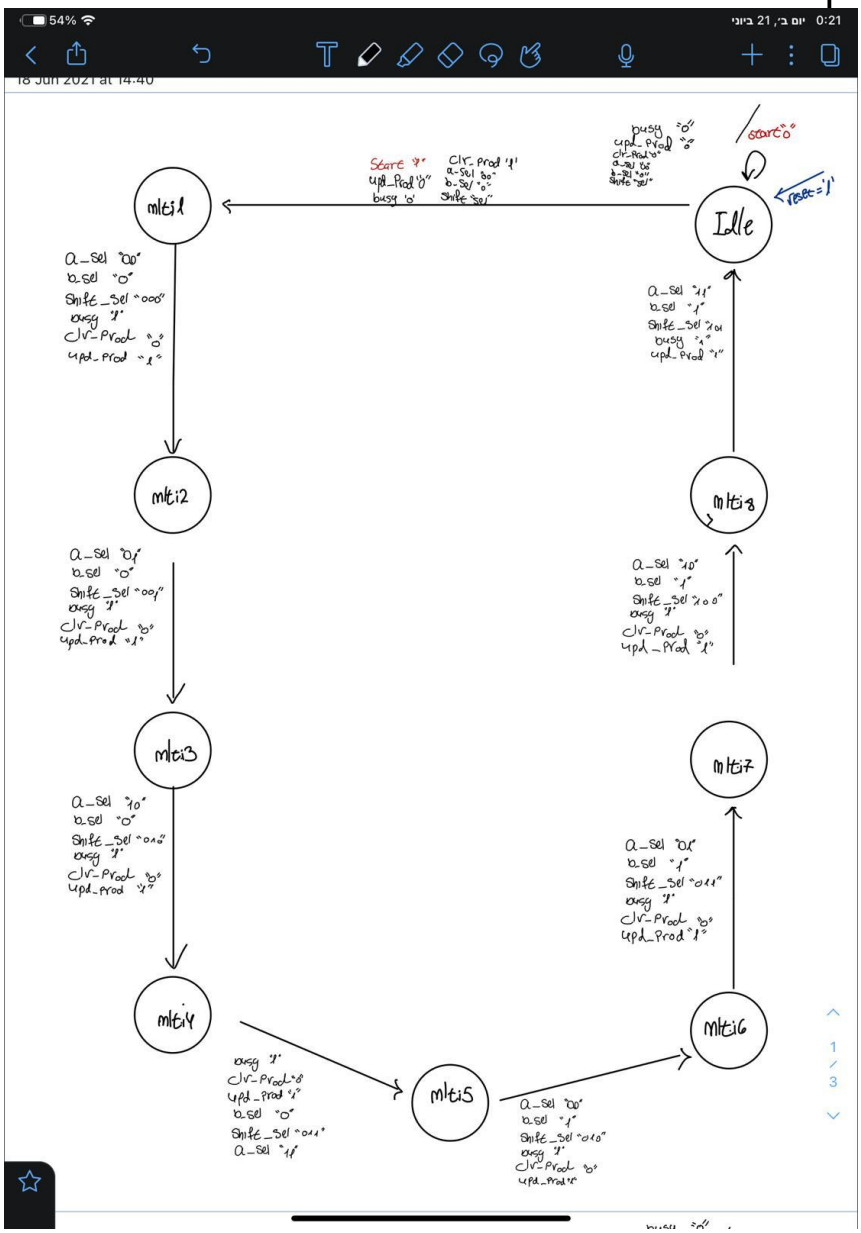
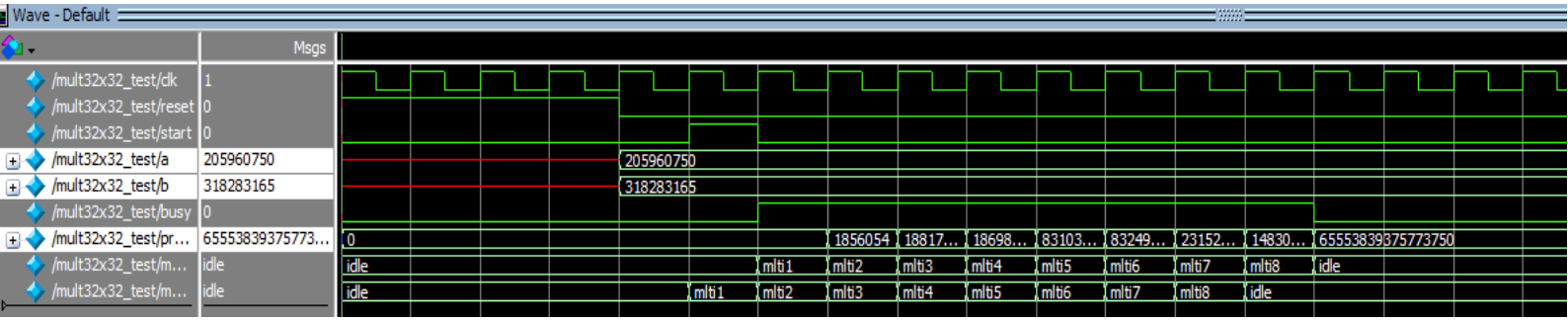
ספרתיות יבש 2

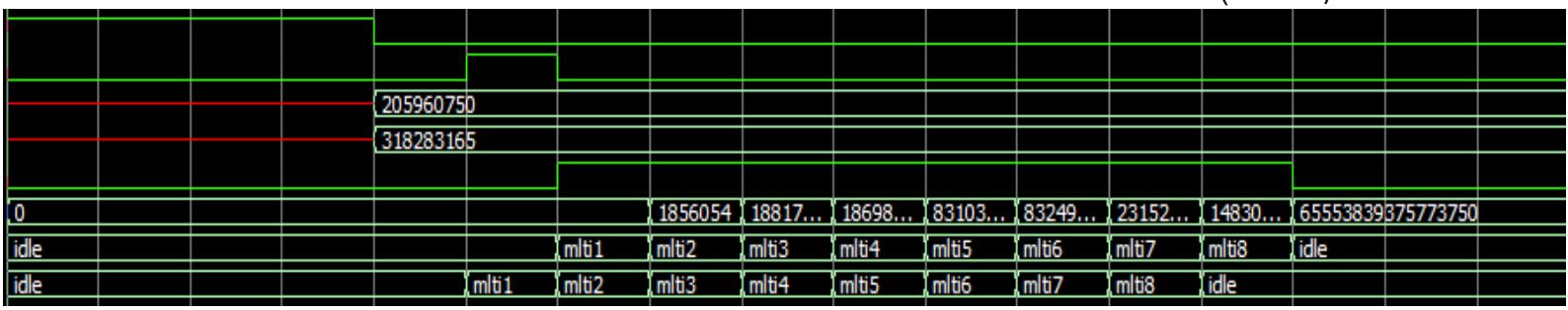
אלון ויינשל	318283165
אורי זוהר	205960750

2.1. תכננו מכונת מצבים (FSM) מסוג Mealy השולטת על פעולת הכפל בתכן הנתון בסרטוט





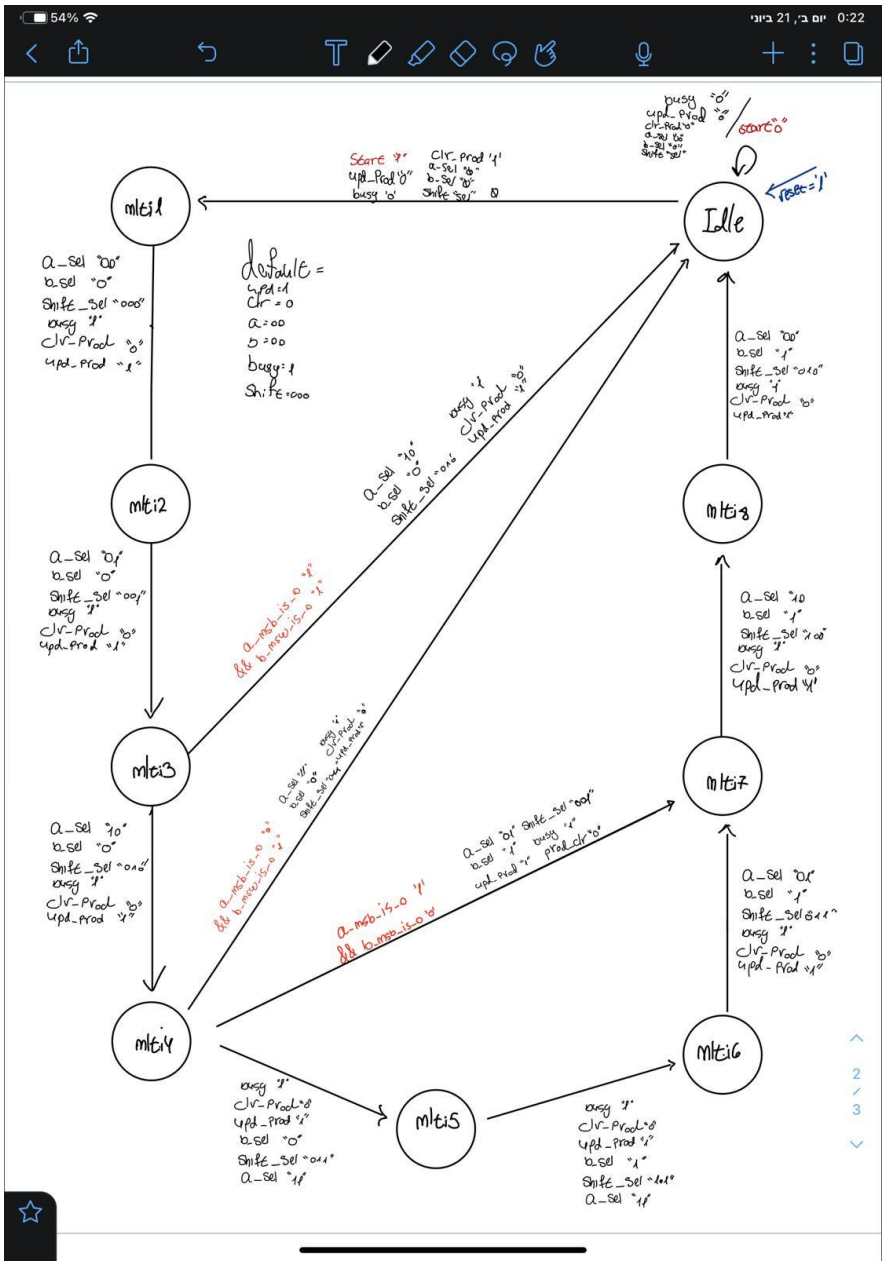
(zoom in)



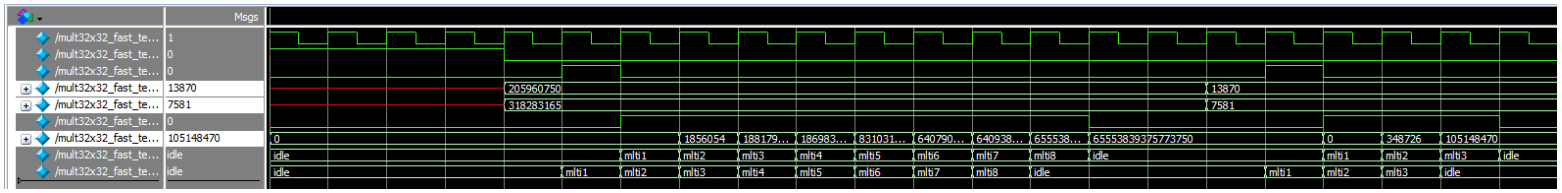
ראשית נשים לב מהדיאגרמה ניתן לראות שאורך מחזור של המערכת הינו 9 מחזורי שעון אותם נספור מרגע עליית הstart ועד לירידת הbusy בנוסף נשים לב שכמו שהתיכננו את המערכת הFsm עובר ממצב ההתחלתי idle לפי הסדר מ mlt1-mlt8 ולבסוף חוזרת אליו .

ובסיום תהליך החישוב בערך של product מופיע המכפלה של הערכים אותם הזנו (ת"ז) .

2.2 מימוש של מכפל מהיר יותר המנצל מצבי 0

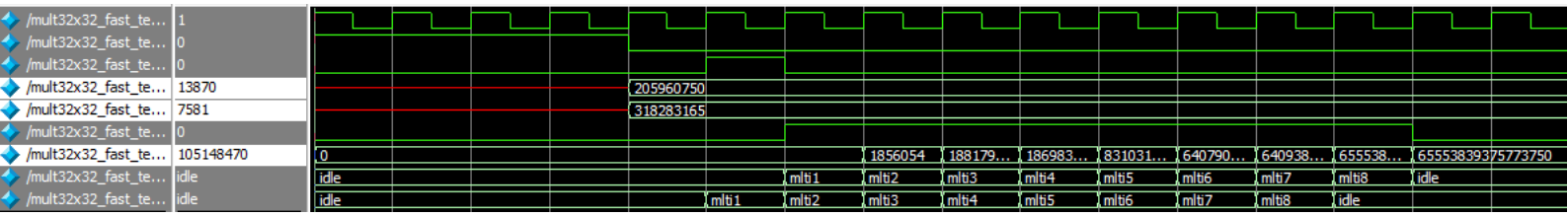


מספר מחזורי שעון	Msb a = 0	Msb b =0
3	1	1
4	0	1
6	1	0
8	0	0

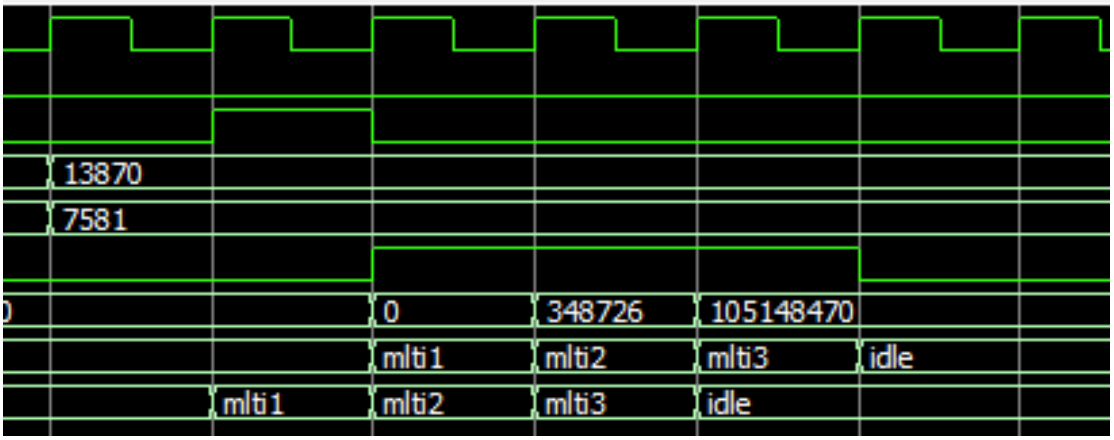


.

חלק 1 הזנת ת"ז הרגילות כמו לפני הזנת השינויים



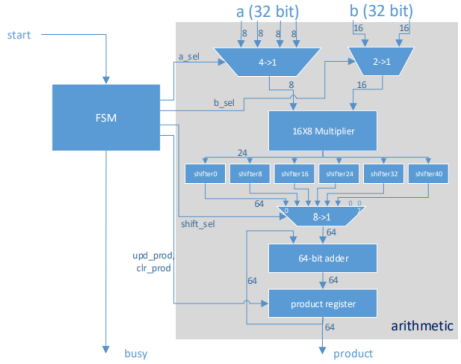
חלק 2 הזנת ת"ז עם אפסים בשני הבתים הראשונים



נשים לב שבחלק הראשון התוצאות יוצאות זהות לחלק הראשון (כי במצב זה ערכי תעודת הזהות ללא בתים מאופסים ולכן אין "דילוגים חכמים" כפי שתיכננו את המערכת .

בחלק 2 נשים לב שתחילה הערך של product מתאפס כנדרש לפני תחילתו של חישוב , בגלל שעבור מצב זה המערכת מבצעת פחות מחזורי שעון ניתן לראות שהפעם עשינו רק את המצבים mlti1-mlti3 ובגלל שבכל אחד מהמצבים יש 2 בתים מאופסים החישוב מסתיים לאחר 3 מחזורי שעון .

שאלה 2.3



האלגוריתם שלנו הוא כפל ארוך עם הזזות שמאלה (slli) .

בעצם נכפול זוג מספרים באורך 8N סיביות .

לצורת הנוחות את a כ - N מספרים באורך 8 סיביות

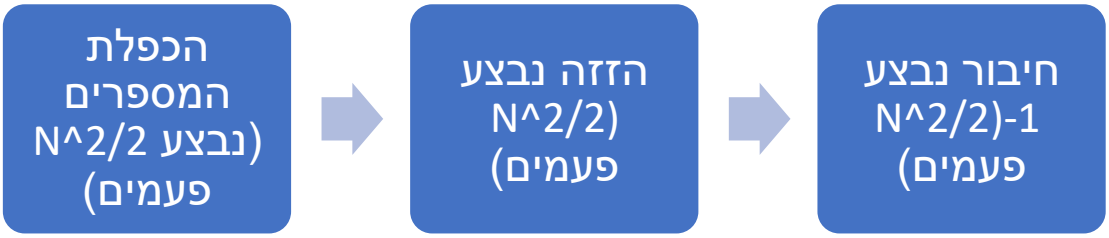
בצורה הבאה : $a_0, a_1, \dots a_N,$. (כל a_i מכיל 8 סיביות)

$a_0, a_1, \dots a_N,$.

ואת b נסמן כ - N/2 מספרים באורך 16 סיביות

ונסמנם $b_0, b_1, \dots b_{N/2}$.

נבצע הכפלה של כל בקומבינציות האפשריות כך שלכל a_i*b_j נבצע הזזה שמאלה של 8*(i+2*j) ביטים ולבסוף נסכום את כל המספרים .



לסיכום ניתן לומר שבקירוב טוב אנחנו עושים $3(\frac{n^2}{2})$ פעולות כלומר big-o notation = n^2

2.4. הקובץ mult 16X16 (הקובץ מצורף) זמן הריצה שלנו הוא 10 מחזורי שעון לכל חישוב נתון

```
.data
a: .word 0x0BAD
b: .word 0xFEED

.text
main:   # Load data from memory
        la      t3, a
        lw      t3, 0(t3)
        la      t4, b
        lw      t4, 0(t4)

        # t6 will contain the result
        add     t6, x0, x0

        # Mask for 16x8=24 multiply
        ori     t0, x0, 0xff
        slli    t0, t0, 8
        ori     t0, t0, 0xff
        slli    t0, t0, 8
        ori     t0, t0, 0xff
####################
# Start of your code

# Use the code below for 16x8 multiplication
#   mul     <PROD>, <FACTOR1>, <FACTOR2>
#   and     <PROD>, <PROD>, t0
add    t1, x0,t3 #t1=a
andi t1,t1,0xff #t1=a[0,7]
add    t2, x0,t3 #t2=a
srli t2,t2,8 #t2=a[8,15]
mul t1, t1, t4
and t1, t1, t0
mul t2, t2, t4
and t2, t2, t0
slli t2, t2, 8
add t6,t1,t2
# End of your code
####################

finish: addi    a0, x0, 1
        addi    a1, t6, 0
        ecall # print integer ecall
        addi    a0, x0, 10
        ecall # terminate ecall
```

10

Editor | Simulator

Run | Step | Prev | Reset | Dump

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
0x0ff2e293	ori x5 x5 255	ori t0, t0, 0xff
0x00829293	slli x5 x5 8	slli t0, t0, 8
0x0ff2e293	ori x5 x5 255	ori t0, t0, 0xff
0x01c00333	add x6 x0 x28	add t1, x0,t3 #t1=a

195065129

s4 (x20)	0x00000000
s5 (x21)	0x00000000
s6 (x22)	0x00000000
s7 (x23)	0x00000000
s8 (x24)	0x00000000
s9 (x25)	0x00000000
s10 (x26)	0x00000000
s11 (x27)	0x00000000
t3 (x28)	0x00000bad
t4 (x29)	0x0000feed
t5 (x30)	0x00000000
t6 (x31)	0x0ba07529

Display Settings: Hex

2.5 הקובץ mult 16X16 אחרי שיפורים של msb/msw=0

```
.data
a: .word 0x00AD
b: .word 0x0000

.text
main:   # Load data from memory
        la      t3, a
        lw      t3, 0(t3)
        la      t4, b
        lw      t4, 0(t4)

        # t6 will contain the result
        add     t6, x0, x0

        # Mask for 16x8=24 multiply
        ori     t0, x0, 0xff
        slli    t0, t0, 8
        ori     t0, t0, 0xff
        slli    t0, t0, 8
        ori     t0, t0, 0xff

####################
# Start of your code

# Use the code below for 16x8 multiplication
#   mul     <PROD>, <FACTOR1>, <FACTOR2>
#   and     <PROD>, <PROD>, t0
beq t4,x0,finish
add    t2, x0,t3 #t2=a
srli t2,t2,8 #t2=a[8,15]
beq t2,x0,MLT
mul t2, t2, t4
and t2, t2, t0
slli t2, t2, 8
MLT:
add    t1, x0,t3 #t1=a
andi t1,t1,0xff #t1=a[0,7]
mul t1, t1, t4
and t1, t1, t0
add t6,t1,t2
# End of your code
####################

finish: addi    a0, x0, 1
        addi    a1, t6, 0
        ecall # print integer ecall
        addi    a0, x0, 10
        ecall # terminate ecall
```

הסבר) אם b שונה מאפס אז כופל את 8 הסיביות האחרונות (LSB) של a ב b

שינוי 1)

מטפל במקרה שבו 16 הסיביות של b אפסים אז קופץ ישירות לסוף התוכנית 0*X=0

שינוי 2)

במידה ו8 הסיביות הראשונות של a כולן אפסים עובר לסוף תהליך ההכפלה ושומר את הערך כי הוא שונה מאפס

כעת נדון בשאלה האם שינוי זה היה משתלם או לא נחלק למקרים

Second code (run time)	First code (run time)	Msb a = 0	B=0
12	10	0	0
9	10	1	0
1	10	0	1
1	10	1	1

כפי שניתן לראות במקרי הקצה שינוי זה היה משתלם בשאר המקרים (להערכתי הרוב) שינוי זה לא ישתלם.