שיעור #2

כפל ב-FIXED POINT

נדון בכפל UNSIGNED

$$X=[\underbrace{X_{n-1},X_{n-2},\dots,X_1,X_0}_{n \text{ ביטים}}]$$

$$Y=[Y_{n-1}, \qquad ,Y_0]$$

$$X=\sum_{i=0}^{n-1} X_i\cdot 2^i \qquad Y=\sum_{i=0}^{n-1} Y_i\cdot 2^i$$

$$Z=X\cdot Y=\left(\sum_{i=0}^{n-1} X_i\cdot 2^i\right)\cdot Y=\sum_{i=0}^{n-1}\left(Y\cdot 2^i\right)\cdot X_i$$

כלומר נבצע n חיבורים כאשר בשלב ה-i נקח את $Y\cdot 2^i$ כלומר Y המוזז שמאלה i מקומות (אם "1"=X_i, אם "0"=X_i נחבר 0):

לדוגמא נבצע כפל של $X=[1101]$ ו-$Y=[1011]$:

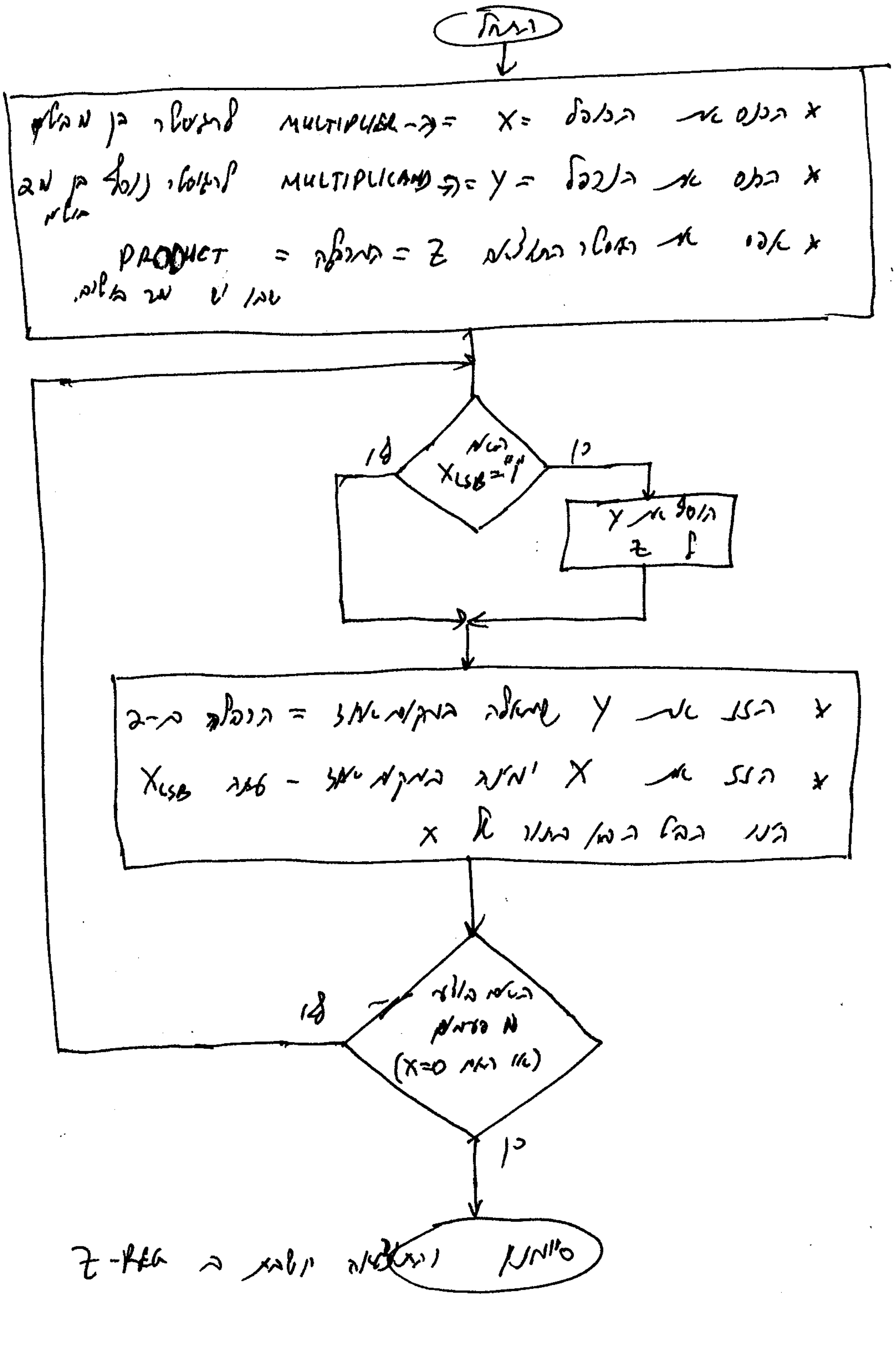
	7	6	5	4	3	2	1	0	
					1	0	1	1	$=Y\cdot 2^0\cdot X_0=Y\cdot 2^0\cdot 1$
+				0	0	0	0		$=Y\cdot 2^1\cdot X_1=Y\cdot 2^1\cdot 0$
			1	0	1	1			$=Y\cdot 2^2\cdot X_2=Y\cdot 2^2\cdot 1$
		1	0	1	1				$=Y\cdot 2^3\cdot X_3=Y\cdot 2^3\cdot 1$
$(143)_{10}=$	1	0	0	0	1	1	1	1	

ואכן $X=13$ $Y=11$ ו- $143=11\times 13$

כמה ביטים נדרשים לתוצאה?

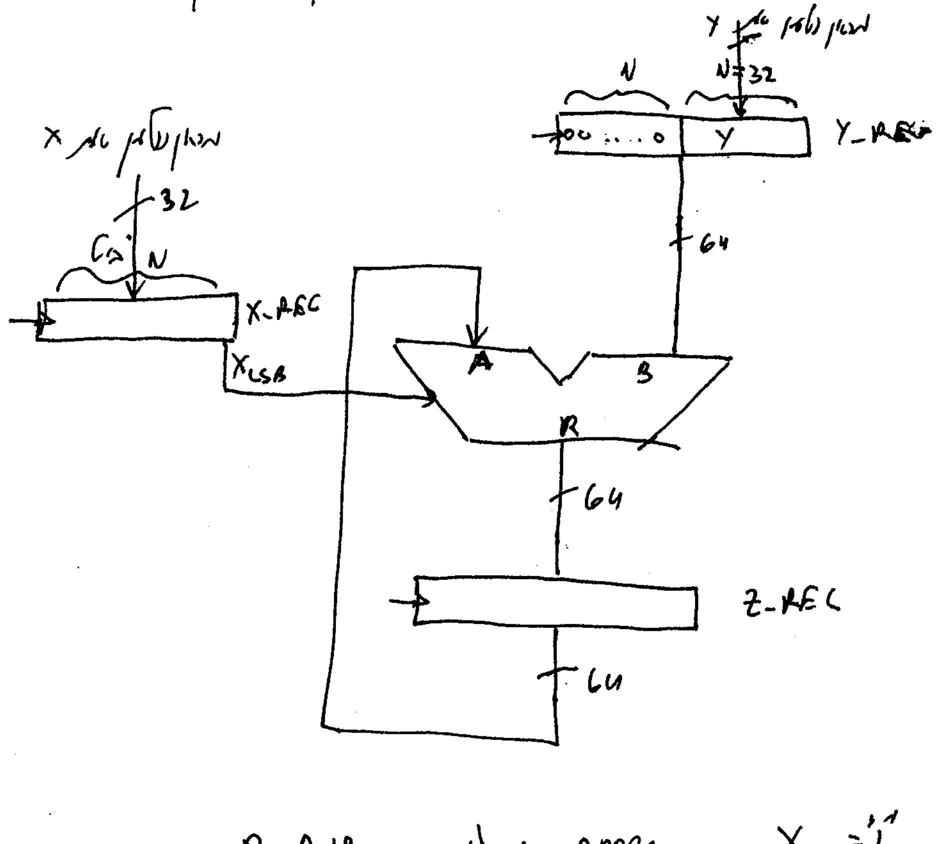
$$Z_{max}=X_{max}\cdot Y_{max}=(2^n-1)\cdot(2^n-1)=2^{2n}-2^{n+1}+1 \Rightarrow$$ נדרשים 2n ביטים

כרצוננו לבדוק את האלגוריתם שמצאנו למימוש חומרה שניתן להכניס ל MIPS. בשלב שלנו נפרט הפעולות המשתתפות בתרשים הזרימה הבא!



כדי לממש את האלגוריתם נשתמש ברגיסטר אחד X
N ביטים (למשל 32 ביט) שמור X המספר הראשון שלנו
הכפלה ויוזז, כמו כן נשתמש ברגיסטר Z מגודל 2N (=64)
שמור Y כאשר זה מוזז הכפלה שמאלה, ולבסוף Z נשתמש
ברגיסטר "הסכום" Z מגודל 2N. נשתמש גם ב-ALU
Z מגודל 2N (=64) ביט. תפקידו של הבית X_{LSB} הוא לקבוע
אם שווה 1 בהתאם הוא מבצע $Z=Z+Y$
אחרת מבצע $Z=Z+0$,

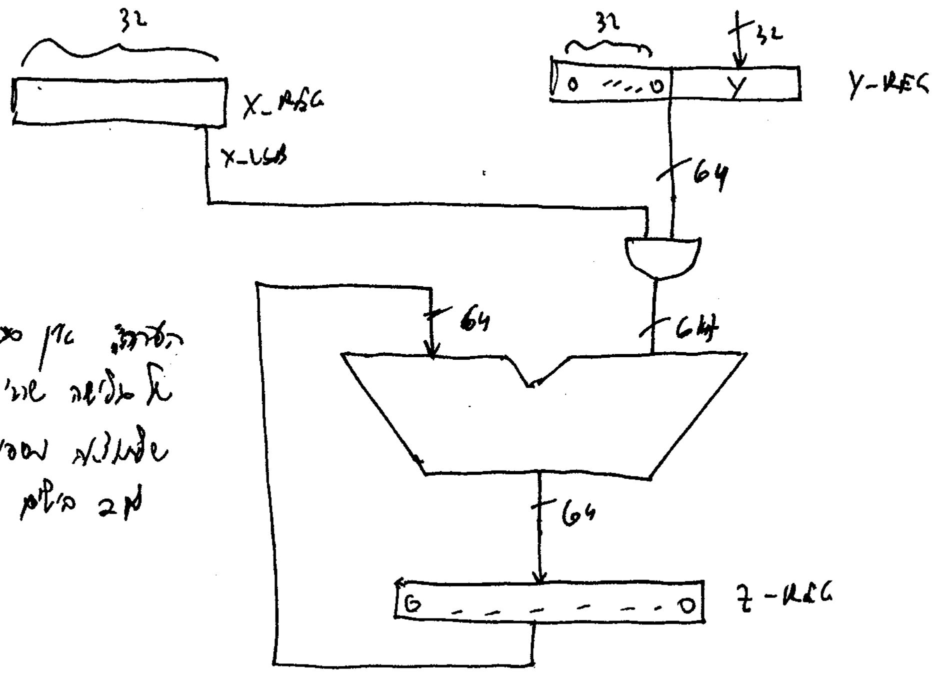
אפשרות אחת לממש המכפל הינה לפיכך כדלקמן:



אם X_{LSB}='1' ה-ADDER מבצע $R=A+B$

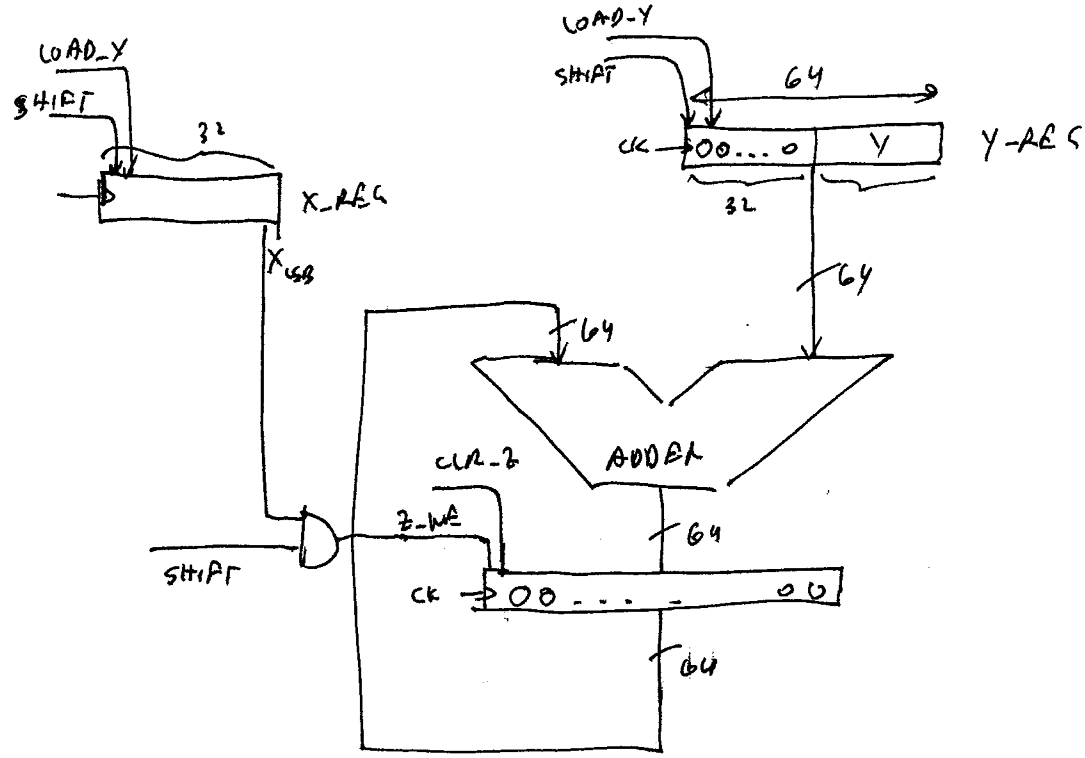
אם X_{LSB}='0' ה-ADDER מבצע $R=A$

במקרה כזה המעגל הוא המעגל בו X_{LSB} מבצע AND
עם Y (= מוכפל ב-X):



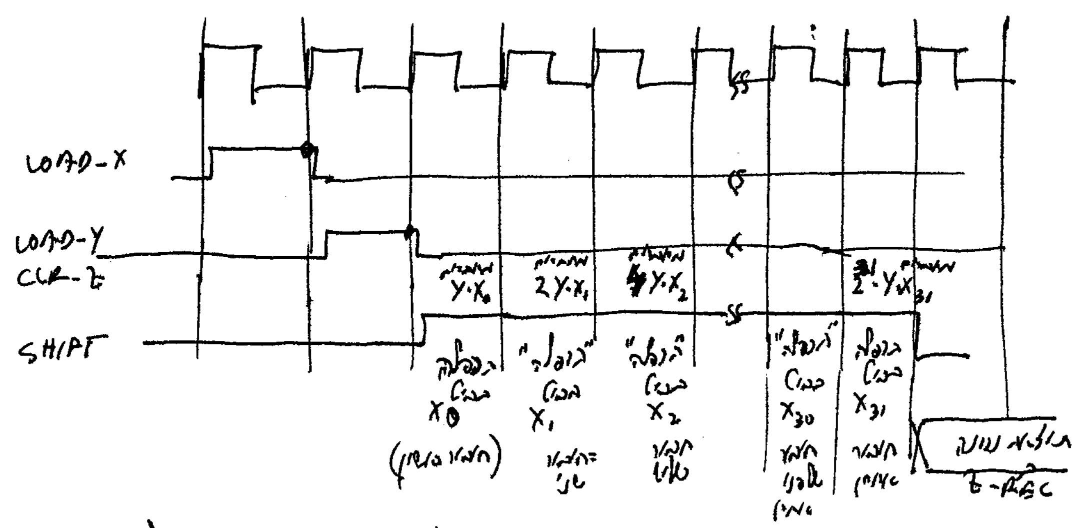
הערה: אין סכנה
של גלישה שכן הוכחנו
שהתוצאה מספיקה
2N ביטים

אפשרות נוספת היא לשלוט על כתיבה לרגיסטר Z
ע"י X_{LSB}. כלומר אם X_{LSB}="1" נכתוב לרגיסטר Z
את תוצאת ה-ALU ואם X_{LSB}="0", פשוט לא נכתוב
(= לא נותן WE. אז ה-DATA נשאר ב-Z ללא שינוי)
[כלומר ה-DATA נשאר ב-Z ללא שינוי זה בעצם שווה ערך
לביצוע Z=Z+0].



* במצב בסיס קודם לטעון את X ואחרי את Y וכן לאפס את Z. אחרי כל מחזור שעון מחשבים Y+Z וכאשר X_{LSB}="1" מכניסים התוצאה ל-Z. אם X_{LSB}="0" פשוט לא מכניסים כלום (Z נשאר ללא שינוי)

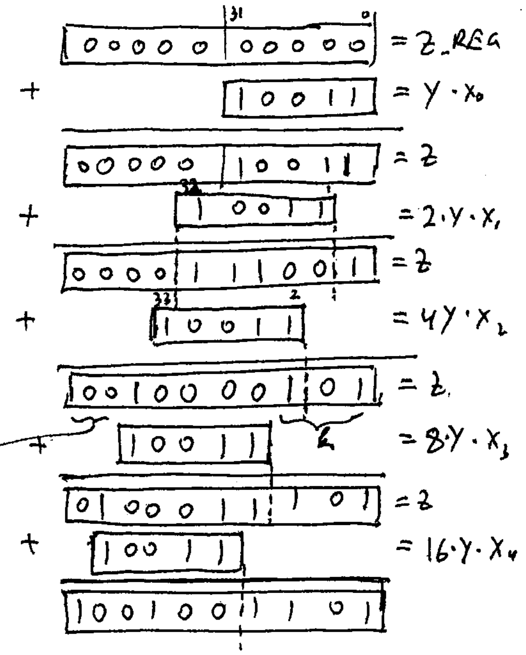
* כמו כן בכל מחזור שעון אנו מזיזים את Y שמאלה ביט (=הכפלה ב-2) ואת X ימינה ב-1 (כדי לקבל X_{LSB} חדש בכל פעם של השעון)



(כמובן שאין משמעות לסדר טעינת X ו Y ואיפוס Z במקביל)

נסתכל שוב על הדוגמא הקודמת כשכל ה-x_i-ים הם 1-ים ו-$n=5$:

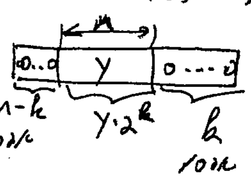
$$Y\cdot X=\underbrace{[10011]}_{(19)_{10}}\times\underbrace{[11111]}_{(31)_{10}}$$



$\leftarrow (589)_{10}=(19)_{10}\times(31)_{10}$

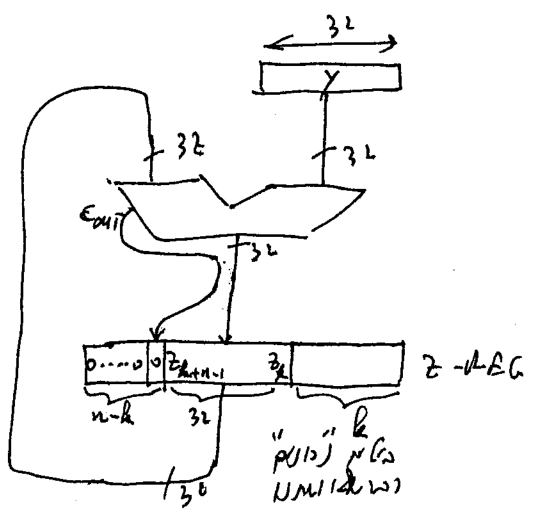
כפי שרואים מהדוגמא בעצם להשתמש ב ALU של $2n=64$ ביט זה בזבוז. כל שלב חיברנו מספר של n ביטים למספר של n ביטים וקבלנו לכל היותר $n+1$ ביטים. כמו כן בכל ב-$k-1$ ביטים כשעובר שלב ה-k אני בעצם ב-Y_REG

יש לי:



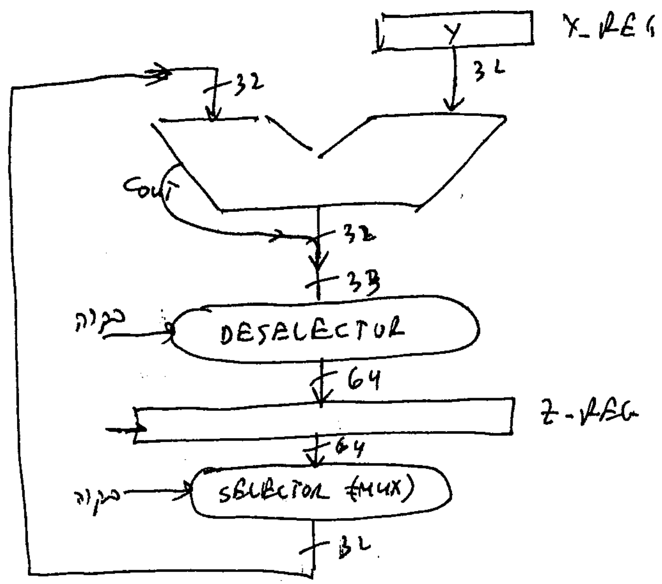
אשר מחוברים רק על n

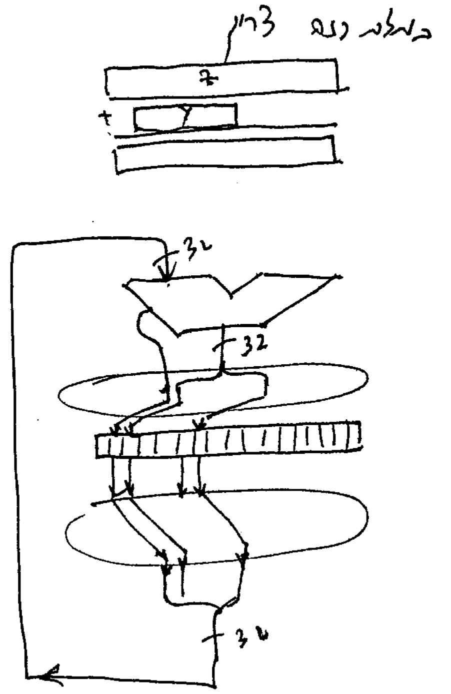
הביטים מכוונים. k עד $n+k-1$ של z עם Y. (המקורי המעוגל)

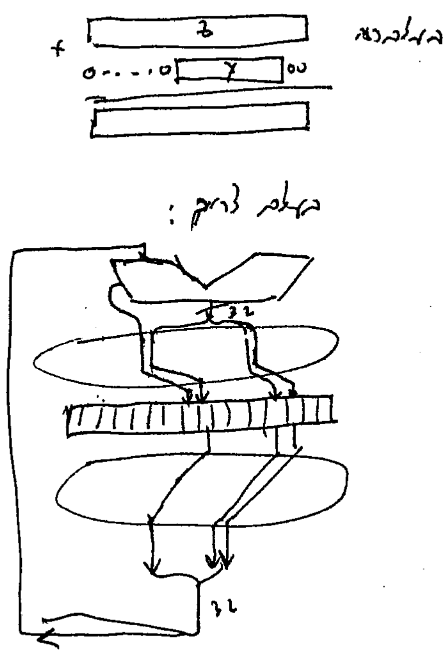


בעצם בכל שלב מספיק לעשות כך:

ואיך נממש משהו כזה? צריך איזשהי לוגיקה קומבינטורית
שתוציא את 32 הביטים הרלוונטים של z $[z_{k+n-1} \dots z_{k+1}, z_k]$
לכניסת ה-ADDER, ואת $33=32+1$ ביטי יציאת ה-ADDER
לביטים המתאימים של z $[z_{k+n}, \dots, z_k]$:



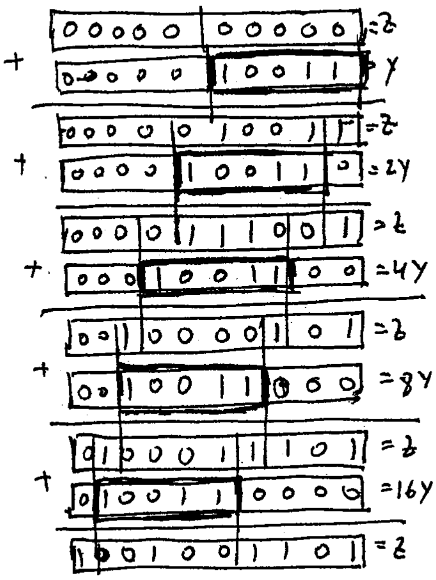




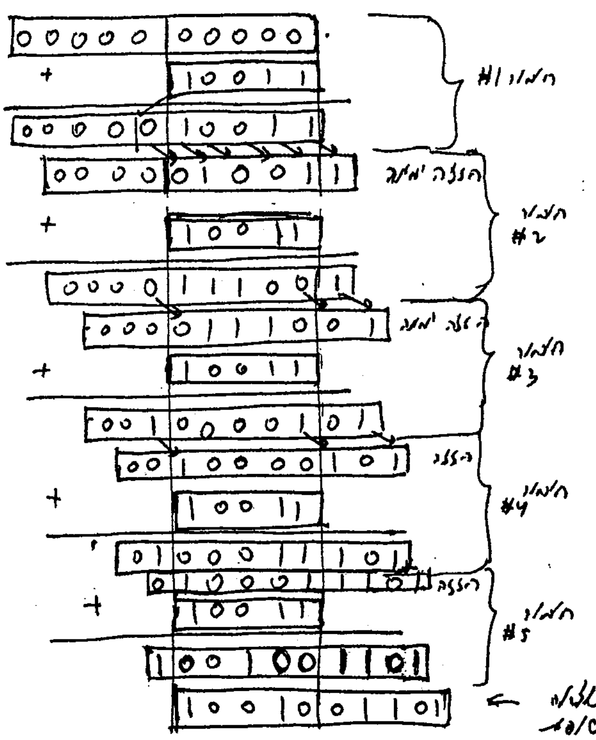
בפועל אנו חוזרים על כל n (או n+1) ביטים שווים ב-2n=64
הביטים של Z ומחפשים אותם לכניסות ויציאות ה-~~ADDER~~.

אבל בעצם במקום להזיז את "כל המספר" ימינה ל-Z אפשר גם
לעשות הפוך = להזיז את המספר ב Z ימינה לפני החיבור.
אם נבצע את זה נוכל לחבר את כניסות ויציאות ה-ALU לקבועות
קבועות ב- Z_REG ורק נזיז את ה DATA שבתוך Z_REG
ימינה לאחר כל פעולת חיבור

במקום:

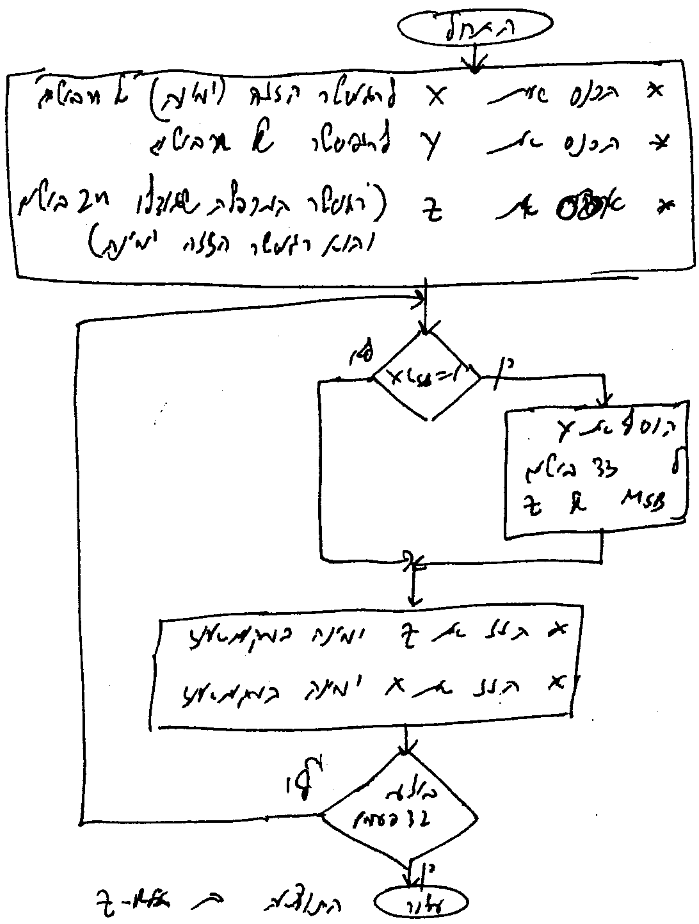


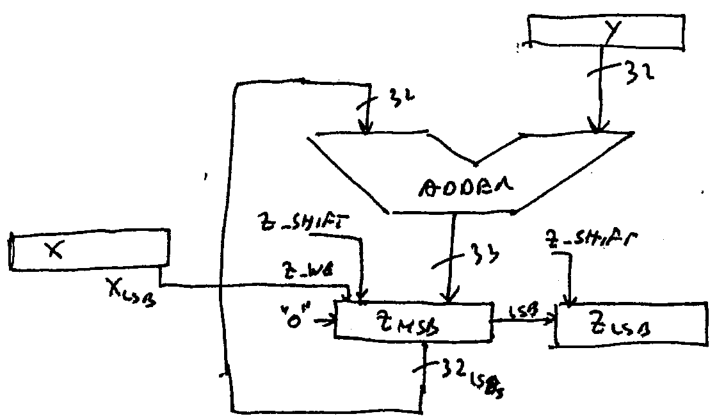
נבצע רק:



כפי שרואים, בכל שלב נכנסים מתחילת האפסים. התחיבור
הינו רק n ביטים עם n ביטים ותוצאה n+1 ביטים.
ואחרי ההזזה ימינה ושוב "חיבור" ל-n ביטים.

תרשים הזרימה הבא והמעגל שמתארו בעזרת מימוש
את הכפל בעזרת ADDER של n ביטים ורגיסטר הזזה ב-Z:





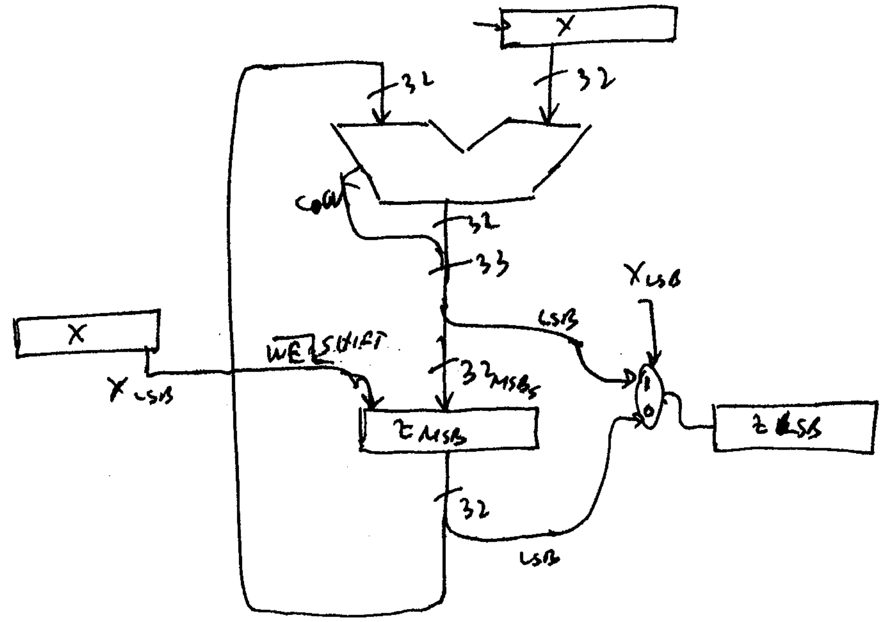
כאן "דיגון" שני מצבי שעון לכל "סבב" (כל ביט מוכפל). לפי X_{LSB}
רואים האם לשמור התוצאה ל-Z (אם $X_{LSB}=1$) או להשאיר ללא שינוי (אם $X_{LSB}=0$)
ואח"כ נבצע הזזה ימינה = סה"כ מצבי שעון

הכפל הסופי (כאשר לא הוספנו את X ל- אוגר Z)

חישוב ביט 0 של המכפלה	Z	0 00000 00000	
	+ Y	10011	
		0 100111 00000	ולאחר הזזה
חישוב ביט 1 של המכפלה	Z	0 010011 10000	
		ביט הסימן = ביט 0	
	+	10011	
		0 111000 10000	ולאחר הזזה
חישוב ביט 2	Z =	0 011100 01000	2 ביטים משמעותיים של המכפלה
	+	10011	
		1 000001 01000	ולאחר הזזה
חישוב ביט 3	Z =	0 100000 10100	
	+	10011	
		1 000111 10100	
חישוב ביט 4	Z =	0 100011 11010	ולאחר הזזה
			4 ביטים של המכפלה
	+	10011	
		1 001000 11010	ולאחר הזזה
		0 100100 01101	

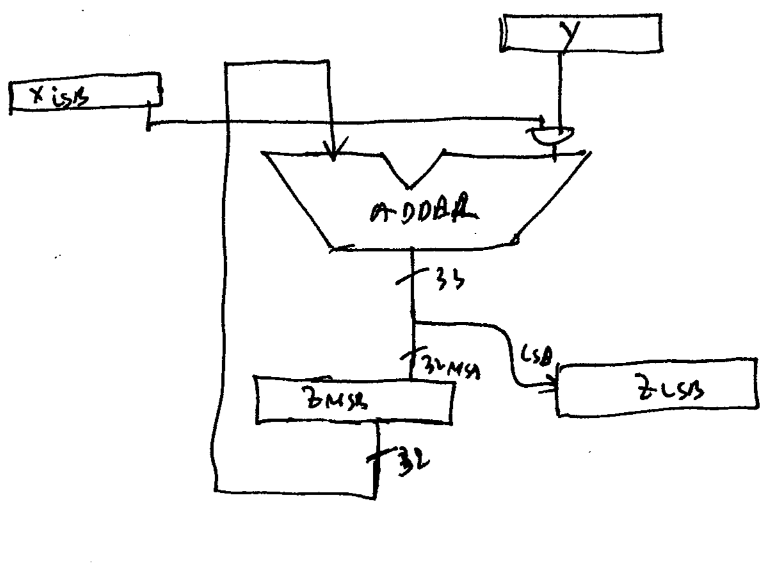
המכפלה הסופית.

כדי לחסוך ולפלוט הכל במחזור שעון אחד צריך מעט
לשלב הזזה בחלק הימני ע"י SHIFT REG והזזה בחלק השמאלי
ע"י הכנסת התוצאה למקום אחד ימינה.



החלק הימני הוא SHIFT REG שמקבל את עצמו מוזז ואת האות הבא LSB של התוצאה (אם X_{LSB}="1") או של התוצאה הקודמת (אם $X_{LSB}=0$ שאז אין שינוי ב z מלפני ההזזה)
החלק השמאלי הוא רגיסטר רגיל שמקבל עדכון של התוצאה מוזזת אם $X_{LSB}=1$ ושולט סתם בהזזה ימינה אם X_{LSB}="0"

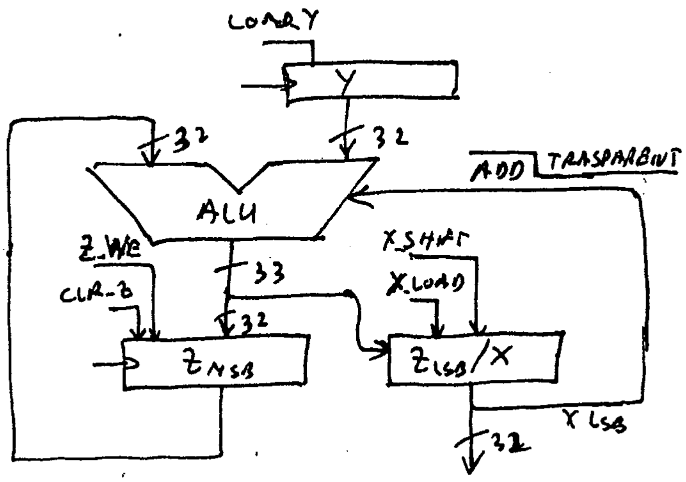
ועוד קל:

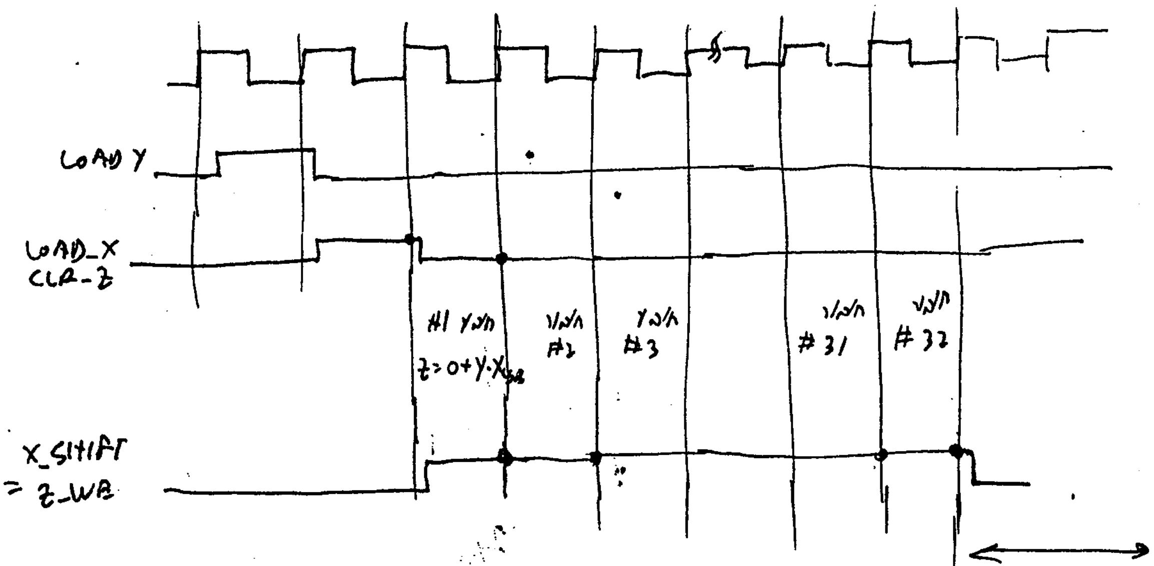


כאן Z_{LSB} = SHIFT REG
שמזיז ימינה כל מחזור שעון
Z_{MSB} = REGULAR REG
שנטען כל שעון
התוצאה מוזזת ימינה

החסכון הנוסף אותו אפשר לעשות הוא הכנסת X
לחלק הימני של Z שקראנו Z_{LSB}. כך נוכל לחסוך
ברגיסטר הנוסף שכן הוא זז ימינה כל מחזור שעון.

ולכן המעגל הסופי הוא





התוצאה נשמרת
ב- Z_{MSB} ו- Z_{LSB}

כפל מספרים בשיטת BOOTH

המטרה העיקרית של השיטה הנ"ל היא לחסוך מכפלה בעזרת ביצוע הזזה במקום חיבורים.

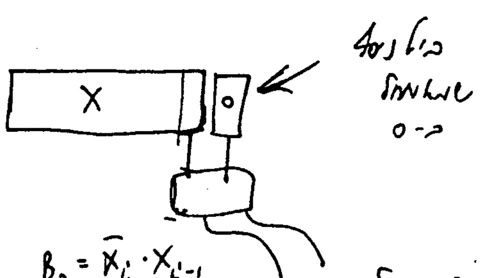
* הרעיון הוא שאם מכפילים ב 1111 = 15 בעצם אפשר לבצע הכפלה ב- (16-1) כלומר חיבור פעם אחת של הזזה ואז חיסור. וזאת במקום 4 חיבורים והזזות.

* לפיכך נבדיל בין 4 מקרים:

1) הביט הקודם היה 0 הביט הנוכחי הוא 0 (ב-X) במקרה כזה רק הזזה
2) הביט הקודם היה 0 והנוכחי 1 = זוהי תחילה של סדרת אחדים. לכן נחסר את Y מהתוצאה החלקית
3) הביט הקודם היה 1 והנוכחי 0 = זהו סיום של סדרת אחדים. לכן נחבר את Y לתוצאה החלקית
4) הביט הקודם היה 1 והנוכחי 1 = אנו באמצע סדרה של אחדים לכן רק הזזה.

* למעשה הערך של $(X_{i-1}-X_i)$ הוא שקובע מה יקרה. אם $(X_{i-1}-X_i)=0$ רק הזזה. אם $(X_{i-1}-X_i)=1$ נחבר ואם $(X_{i-1}-X_i)=-1$ נחסר.

לערך $(X_{i-1}-X_i)$ קוראים BOOTH RECODING



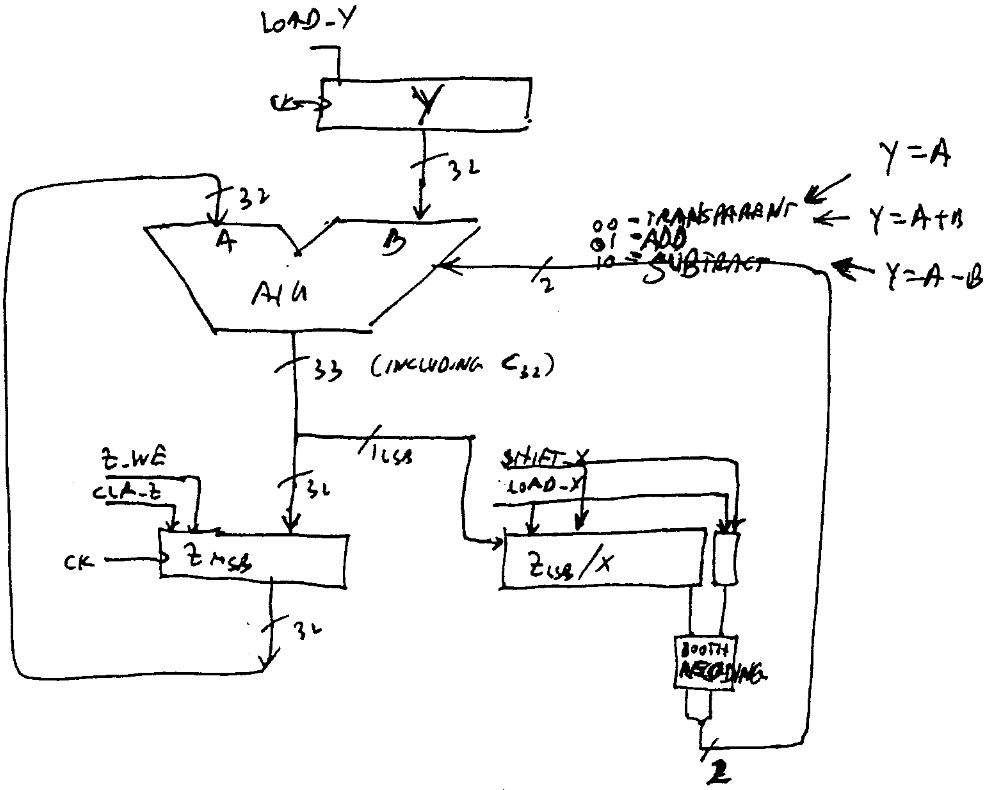
X_i	X_{i-1}	B_1	B_0
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

$B_0=\bar{X}_i\cdot X_{i-1}$

$B_1=X_i\cdot\bar{X}_{i-1}$

$$B_1B_0=\begin{cases}00 = \text{SHIFT}\\01 = \text{ADD} + \text{SHIFT}\\10 = \text{SUB} + \text{SHIFT}\end{cases}$$

מעגל המכפל הוא כדלקמן



כאשר המספר X הינו בגודל n במשלים שני מקבלים למעשה!

$$Z = (X_{-1}-X_0)\cdot Y\cdot 2^0 + (X_0-X_1)\cdot Y\cdot 2^1 + (X_1-X_2)\cdot Y\cdot 2^2 + \dots + (X_{n-2}-X_{n-1})\cdot Y\cdot 2^{n-1}$$

$$= \underset{0}{\underset{\uparrow}{X_{-1}}}\cdot Y\cdot 2^0 + X_0\cdot\underbrace{(2^1-2^0)}_{2^0}\cdot Y + X_1\cdot\underbrace{(2^2-2^1)}_{2^1}\cdot Y + X_2\cdot\underbrace{(2^3-2^2)}_{2^2}\cdot Y + \dots + X_{n-2}\overbrace{(2^{n-1}-2^{n-2})}^{2^{n-2}}\cdot Y - X_{n-1}\cdot 2^{n-1}\cdot Y$$

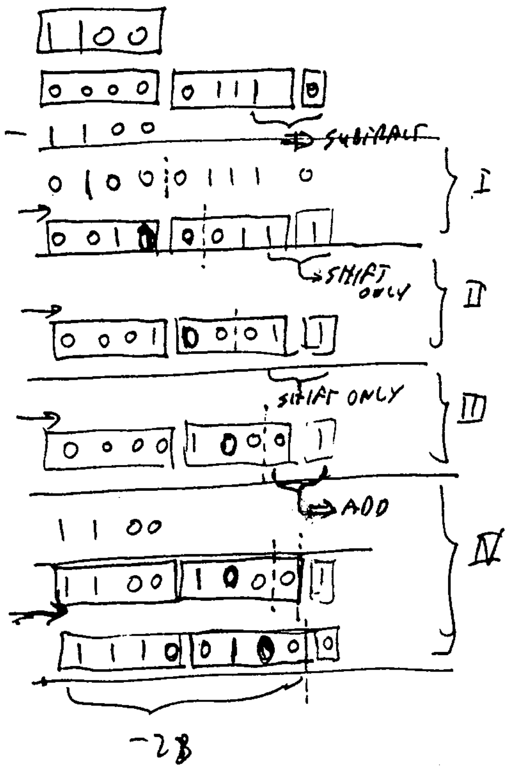
$$= \left(X_0\cdot 2^0 + X_1\cdot 2^1 + X_2\cdot 2^2 + \dots + X_{n-2}\cdot 2^{n-2} - X_{n-1}\cdot 2^{n-1}\right)\cdot Y =$$

שם X הוא בשיטת
המשלים ל-2

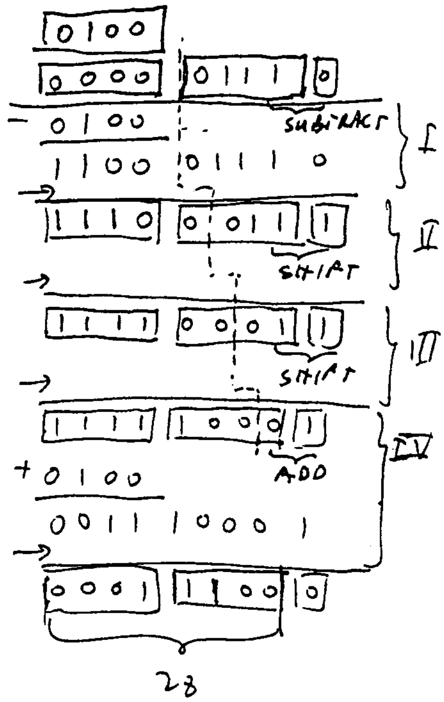
$$= X\cdot Y$$

(כמובן שגם Y בשיטת המשלים ל-2 צריך לבצע SIGN EXTENSION ל-Y)

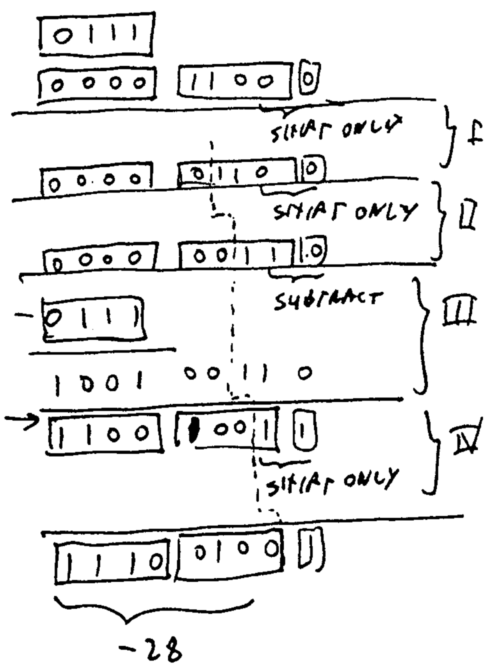
(-4) × 7


SUBTRACT
I
SHIFT ONLY
II
SHIFT ONLY
III
ADD
IV
-28

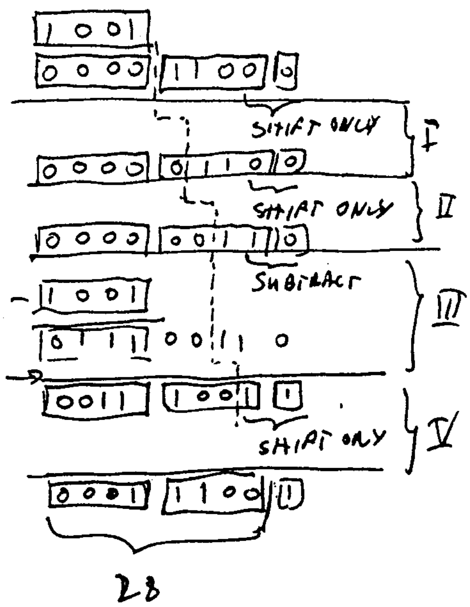
4 × 7


SUBTRACT
I
SHIFT
II
SHIFT
III
ADD
IV
28

7 × (-4)


SHIFT ONLY
I
SHIFT ONLY
II
SUBTRACT
III
SHIFT ONLY
IV
-28

(-7) × (-4)


SHIFT ONLY
I
SHIFT ONLY
II
SUBTRACT
III
SHIFT ONLY
V
28