**ADDER**

במימוש מחבר/מחסר השתמשנו במחבר גלי עם תוספת קו בקרה לחיסור. ה-cin משמש את פעולת (1+) אשר נדרשת כאשר עושים 2’s complement ל-X. את החלק שני של 2’s complement כלומר היפוך ביטים ,עושה שער XOR עם '1' עם כל אחד מהביטים של X.  
3 ביטים נמוכים של ALUFN (נסמן ב- ALU20) הם אלו שקובעים את אופן העבודה של הרכיב. במקרים בהם ALU20=”010”/”001” נפעיל את המחסר ונכניס cin=’1’. במקרה שבו ALU20=”010” גם נאפס את Y כדי לחבר את -X עם 0 ולקבל neg(X). שאר המצבים הם חיבור או מצב לא מוגדר עם פלט קבוע 0.

**Shape, square

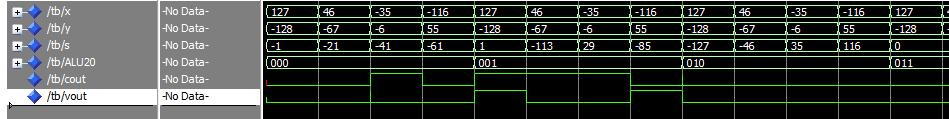
Description automatically generated**

Vout

A picture containing text, clock, scale

Description automatically generated

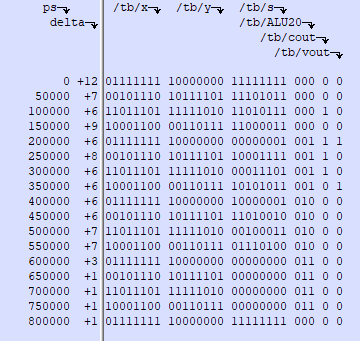
**דיאגרמת WAVES:**

במקרה של מחבר/מחסר נציג את האותות בייצוג דצימלי כי כך ניתן לבדוק את התקינות שלו בקלות.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| “011” & other | “010” | “001” | “000” | ALU20 |
| S = 0 | S = ( – X ) | S = Y – X | S = Y + X | Operation |
| 0 | -127 | -128-127=**1** | -128+127=-1 | Y= –128 , X= 127 |
| 0 | -46 | -67-46=-113 | -67+46=-21 | Y= –67 , X= 46 |
| 0 | 35 | -6+35=29 | -6-35=-41 | Y= –6 , X= –35 |
| 0 | 116 | 55+116=**-85** | 55-116=-61 | Y= 5 , X= –116 |

לקחנו כמה דגימות מהאותות על מנת להבין מה קורה שם ולהשוות עם חישוב ידני שלנו. נשים לב שהפלט תקין מלבד 2 מקרים בהם עקב overflow/carry יש שוני בגלל מוגבלויות הייצוג של 8 ביט. נסביר את התופעה ע"י חישוב הבא:

עבור הדוגמאות הנ"ל ניתן לראות כי אכן ביט הoverflow דולק.

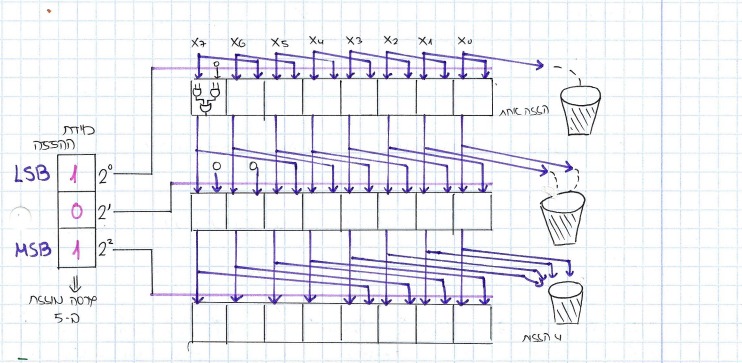
**LIST** עבור אותה הרצה:

**SHIFTER**

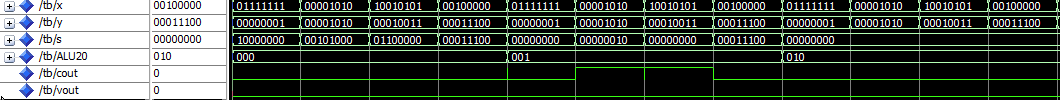
****את ה-שיפטר מימשנו באמצעות מטריצה.

כפי שנלמד בקורס "מבוא למחשבים" על מימוש   
“shift barrel” ע"י k הביטים הראשונים של הווקטור x באופן איטרטיבי –

וקטור התחלתי y   
עוברים על k הביטים הראשונים של x :  
ביט שהינו 1 מזיז את הווקטור y' האחרון שנכנס למטריצה 1\*2^(אינדקס הביט) כלומר הערך הדצימלי של הביט ומכניס וקטור חדש למטריצה.  
ביט שהינו 0 מכניס למטריצה ישירות את הווקטור הקודם שתופס עוד עמודה/שורה במטריצה.  
\*\*הבחירה האם ההזזה היא שמאלה או ימינה לפי כניסת ה- ALU   
לאחר k ביטים קיימת מטריצה nXk שעמודותיה הם ווקטורי y מוזזים.  
הווקטור האחרון יהיה וקטור המוצא.  
גם כאן vout תמיד 0 כי פעולת השיפטר אינה מייצרת overflow אבל כן מייצרת cout.

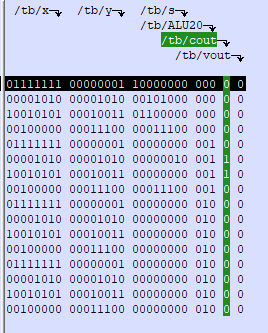


השיפטר שממשנו אינו מממש רכיבי mux אלא ההזזה מתבצעת בעזרת שרשור וקטור חדש עם וקטור אפסים לפי מספר ההזזות הדרושות.

**דיאגרמת WAVES:**

|  |  |  |
| --- | --- | --- |
| Y = 00001010 | Y = 00010011 | Y = 00010011 |
| X = 00001**010** => **2** shifts | X = 10010**101** => **5** shifts | X = 10010**101** => **5** shifts |
| ALU20 = 00**0** => **left** | ALU20 = 00**0** => **left** | ALU20 = 00**1** =>**right** |
| S = 00101000 | S = 01100000 | S = 00000000 |

לקחנו כמה דגימות מהאותות על מנת להבין מה קורה שם ולהשוות עם חישוב ידני שלנו.  
נמצא כי הפלט תקין עבור כל קלט.  
\*עבור ALU20 אחרים, הפעולה לא מוגדרת. נקבל פלט אפס



**LIST**

הפלט תקין.

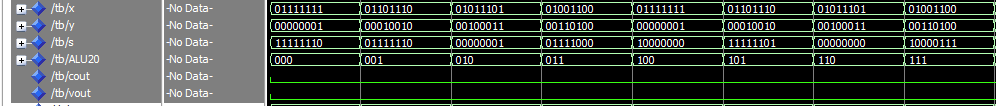
**LOGIC**

את הלוגיקה ממשנו באמצאות שערים ייעודיים לכל מצב פעולה. כל זוג סיבים מאותו מיקום מ-X ו-Y נכנס לשער עם Fan-in של 2 והפלט הוא פשוט מוצא של כל שאר בהתאמה באותו סדר של סיביות שבאו מ-X ו-Y.  
כאן הדגלים Cout ו Vout תמיד 0.

Shape, square

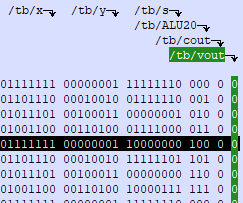
Description automatically generated

**דיאגרמת WAVES:**



|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| “111” | “110” | “101” | “100” | “011” | “010” | “001” | “000” | ALU20 |
| Y xnor X | **0** | Y nand X | Y nor X | Y xor X | Y and X | Y or X | Not(Y) | OPER. |
| 01001100 | 01011101 | 01101110 | 01111111 | 01001100 | 01011101 | 01101110 | 01111111 | X |
| 00110100 | 00100011 | 00010010 | 00000001 | 00110100 | 00100011 | 00010010 | 00000001 | Y |
| 10000111 | 00000000 | 11111101 | 10000000 | 01111000 | 00000001 | 01111110 | 11111110 | S |

הפלט S חושב ידנית על ידינו על מנת להשוות עם הפלט של המערכת, קיבלנו התאמה מלאה.

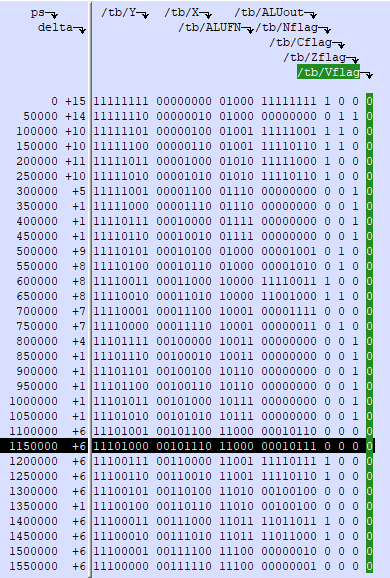
**LIST:**

**TOP**

הTOP הוא זה שמחבר את כל הרכיבים המוזכרים לעיל. הקלט נכנס אליו בהתחלה ובאמצאות קו בקרה ALUFN בעל 5 ביטים, נבחר הרכיב שיפעל והמצב שבו הוא יפעל. 2 ביטים גבוהים סימנו אותם ב-ALU43 בוחרים את הרכיב שיפעל, בעוד ש-3 ביטים נמוכים שסימנו ב-ALU20 בוחרים את אופן הפעולה הפנימי של הרכיב שנבחר. בנוסף לפלט שמקבל מרכיב, הוא בודק האם יש צורך להרים דלגים.

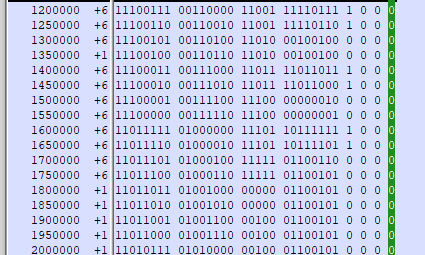
Diagram

Description automatically generated with medium confidence

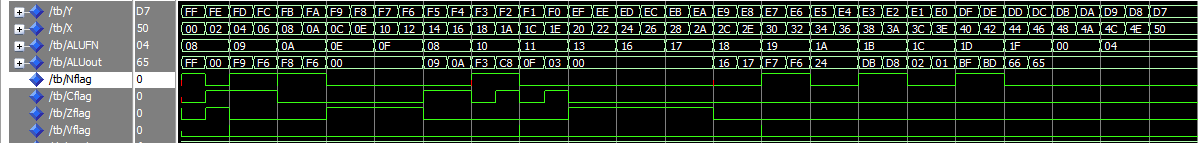
**דיאגרמת LIST:**נתחיל הפעם מרשימה כי יש הרבה מאוד נתונים.  
נסמן אזורים ספציפיים באופן הבא:

ניקח מספר דגימות מכל רכיב ונבדוק מה קורה שם והם פלט תקין.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Res/Flags | X | Y | Operation | ALUFN |
| **100000000**=0 / **C,Z** | 00000010=2 | 11111110=-2 | Y + X | 01000 |
| **11**1111001=-7 / **C,N** | 00000100=4 | 11111101=-3 | Y - X | 01001 |
| **1**1110110=-10 / **N** | 00001010=10 | Don’t care | - X | 01010 |
| **1**1001000 / **N** | 00011**010**=’2’ | 11110010 | SHL Y , X | 10000 |
| 00000011/ **C** | 00011**110**=’6’ | 11110000 | SHR Y , X | 10001 |
| 00010111 | Don’t care | 11101000 | not(Y) | 11000 |
| 11110111 | 00110000 | 11100111 | Y or X | 11001 |
| 00100100 | 00110110 | 11100100 | Y and X | 11010 |
| 11011000 | 00111010 | 11100010 | Y xor X | 11011 |
| 00000001 | 00111110 | 11100000 | Y nor X | 11100 |
| 10111111 | 01000000 | 11011111 | Y nand X | 11101 |
| 01100101 | 01000110 | 11011100 | Y xnor X | 11111 |

הערכים בטבלה חושבו ידנית ונמצא כי הפלט תקין.

**דיאגרמת WAVES:**

נציג את הדיאגרמה, כרגיל ניקח כמה דגימות (אחרות, שלא היו ב-LIST) ונבדוק מה קורה שם והאם זה נכון.  
נציג אותה בייצוג הקסא כדי שכל הנתונים יכנסו לתמונה אחת.

3

2

1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Res | X | Y | Operation | ALUFN | # |
| **FF**h=11111111b=-1+0=-1 | 00 | FF | Y+X | 08h=01000b | 1 |
| 00001111=**0F**h | 1C=00011**100** | F1=11110001 | SHR Y , X | 11h=10001b | 2 |
| 00010110=**16**h | 2C=00101100 | E9=11101001 | Not Y | 18h=11000b | 3 |

הפלט תקין.