PREPARATION REPORT LAB1

ADVANCED CPU ARCHITECTURE AND HARDWARE

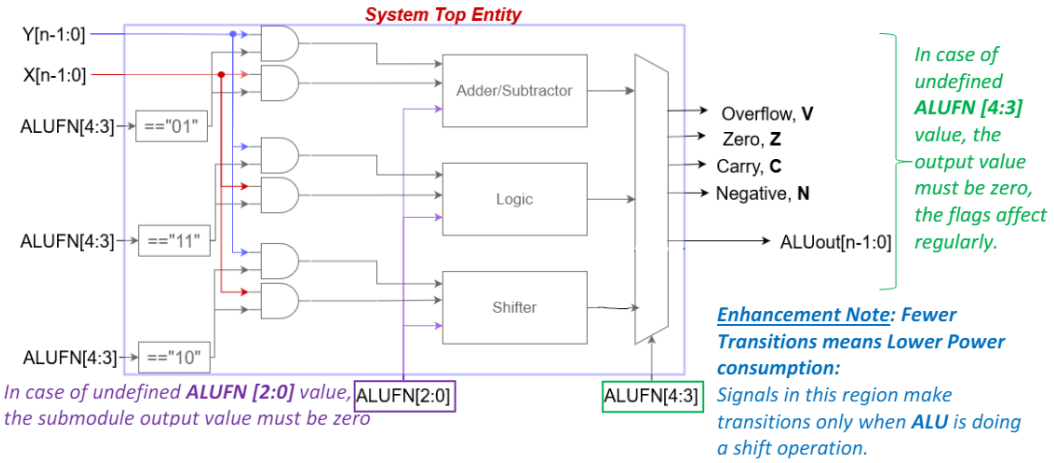
ACCELERATORS LAB

Tal Adoni – 319087300

Omri Aviram – 312192669

בקוד מימשנו מערכת ALU ע״י 3 יח׳ הלוגיות (Boolean ,Shifter ,Adder/ Subtractor) זאת יחד עם כניסות Y,X שעליהן נעשות פעולות בהתאם לקלט מתאים המפורט בהמשך וע״י הפעלת דגלים (Carry, Zero, Overflow, Negative) על פי המוצא שלנו.

כמו כן עבור מוצאים ״לא חוקיים״ (ללא הגדרה) קבענו מוצא 0 ולפיכך הופעל דגל Zero.



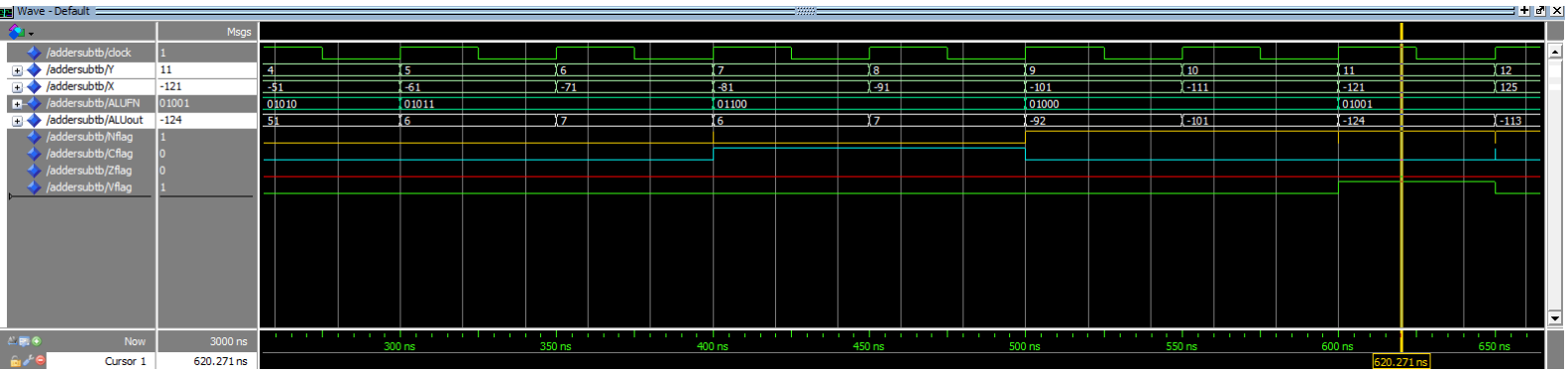
Adder/ Subtractor (Operation Code = 01):

בחלק זה מימשנו את מחבר/ מחסר על פי קוד כניסה מבוקש:

- 000 – חיבור
- 001 – חיסור
- 010 – מוצא שלילי של Y
- 011 – הגדלה של Y ב- 1
- 100 – הקטנה של Y ב- 1

מימוש הקוד נעשה על ידי בדיקת שלושת הביטים ה-LSB של הקלט וסיווג על פיהם מהי הפעולה הרצויה.

כפי שניתן לראות בתמונה המצורפת בעמ' הבא:



הסמן יושב בנק' 620 ns עבור הכניסות – ALUin = 01001 ,X= -121 ,Y = 11 כלומר מתבצעת הפעולה Y-X כאשר X ו-Y מיוצגים באופן דצימלי

המוצא שלנו הוא ALUout = -124 ולכן הדגלים שפועלים הם overflow=1, negative=1 זאת מפני שאנו עובדים לפי שיטת משלים ל-2 ולכן התוצאה שלנו אמורה להיות חיובית, שהרי X שלילי ולכן הפעולה זהה לחיבור שני מספרים חיוביים, אך התוצאה שקיבלנו הינה שלילית הביט של הסימן שהינו ה-MSB משתנה ל-1 בעקבות החיבור (לפי שיטת משלים ל-2) ובשל סיבה זו קיבלנו גם דגל overflow.

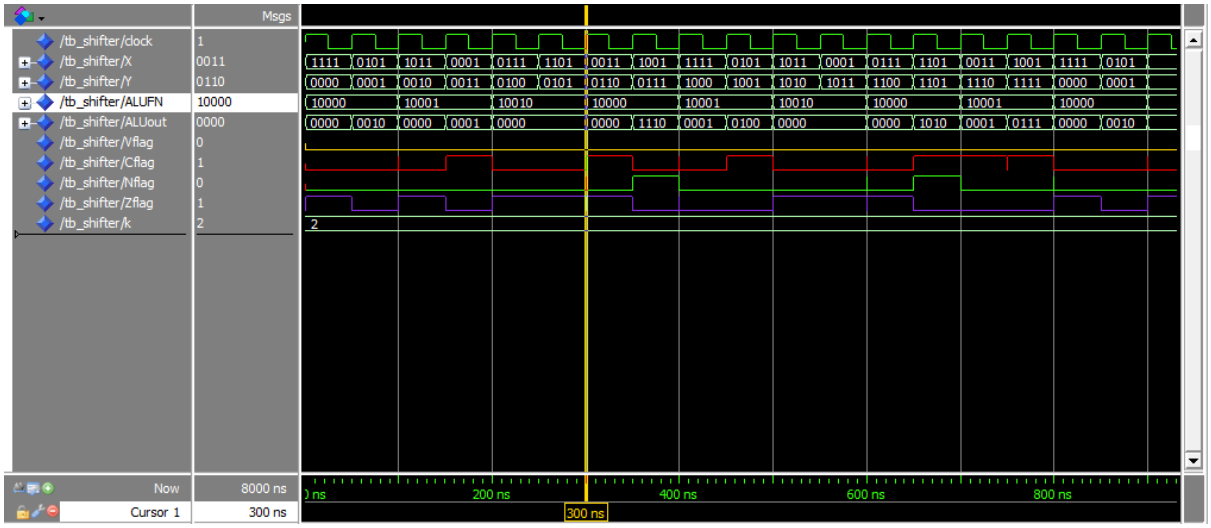
Shifter (Operation Code = 10):

בחלק זה מימשנו Shifter על פי קוד כניסה מבוקש:

- 000 - הזזה שמאלה של Y לפי ה-K ספרות של X כאשר $K = \log_2(n)$
- 001 - הזזה ימינה של Y לפי ה-K ספרות של X כאשר $K = \log_2(n)$

מימוש הקוד נעשה על ידי על פי K הביטים ה-LSB של X ועבורם נעשית הזזה בלוגיקת Barrel Shifter.
בנינו מטריצה מגודל K x N, בכל איטרציה אנו בודקים עבור כל שורה את הביט המתאים במידה והביט 1 אנחנו דוחפים $2^{row\ number}$ אפסים מימין/ משמאל (על פי הקידוד) ולאחר מכן עוברים לשורה הבאה.

כפי שניתן לראות בתמונה המצורפת:



הסמן יושב בנק' 300 ns עבור הכניסות – Y = 00011101, X=11011101, ALUin = 10001
כלומר מתבצעת הפעולה SHR(Y); X(k-1 to 0).

k=3 כלומר אנו מזיזים את Y לפי המספר המיוצג בשלוש הסיביות ה-LSB של X, כלומר נבצע הזזה ימינה 5 פעמים.

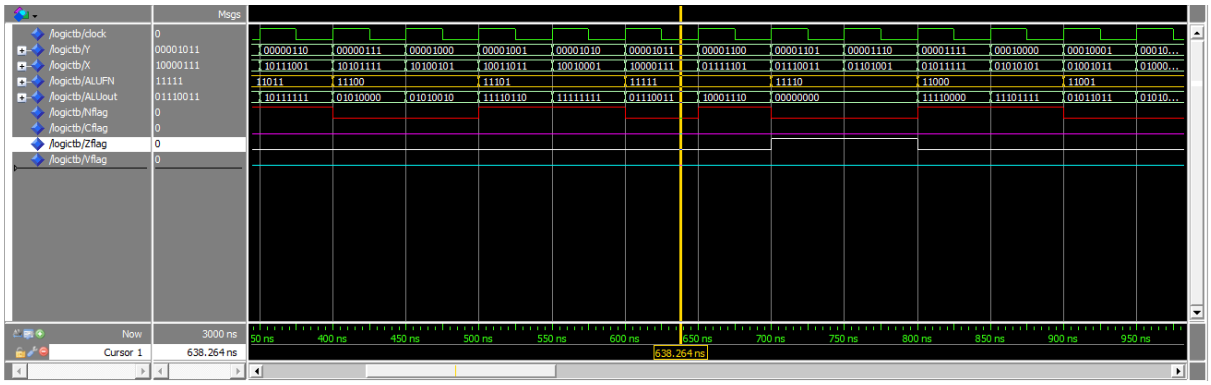
המוצא שלנו הוא ALUout = 00000000 ולכן הדגלים שפועלים הם carry = 1, zero=1
הדבר נובע מכך שכאשר אנחנו מבצעים הזזה ימינה אנחנו מכניסים 0 ל-MSB וכל הסיביות האחרות מ-n-1 עד ל-n-3 זזות גם הן ימינה והסיבית ה- 5 עוברת ל- carry, במקרה זה הסיבית הינה 1 ולכן מופעל דגל carry.

Boolean (Operation Code = 11):

בחלק זה מימשנו Shifter על פי קוד כניסה מבוקש:

- $000 - NOT(Y)$
- $001 - Y\ OR\ X$
- $010 - Y\ AND\ X$
- $011 - Y\ XOR\ X$
- $100 - Y\ NOR\ X$
- $101 - Y\ NAND\ X$
- $111 - Y\ XNOR\ X$

כפי שניתן לראות בתמונה המצורפת:

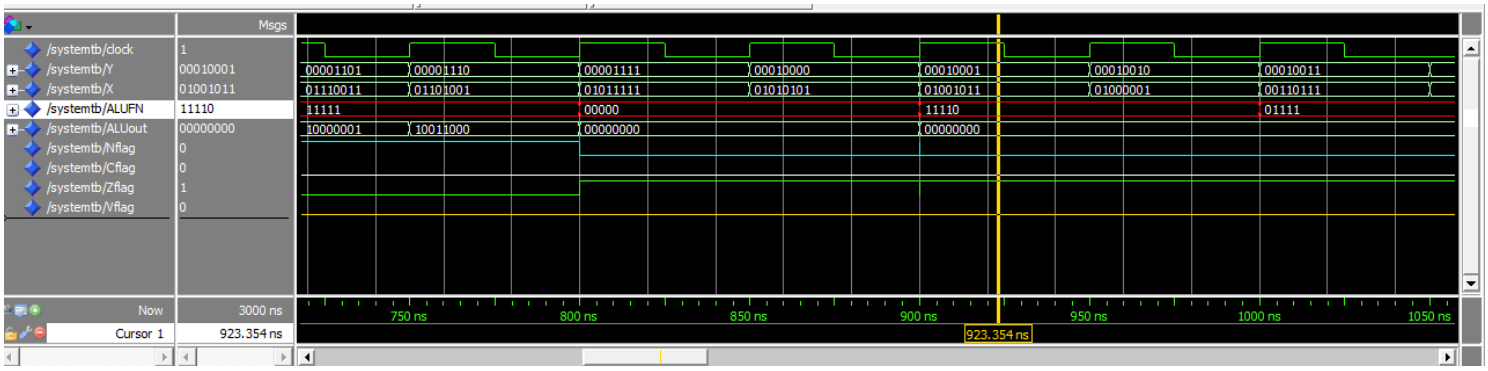


הסמן יושב בנק' 638 ns עבור הכניסות – Y = 00001011, X= 10000111, ALUin = 11111 כלומר מתבצעת הפעולה XNOR(X,Y) ולכן המוצא הינו ALUout=01110011 כלומר היכן שהסיביות היו זהות נקבל 1 והיכן שהיו שונות נקבל 0.

סימולציה על ה-Top:

את מודל ה-Top אנו ממשים את התכנית הכוללת של ה- ALU המכילה את שלושת היח' הלוגיות (Boolean ,Shifter ,Adder/ Subtractor) זאת יחד עם כניסות Y,X וקובץ ה-Top בורר על פי קידוד את היח' עליו להשתמש לפי קוד הכניסה מ- ALUFN.

כפי שניתן לראות בתמונה המצורפת:



הסמן יושב בנק' 923 ns עבור הכניסות – Y = 00010001, X= 01001011, ALUin =11110
כלומר אנחנו ממשים Logic אך הקידוד אינו נכון ולכן המוצא הינו ALUout = 00000000
ולכן נקבל גם דגל zero=1.

סימולציית דוגמה – ref1:

כעת נראה את תוצאת ה-list של ה-ALU שבנינו לעומת התוצאה המתקבלת לפי הטסט לדוגמה.

בתמונה המצורפת רואים כי התוצאות זהות, מלבד הדלטאות שמסומנות בכתום, שמהן נתבקשנו להתעלם.

	ps	delta	/tb/Y	/tb/X	/tb/ALUFN	/tb/ALUout	/tb/Nflag	/tb/Cflag	/tb/Zflag	/tb/Vflag
	0	+7	11111111	11111111	01000	11111110	1	1	0	0
	50000	+7	11111110	11110101	01000	11110011	1	1	0	0
	100000	+6	11111101	11101011	01001	00010010	0	1	0	0
	150000	+6	11111100	11100001	01001	00011011	0	1	0	0
	200000	+8	11111011	11010111	01010	00101001	0	0	0	0
	250000	+6	11111010	11001101	01010	00110011	0	0	0	0
	300000	+9	11111001	11000011	01011	11111010	1	0	0	0
	350000	+6	11111000	10111001	01011	11111001	1	0	0	0
	400000	+8	11110111	10101111	01100	11110110	1	1	0	0
	450000	+6	11110110	10100101	01100	11110101	1	1	0	0
	500000	+10	11110101	10011011	01000	10010000	1	1	0	0
	550000	+8	11110100	10010001	01000	10000101	1	1	0	0
	600000	+7	11110011	10000111	01001	01101100	0	1	0	0
#	650000	+10	11110010	01111101	01001	01110101	0	1	0	1
#	700000	+4	11110001	01110011	01111	00000000	0	0	1	0
	750000	+1	11110000	01101001	01111	00000000	0	0	1	0
#	800000	+7	11101111	01011111	10000	10000000	1	1	0	0
#	850000	+7	11101110	01010101	10000	11000000	1	1	0	0
#	900000	+6	11101101	01001011	10001	00011101	0	1	0	0
#	950000	+6	11101100	01000001	10001	01110110	0	0	0	0
#	1000000	+4	11101011	00110111	10010	00000000	0	0	1	0
	1050000	+1	11101010	00101101	10010	00000000	0	0	1	0
#	1100000	+7	11101001	00100011	10000	01001000	0	1	0	0
#	1150000	+7	11101000	00011001	10000	11010000	1	1	0	0
#	1200000	+6	11100111	00001111	10001	00000001	0	1	0	0
#	1250000	+6	11100110	00000101	10001	00000111	0	0	0	0
#	1300000	+4	11100101	11111011	10010	00000000	0	0	1	0
	1350000	+1	11100100	11110001	10010	00000000	0	0	1	0
#	1400000	+4	11100011	11100111	11001	11100111	1	0	0	0
#	1450000	+4	11100010	11011101	11001	11111111	1	0	0	0
#	1500000	+4	11100001	11010011	11010	11000001	1	0	0	0
#	1550000	+4	11100000	11001001	11010	11000000	1	0	0	0
#	1600000	+4	11011111	10111111	11101	01100000	0	0	0	0
#	1650000	+4	11011110	10110101	11101	01101011	0	0	0	0
#	1700000	+4	11011101	10101011	11111	10001001	1	0	0	0
#	1750000	+4	11011100	10100001	11111	10000010	1	0	0	0
#	1800000	+4	11011011	10010111	11011	01001100	0	0	0	0
#	1850000	+4	11011010	10001101	11011	01010111	0	0	0	0
#	1900000	+2	11011001	10000011	00100	00000000	0	0	1	0
	1950000	+1	11011000	01111001	00100	00000000	0	0	1	0
	2000000	+1	11010111	01101111	00100	00000000	0	0	1	0

	ps	delta	/tb/Y	/tb/X	/tb/ALUFN	/tb/ALUout	/tb/Nflag	/tb/Cflag	/tb/Zflag	/tb/Vflag
	0	+5	11111111	11111111	01000	11111110	1	1	0	0
	50000	+7	11111110	11110101	01000	11110011	1	1	0	0
	100000	+6	11111101	11101011	01001	00010010	0	1	0	0
	150000	+6	11111100	11100001	01001	00011011	0	1	0	0
	200000	+8	11111011	11010111	01010	00101001	0	0	0	0
	250000	+6	11111010	11001101	01010	00110011	0	0	0	0
	300000	+9	11111001	11000011	01011	11111010	1	0	0	0
	350000	+6	11111000	10111001	01011	11111001	1	0	0	0
	400000	+8	11110111	10101111	01100	11110110	1	1	0	0
	450000	+6	11110110	10100101	01100	11110101	1	1	0	0
	500000	+10	11110101	10011011	01000	10010000	1	1	0	0
	550000	+8	11110100	10010001	01000	10000101	1	1	0	0
	600000	+7	11110011	10000111	01001	01101100	0	1	0	0
#	650000	+9	11110010	01111101	01001	01110101	0	1	0	1
#	700000	+7	11110001	01110011	01111	00000000	0	0	1	0
	750000	+1	11110000	01101001	01111	00000000	0	0	1	0
#	800000	+9	11101111	01011111	10000	10000000	1	1	0	0
#	850000	+8	11101110	01010101	10000	11000000	1	1	0	0
#	900000	+7	11101101	01001011	10001	00011101	0	1	0	0
#	950000	+8	11101100	01000001	10001	01110110	0	0	0	0
#	1000000	+9	11101011	00110111	10010	00000000	0	0	1	0
	1050000	+1	11101010	00101101	10010	00000000	0	0	1	0
#	1100000	+9	11101001	00100011	10000	01001000	0	1	0	0
#	1150000	+8	11101000	00011001	10000	11010000	1	1	0	0
#	1200000	+7	11100111	00001111	10001	00000001	0	1	0	0
#	1250000	+7	11100110	00000101	10001	00000111	0	0	0	0
#	1300000	+9	11100101	11111011	10010	00000000	0	0	1	0
	1350000	+1	11100100	11110001	10010	00000000	0	0	1	0
#	1400000	+5	11100011	11100111	11001	11100111	1	0	0	0
#	1450000	+5	11100010	11011101	11001	11111111	1	0	0	0
#	1500000	+5	11100001	11010011	11010	11000001	1	0	0	0
#	1550000	+5	11100000	11001001	11010	11000000	1	0	0	0
#	1600000	+5	11011111	10111111	11101	01100000	0	0	0	0
#	1650000	+5	11011110	10110101	11101	01101011	0	0	0	0
#	1700000	+5	11011101	10101011	11111	10001001	1	0	0	0
#	1750000	+5	11011100	10100001	11111	10000010	1	0	0	0
#	1800000	+5	11011011	10010111	11011	01001100	0	0	0	0
#	1850000	+5	11011010	10001101	11011	01010111	0	0	0	0
#	1900000	+3	11011001	10000011	00100	00000000	0	0	1	0
	1950000	+1	11011000	01111001	00100	00000000	0	0	1	0
	2000000	+1	11010111	01101111	00100	00000000	0	0	1	0