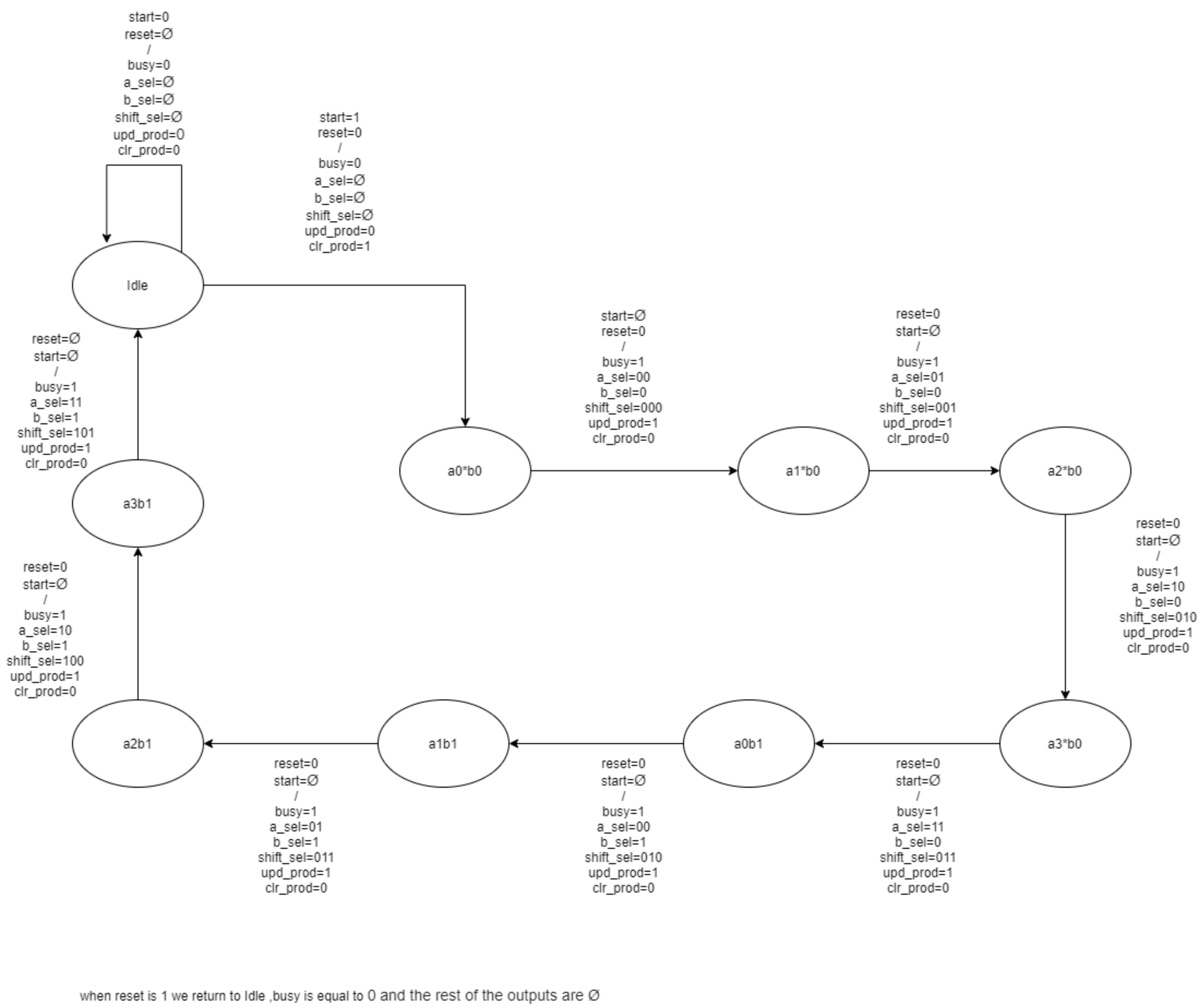
תרגיל בית 2

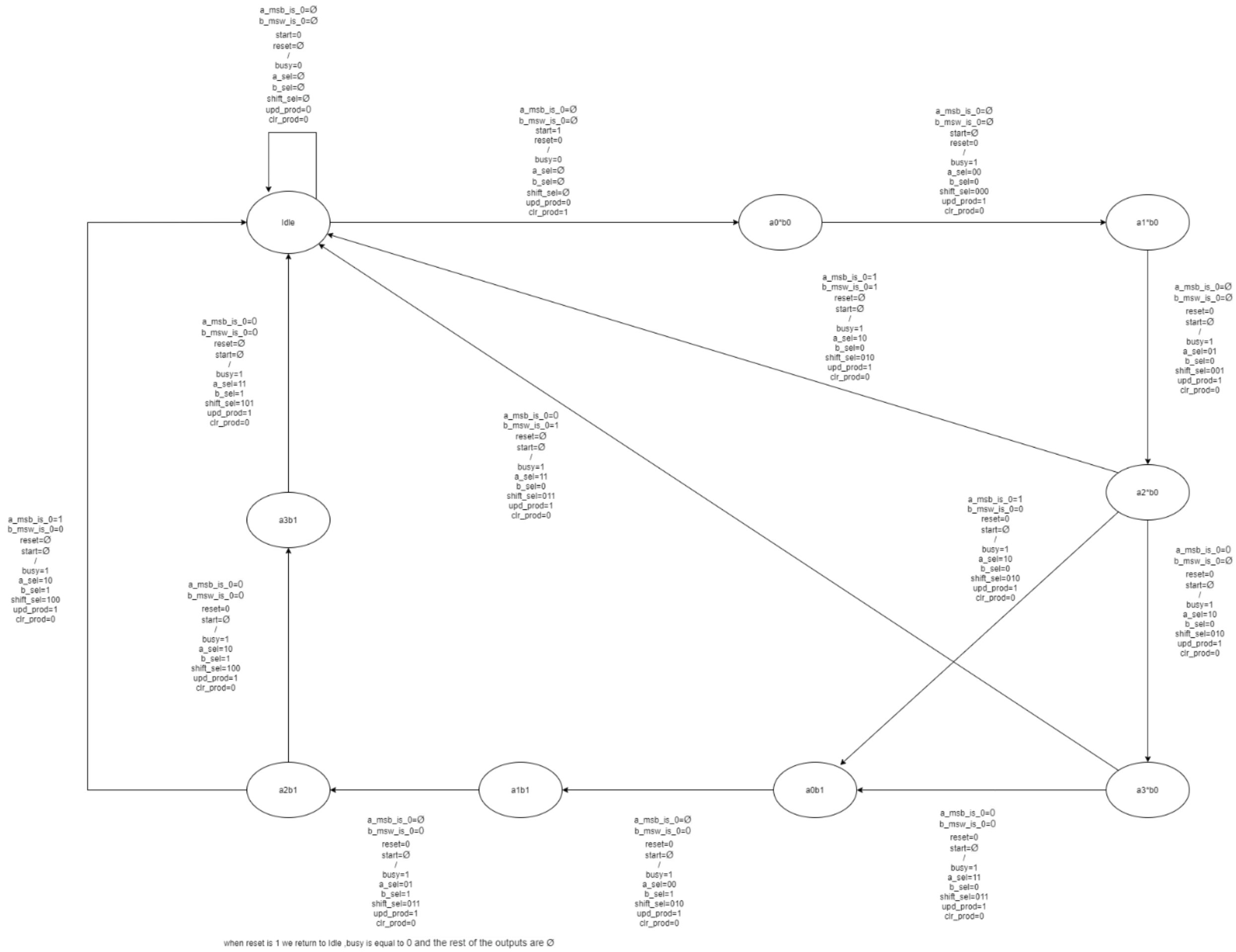
מוחמד חוג'ירי	206950149
רדואן עתמה	207363151

:2.1



פעולת הכפל לוקחת 9 מחזורי שעון

:2.2



מחזורי השעון:

a=0,b=0: 9 מחזורי שעון כמו קודם

a=1,b=0: 7 מחזורי שעון

a=0,b=1: 5 מחזורי שעון

a=1,b=1: 4 מחזורי שעון

ניתן לראות המצב הכי מהיר הוא כאשר a=1,b=1

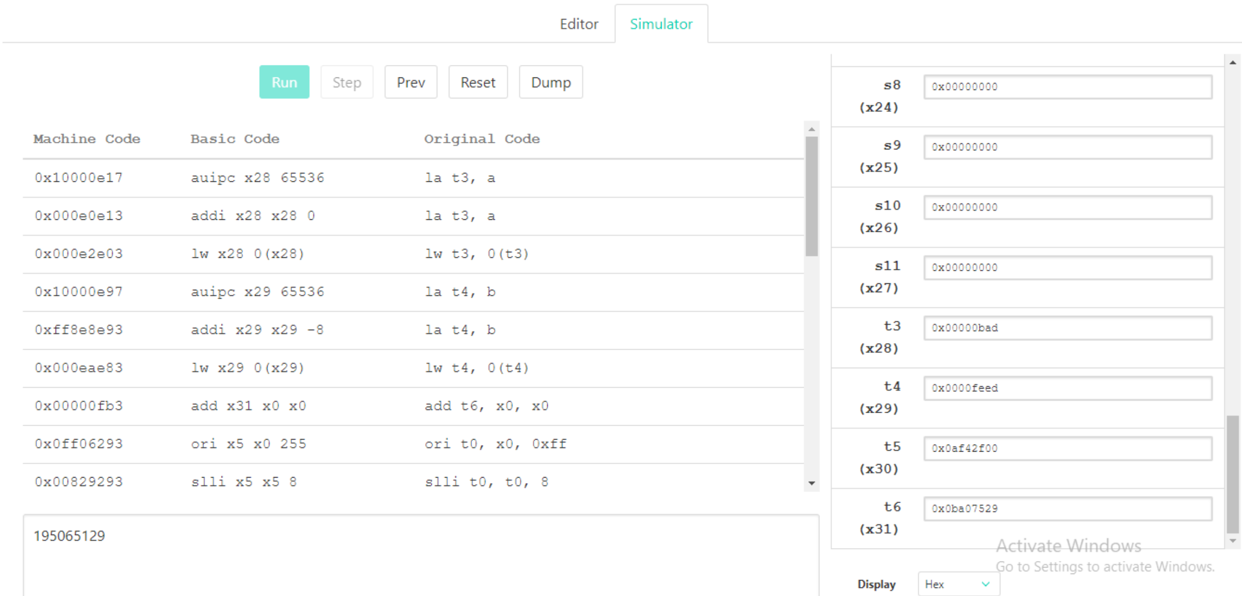
:2.3

divide a and b to 16 bit chunks ($a_1 a_2 \ldots a_{N/2}$), ($b_1 b_2 \ldots b_{N/2}$)

```
for(i=0;i<N/2;i++){
	for(j=0;j<N/2;j++){
		temp_sum=Mult_16X16(a_j +b_i ) # 16X16 multiplier,we wrote the function down below
		shift_sum=shifter(i+j,temp_sum) # we shift the number by i+j bits
		sum=sum+shift_sum
	}
}
return sum


Mult_16X16(a,b){
	b0=b[7:0]
	b1=b[15:8]
	Sum=Mult_16X8(a , b0) # we use 16X8 multiplier given in the question
	Sum=sum+shifter(8 , Mult_16X8(a,b1)) # shift the output by 8 bits and add to the original sum
	Return Sum
}
```

:2.4



פעולת הכפל לוקחת 9 מחזורי שעון

:2.5

נבדוק אם a or b אחד מהם החלק העליון שלו מתאפס אז קופצים ישר לקטע הקוד הבא

```
mul        t6, t4, t3
and        t6, t6, t0
```

ואחר כך ממשיכים לסוף הקוד

הבעיה בשיטה הזו שכדי לבדוק ש $a_{msb}=0$ or/and $b_{msb}=0$ צריכים לפחות 6 פקודות ופקודת jump בסוף ולכן

במקרה ש $a_{msb}\neq 0$ and $b_{msb}\neq 0$:

זמן הריצה עלה מ 9 מחזורי שעון ל 16=9+6+1

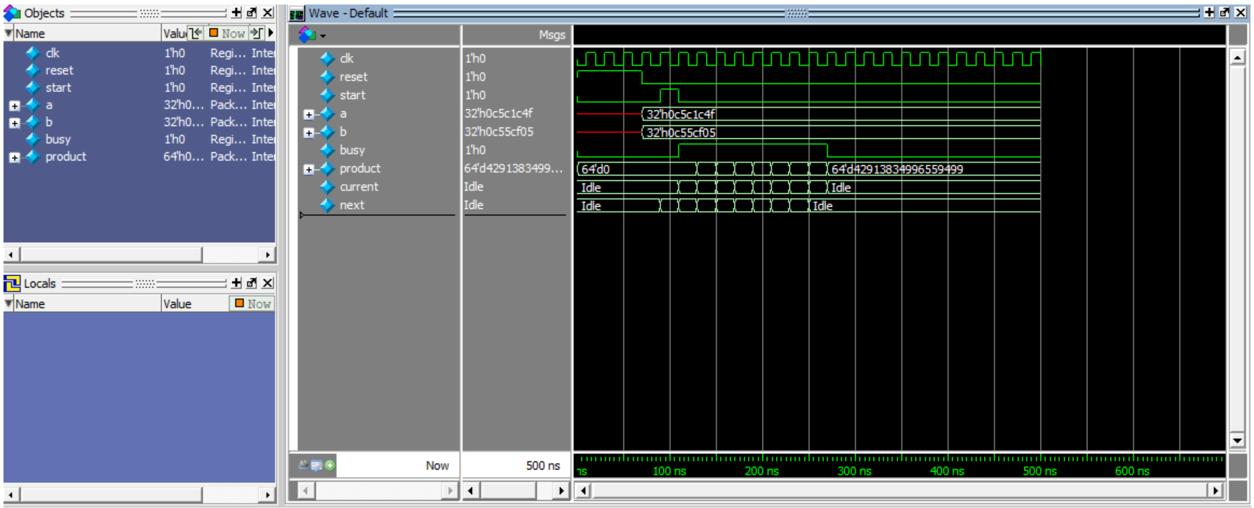
במקרה ש $a_{msb}=0$ or/and $b_{msb}=0$:9

זמן הריצה ירד מ 9 מחזורי שעון ל 8 מחזורי שעון לכל היותר

ניתן לראות כי זה לא משתלם

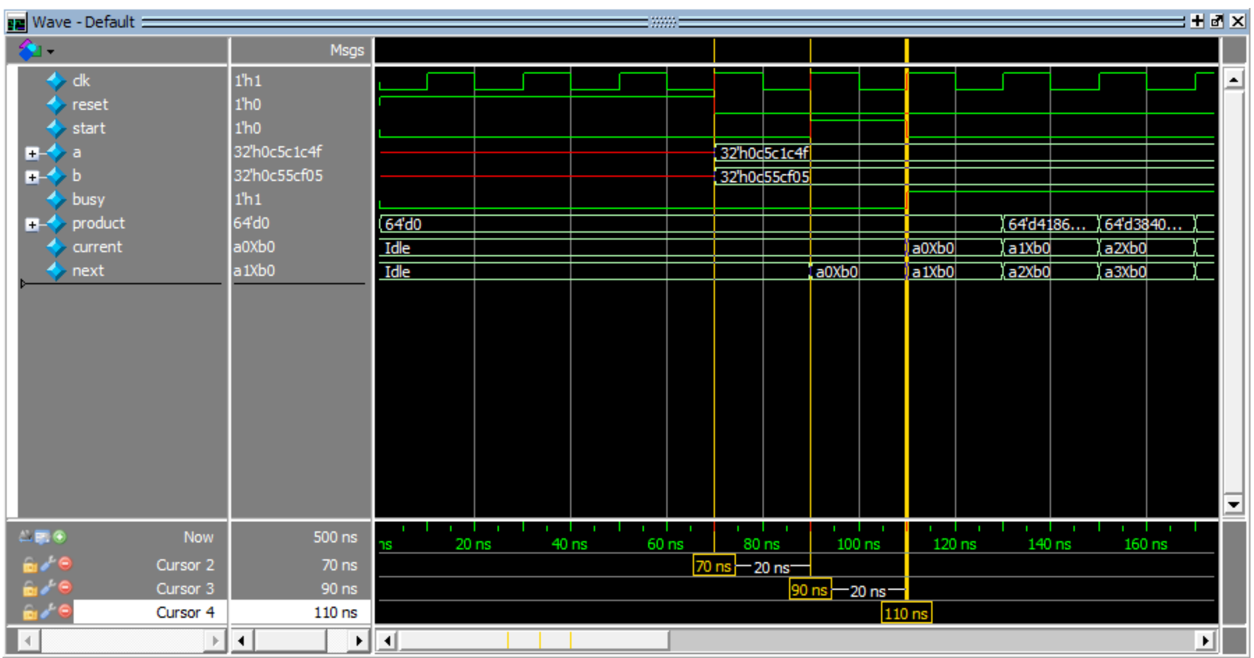
Wave Diagrams:

3.4:



For this simulation we changed the clk every 10ns so a full clock cycle is 20ns

We ran the simulation for 500 ns and inserted our id numbers in



We started the simulation with the following values:

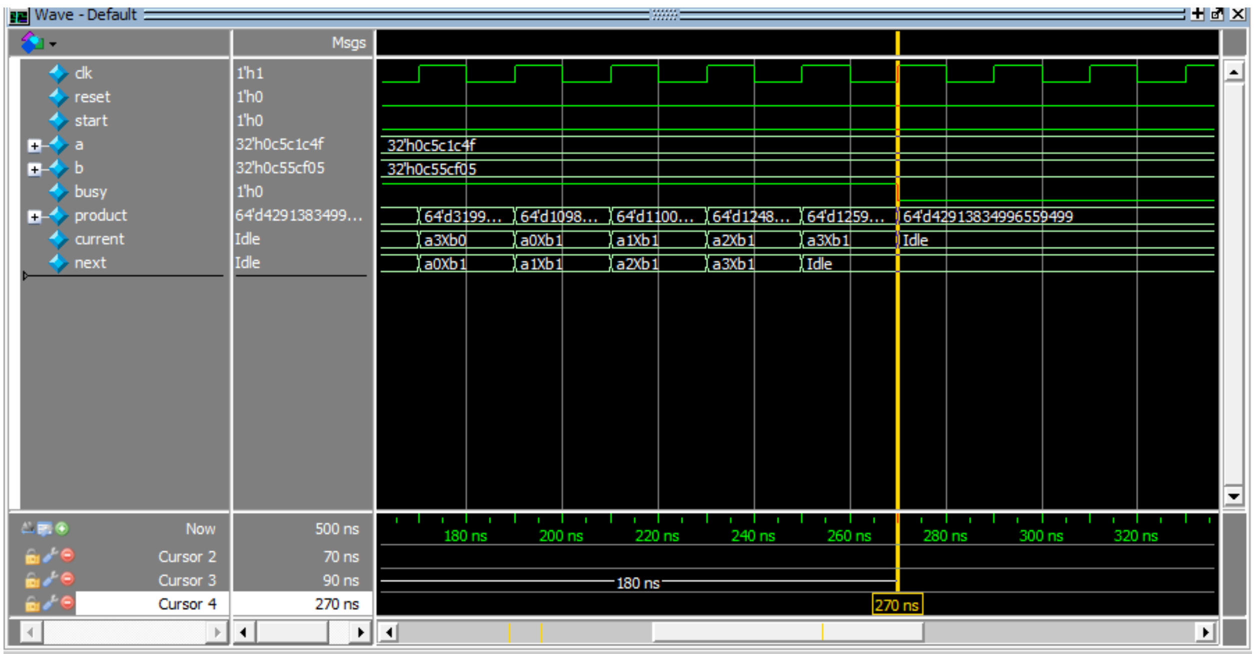
Clk=0 , reset=1, start=0

Since reset is equal to 1 we start with the Idle state, and since our default value for clr_product is 0 the value of product is set to 0 as well

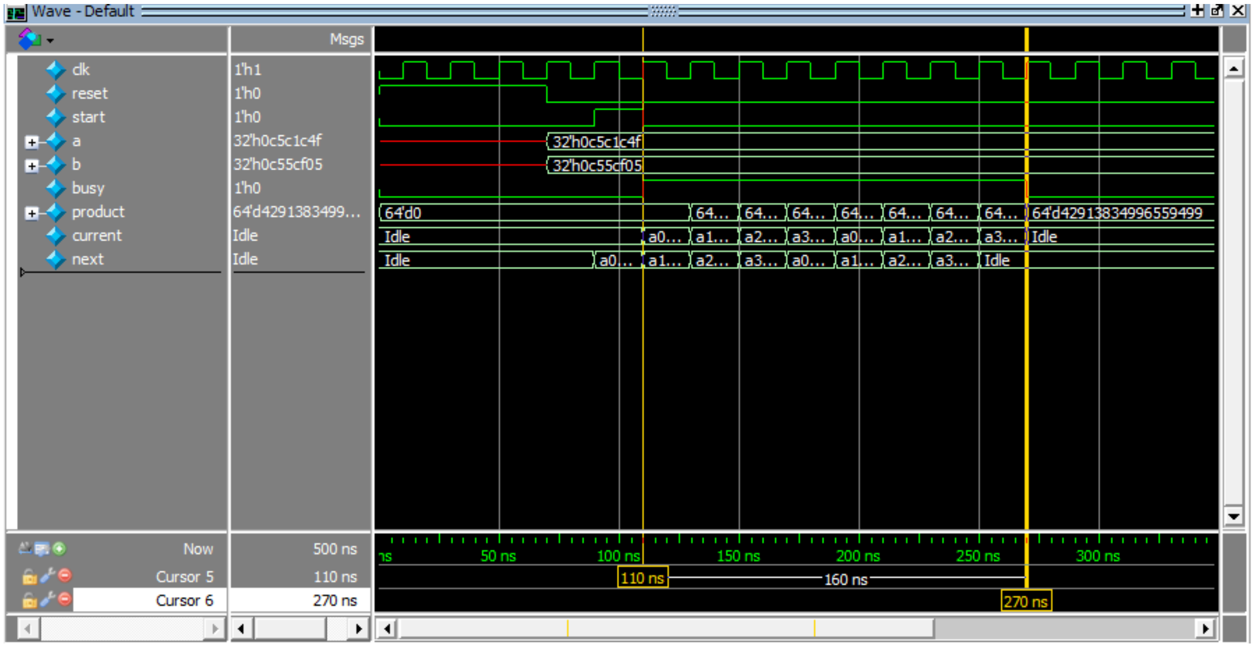
After 4 clock cycles we set reset to 0 and inserted our id numbers in a and b, we see that the machine has not started working sinced we didn't give the signal to start yet(start = 1)

A clock cycle after that we changed the value of start to 1 and so the machine started its work, the next state changed to a0Xb0 and a clock cycle after that busy value changed to 1 indicating that we started working and the machine started jumping between states

At the end of each state the machine updates its product value (ex. 130 ns after we exit a0Xb0) and continues to do so until we do a full circle and finish back again in the Idle state with the number fully calculated

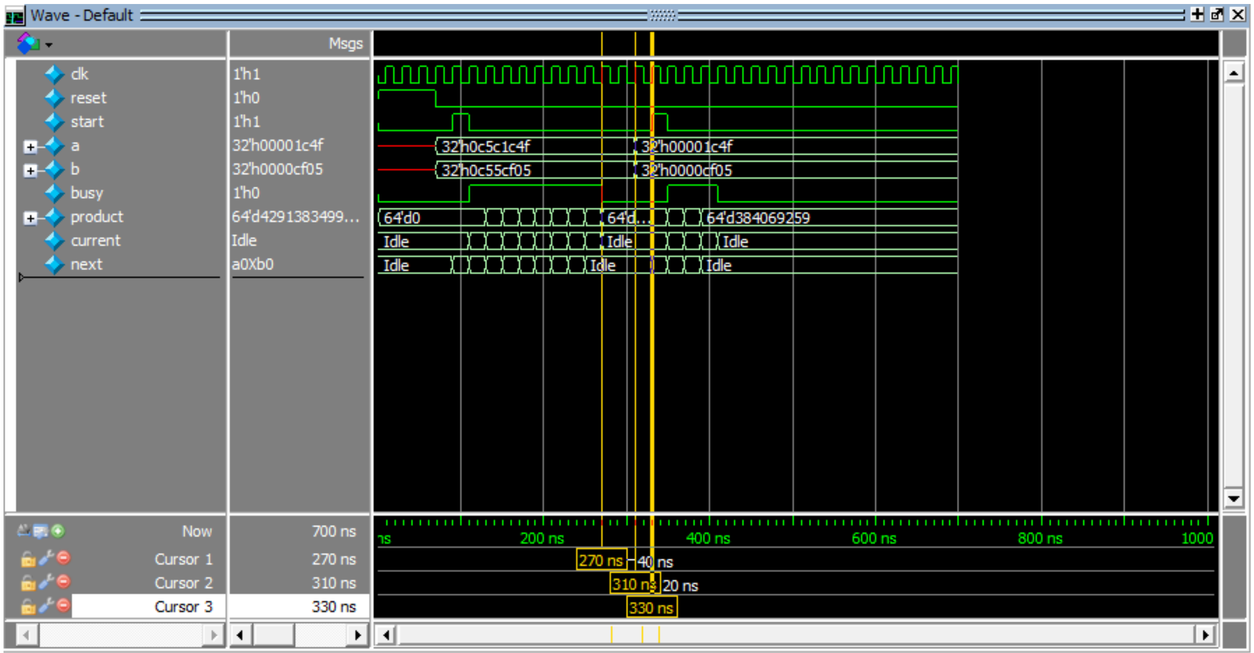


And here we have the number fully calculated and matching to the result of our calculator

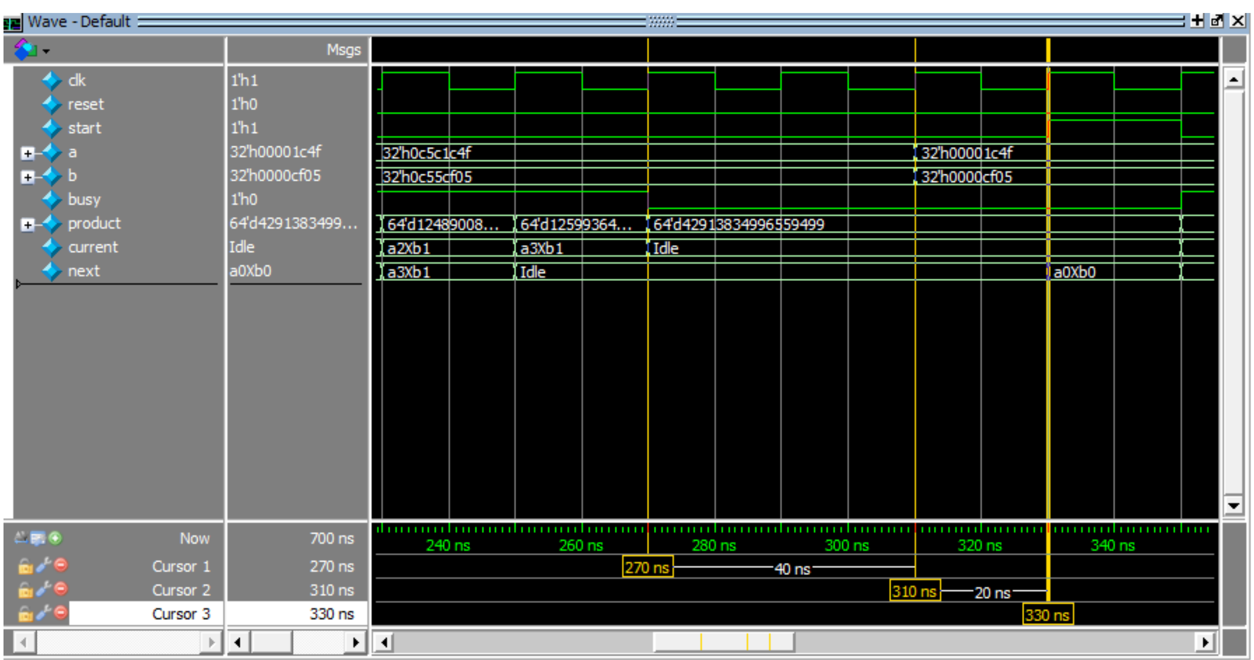


We can see here that the machine switches 9 different states in 9 clock cycles to finish the calculation

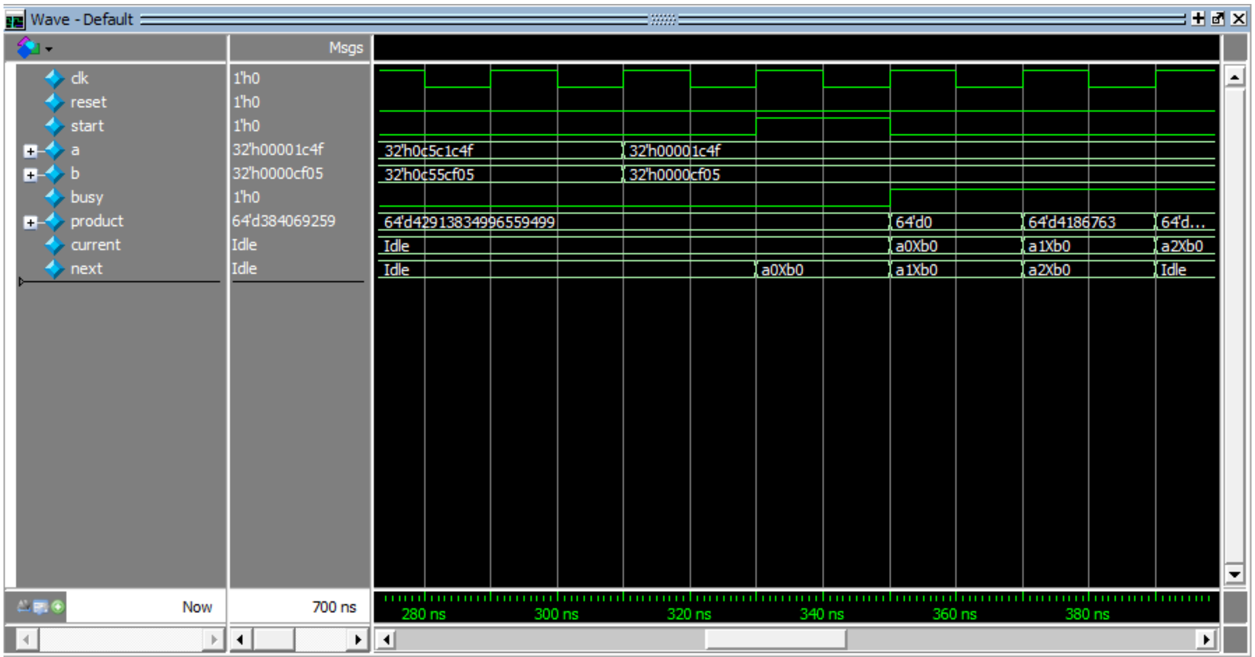
3.7:



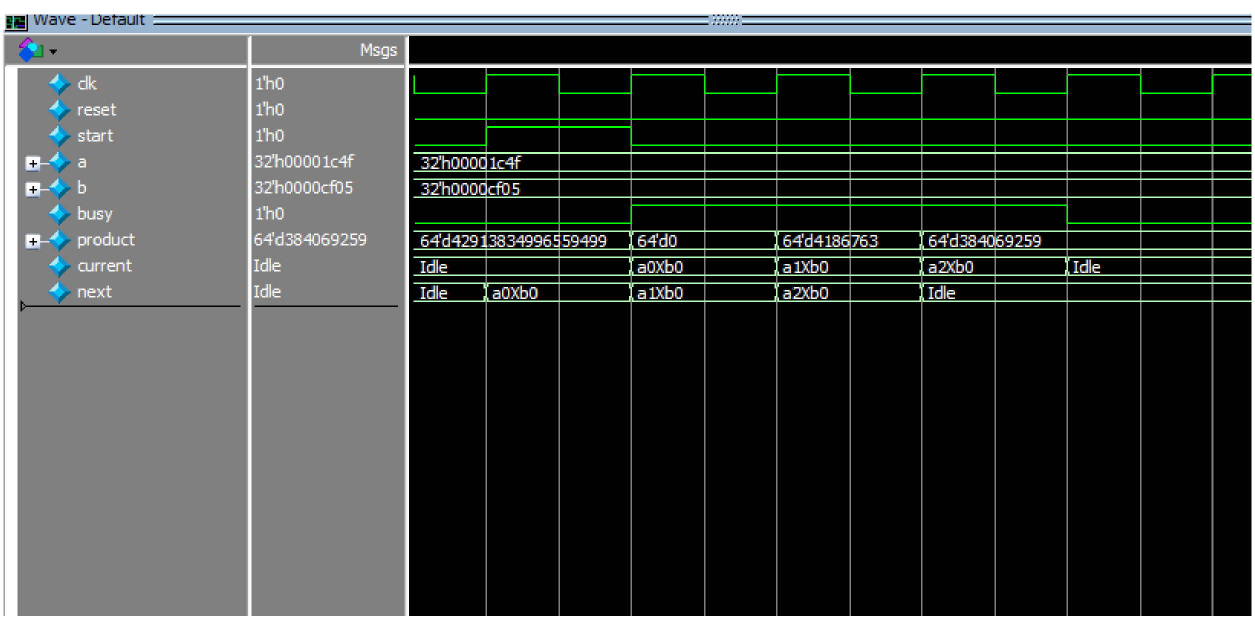
We can see that this diagram is identical to the previous diagram until the 270 ns point



At 310ns we changed the values of a and b by changing the value of the two most significant bytes to 0, a clock cycle after that we gave the signal to start by changing the value of start to 1, and so the machine started changing states



At 330 ns at the rest of clock cycle we set start to 1, a clock cycle after that the machine deleted its previous calculations and entered into the next state (a0Xb0) to start a new calculation



We can see here that the calculations go through 4 different states

In the last state the calculation result stays the same since we set the 2 most significant bytes to zero meaning a2 is 0

(a0Xb0->a1Xb0->a3Xb0->Idle) meaning it takes 4 clock cycles to do the calculations which matches our answer in 2.2