פרויקט סיום- מעבדת VLSI דיגיטלי

מגיש: נפתלי ריפנר

# תקציר

הפרויקט הינו בניית מחשבון (חיבור וחיסור) באמצעות שימוש בזיכרון מסוג SRAM .

תוכן

[תקציר 2](#_Toc78488424)

[מבוא 3](#_Toc78488425)

[תא SRAM בודד 3](#_Toc78488426)

[Schematic 3](#_Toc78488427)

[Layout 4](#_Toc78488428)

[סימולציה 5](#_Toc78488429)

[מטריצת הזיכרון 5](#_Toc78488430)

[Schematic 5](#_Toc78488431)

[Layout 9](#_Toc78488432)

[סימולציה 11](#_Toc78488433)

[Full adder and full subtractor 12](#_Toc78488434)

[Schematic 12](#_Toc78488435)

[Layout 14](#_Toc78488436)

[סימולציה 15](#_Toc78488437)

[Decoder 18](#_Toc78488438)

[Schematic 18](#_Toc78488439)

[Layout 19](#_Toc78488440)

[סימולציה 19](#_Toc78488441)

[חיסור בשיטת המשלים ל-2 20](#_Toc78488442)

[משלים ל-2 20](#_Toc78488443)

[חיסור 20](#_Toc78488444)

[מחשבון 21](#_Toc78488445)

[Schematic 21](#_Toc78488446)

[Layout 24](#_Toc78488447)

[סימולציה 26](#_Toc78488448)

[תהליך הסימולציה של המחשבון 26](#_Toc78488449)

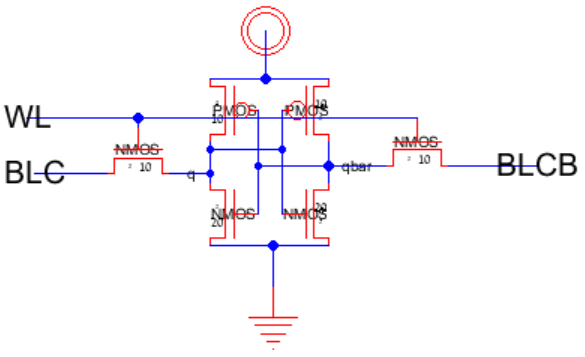
[רשימת איורים 28](#_Toc78488450)

# מבוא

אנו רוצים לחבר/לחסר שני מספרים בעלי 4 ביט כל אחד באמצעות adder-full( המשמש גם כ subtractor full )כאשר המספרים (כולם! גם התוצאות של החיבור/חיסור) צריכים לעבור דרך הזיכרון. לאחר ביצוע פעולת כתיבה נדרש לעשות פעולת קריאה לאותה השורה ע"מ לוודא כתיבה נכונה (בין אם זה ערך שאני הכנסתי ובין אם זה ערך שהגיע מהמטריצה!). מטריצת הזיכרון תהיה בת 16 שורות כאשר בכל שורה ישנם ארבעה תאי זיכרון (סה"כ 64 ביט). שורות המטריצה יחוברו ל decoder ע"מ לבחור את השורה הרצויה לקריאה/כתיבה. צריך לדאוג לכך שלמטריצה ניתן יהיה לכתוב בשני אופנים: דרך הספקים (שורות קוד) או דרך לוגיקה (מוצא ה adder full ). נדרש לסנכרן את האותות בצורה הנכונה כך שבסימולציה אחת ירוצו גם חיבור וגם חיסור (לשים לב, לא לדרוס שורות במטריצה בחיבור/חיסור). לגבי החיסור, מצורף הסבר על שיטת המשלים ואיך בדיוק מתבצע החיסור (מה קורה עם התוצאה היא שלילית וכו').

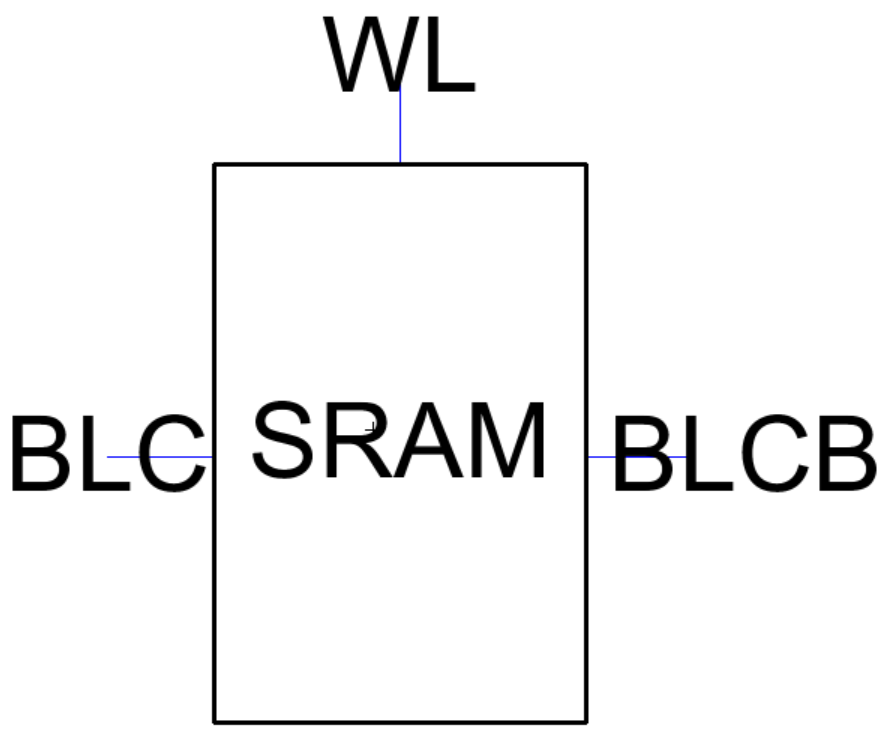
## תא SRAM בודד

### Schematic



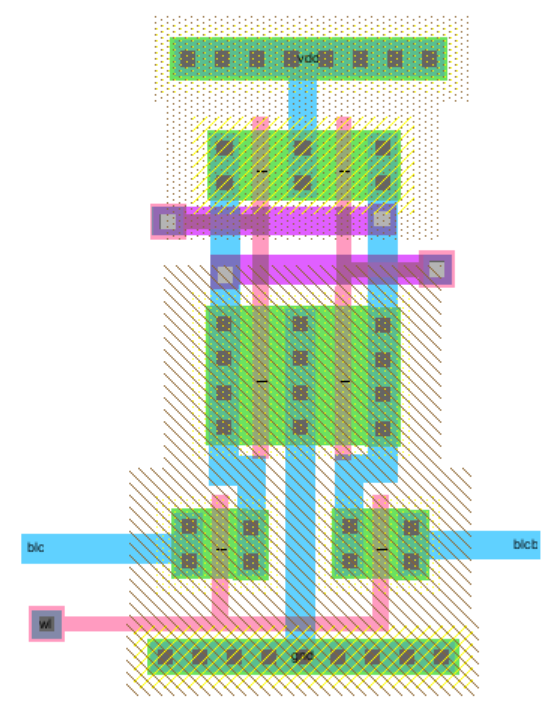
איור 1 תא SRAM בודד ב- schematic

ב- SRAM יש 3 חלקים חלק הPULL UP- חלק ה-PUL DOWN וחלק ה-Access. תא הזיכרון מורכב משני מהפכים המחוברים ביניהם בין המוצא של האחד לכניסה של האחר וכך משמרים את הערך הנגדי אחד של השני. בחלק ה-Access ניתן לראות שכאשר ה-wl במצב '1' אז הטרנזיסטורים המחוברים ל- blc,blcb במצב הולכה וכך מתאפשרת גישה לתוכן של תא הזיכרון. טרנזיסטורי ה-PULL DOWN צריכים להיות החזקים ביותר אחריהם טרנזיסטורי ה- Access והכי חלשים צריכים להיות טרנזיסטורי ה- PULL UP ולכן הם מינימאליים, ה-nmos יותר חזק מ-pmos ולכן גם הם בגודל מינימלי וה-PULL DOWN הגדולים יותר.



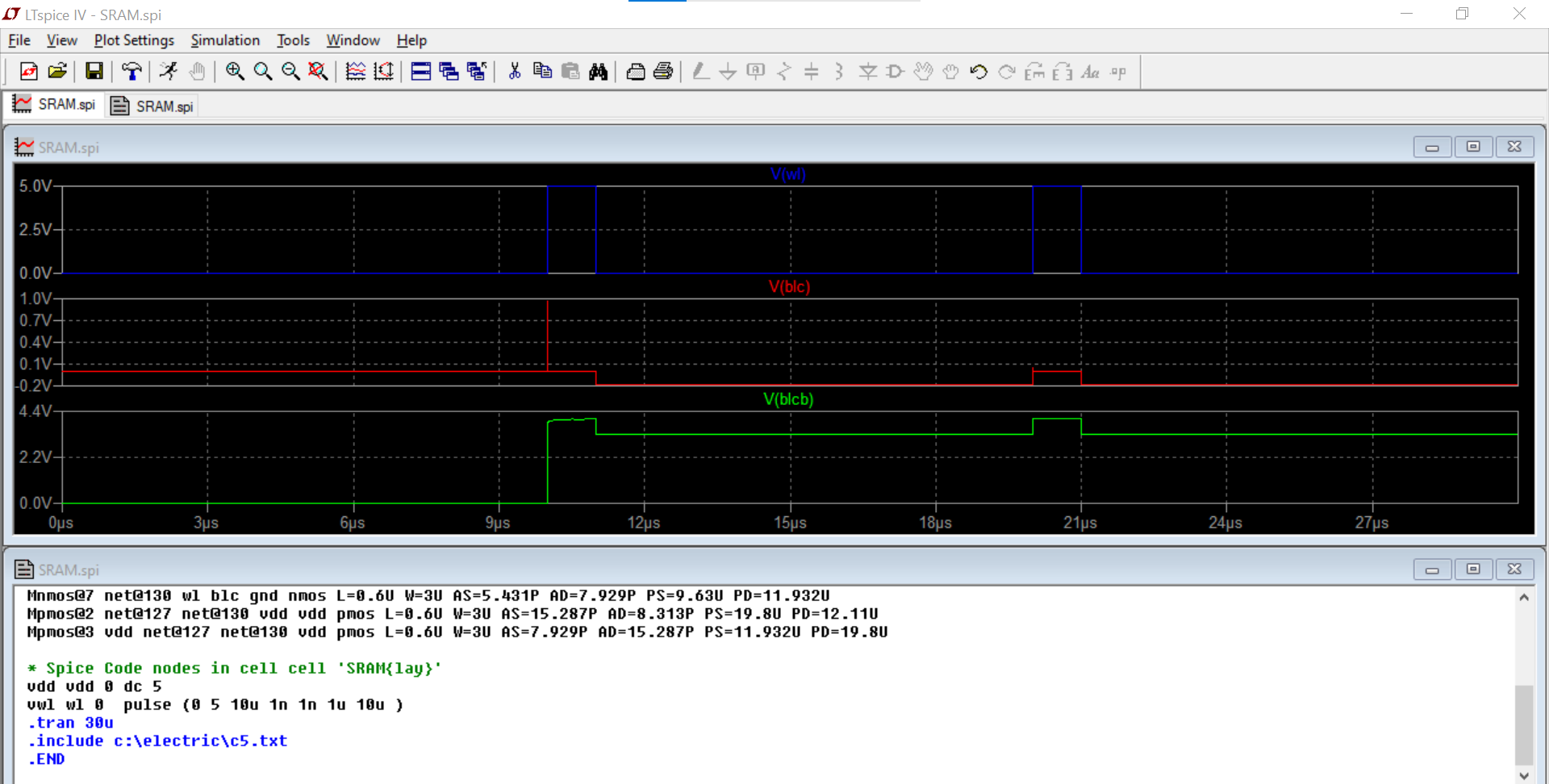
איור 2 מראה ה- icon של ה- SRAM

### Layout



איור 3 תא SRAM בודד

### סימולציה

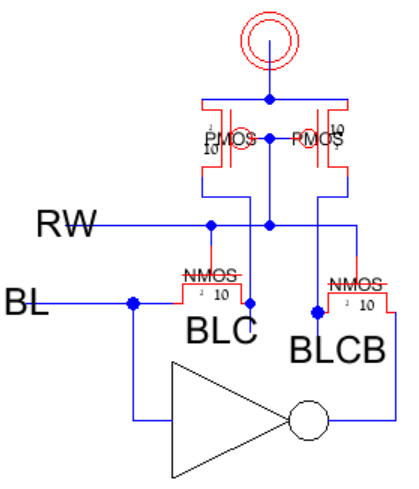


איור 4 סימולציית ה- SRAM עם שורת הקוד

ניתן לראות בסימולציה שכאשר ה- wl משנה את ערכו ל- '1' אז מתרחשים שינויים ב- blc,blcb. לאחר העלייה הראשונה של wl ניתן לראות שblc – ו- blcb הפוכים בערכם.

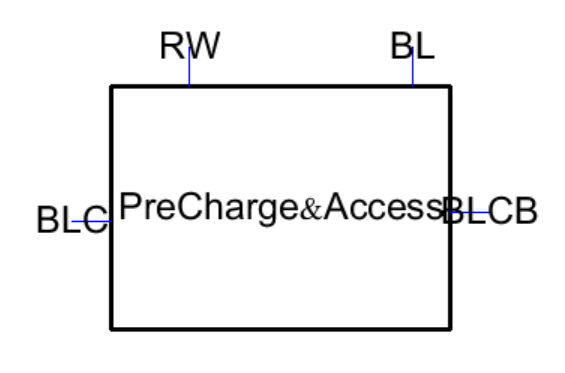
## מטריצת הזיכרון

### Schematic

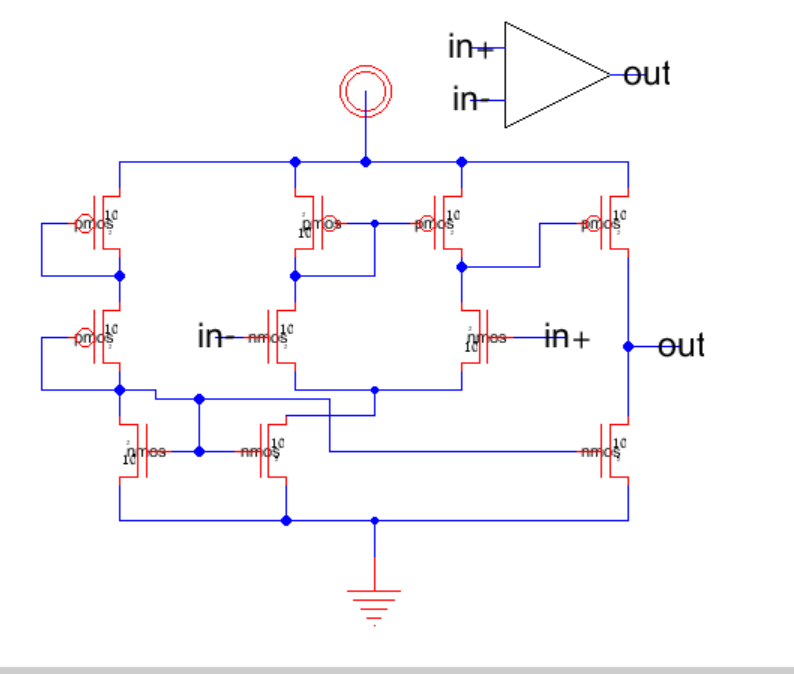


איור 5 מעגל ה-pre charge

מטרת המעגל היא לטעון את ה- blc,blcb שאמורים להיות מחוברים ליחידת הזיכרון SRAM בהתאם ל- bl כאשר ה- rw הוא שולט על פעולת הקריאה והכתיבה, כלומר כאשר ה- rw במצב של '1' אז מתבצעת פעולת כתיבה, כלומר ה- blc,blcb מקבלים את ערכם מה- bl כי הטרנזיסטורים של nmos במצב הולכה והטרנזיסטורים pmos בקיטעון, כאשר ה-rw ב- '0' אז מתבצעת פעולת קריאה , כלומר ה- pmos פעילים וה- nmos בקיטעון ולכן ה- blc,blcb נטענים ל- vdd עם הפרש קטן ביניהם בעקבות העובדה שאחד מהם מחובר ל- '0' בעקבות המצב ההפוך שלם בתוך תא הזיכרון כפי שהוזכר בחלק של ה- SRAM.

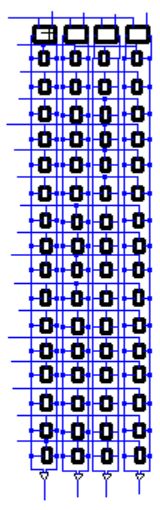


איור 6 הסימון של ה- pre charge



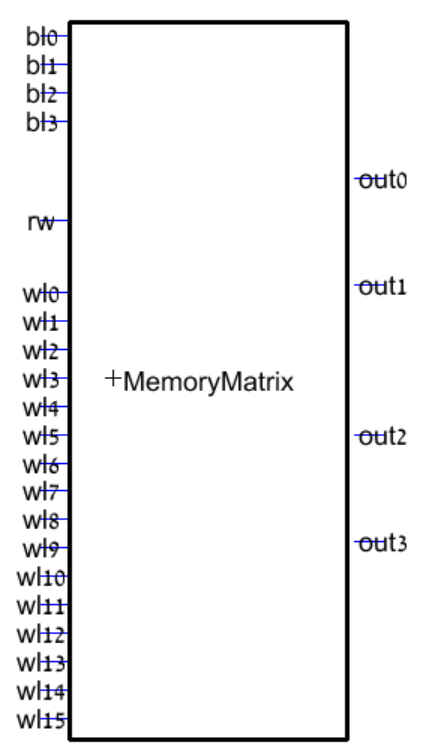
איור 7 מעגל המגבר והסימון שלו

מטרת המגבר היא להעצים את הפער של המוצאים ביחידת הזיכרון, כפי שנאמר במעגל ה- pre charge כאשר מתבצעת פעולת קריאה ההפרש בין המוצאים שבתוך יחידת הזיכרון (q', q) אמנם הפוכים אבל ה- blc, lblcb נטענים שניהם ל- vdd ולכן יש צורך להגדיל את הפער ביניהם על מנת לקבל '0' ו- '1' ולא '1' ו- '1'.

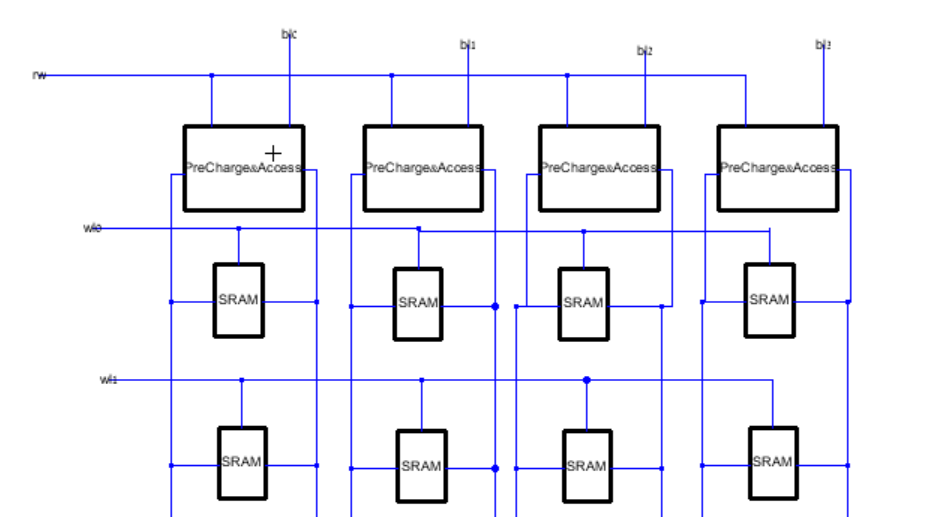


איור 8 מטריצת הזיכרון השלמה ב- schematic

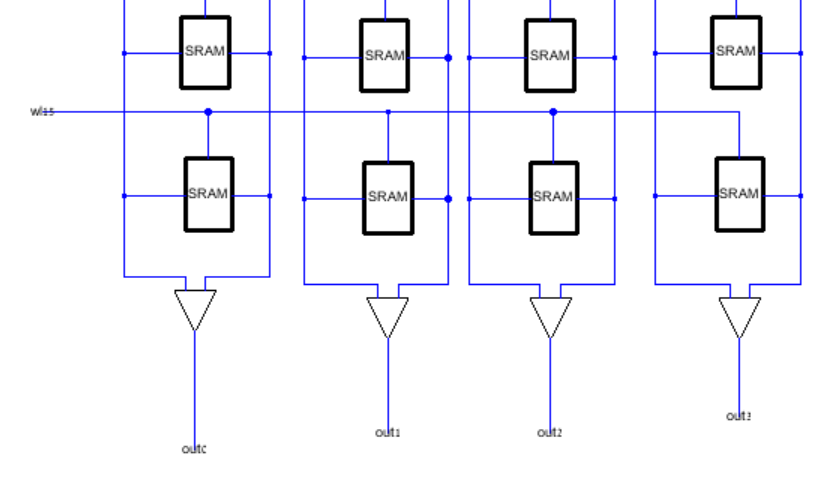
מעגל מטריצת הזיכרון של 64 ביט, המעגל השלם בנוגע לפעולת הזיכרון, כלומר יש לו את המערכת של ארבעת ה- pre charge שאחראים על סיווג פעולת הקריאה או הכתיבה, את יחידות הזיכרון של ה- SRAM המחלוקות לשורות של רביעיות ואת המגברים שנועדו לגרום למוצא של מטריצת הזיכרון להיות יותר מדויק.



איור 9 מראה ה- icon של מטריצת הזכרון

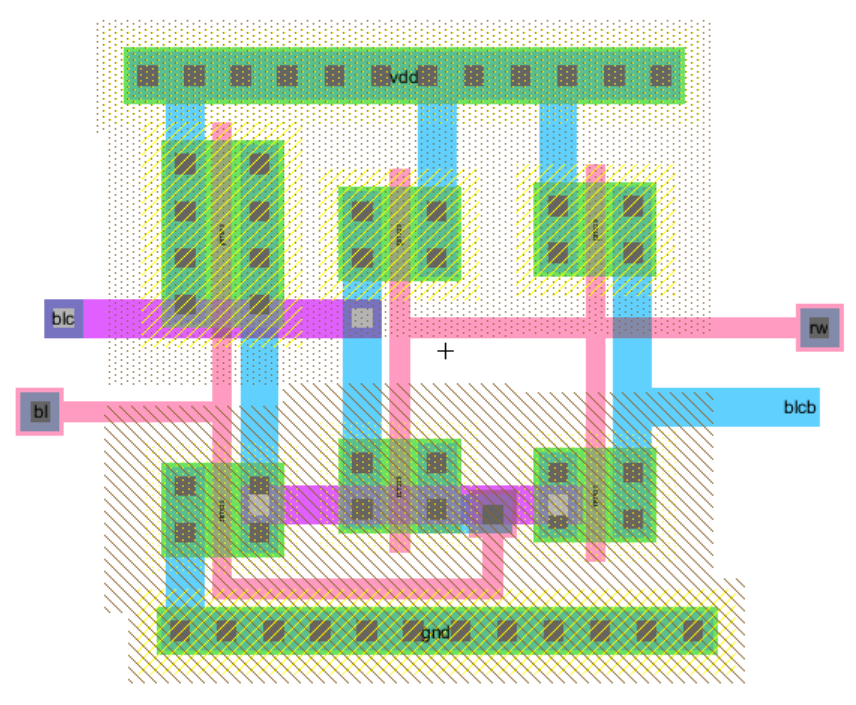


איור 10 החלק העליון של מטריצת הזיכרון

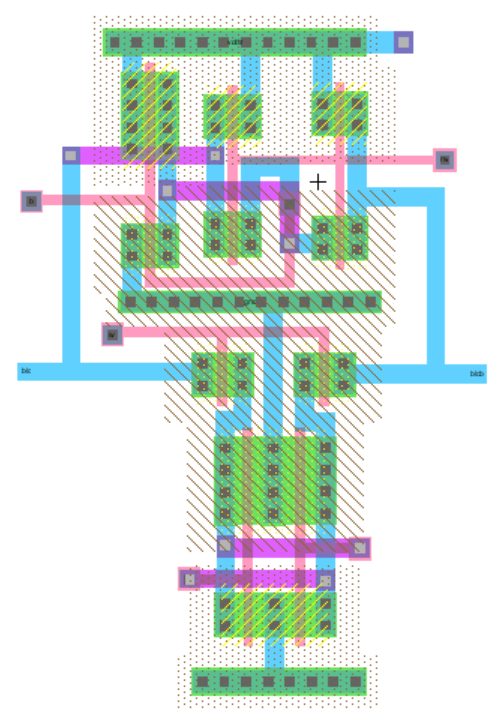


איור 11 החלק התחתון של מטריצת הזיכרון

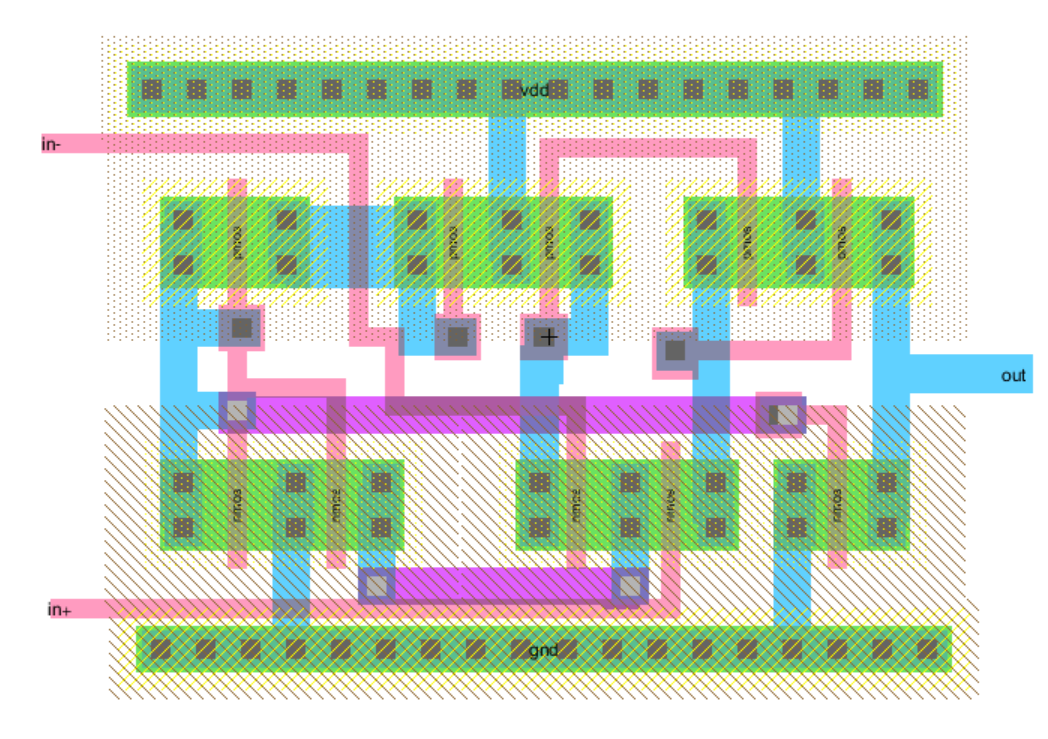
### Layout



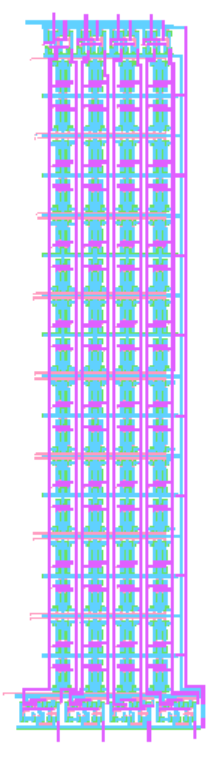
איור 12 מעגל ה- Pre Charge



איור 13 מעגל pre charge יחיד עם תא SRAM בודד

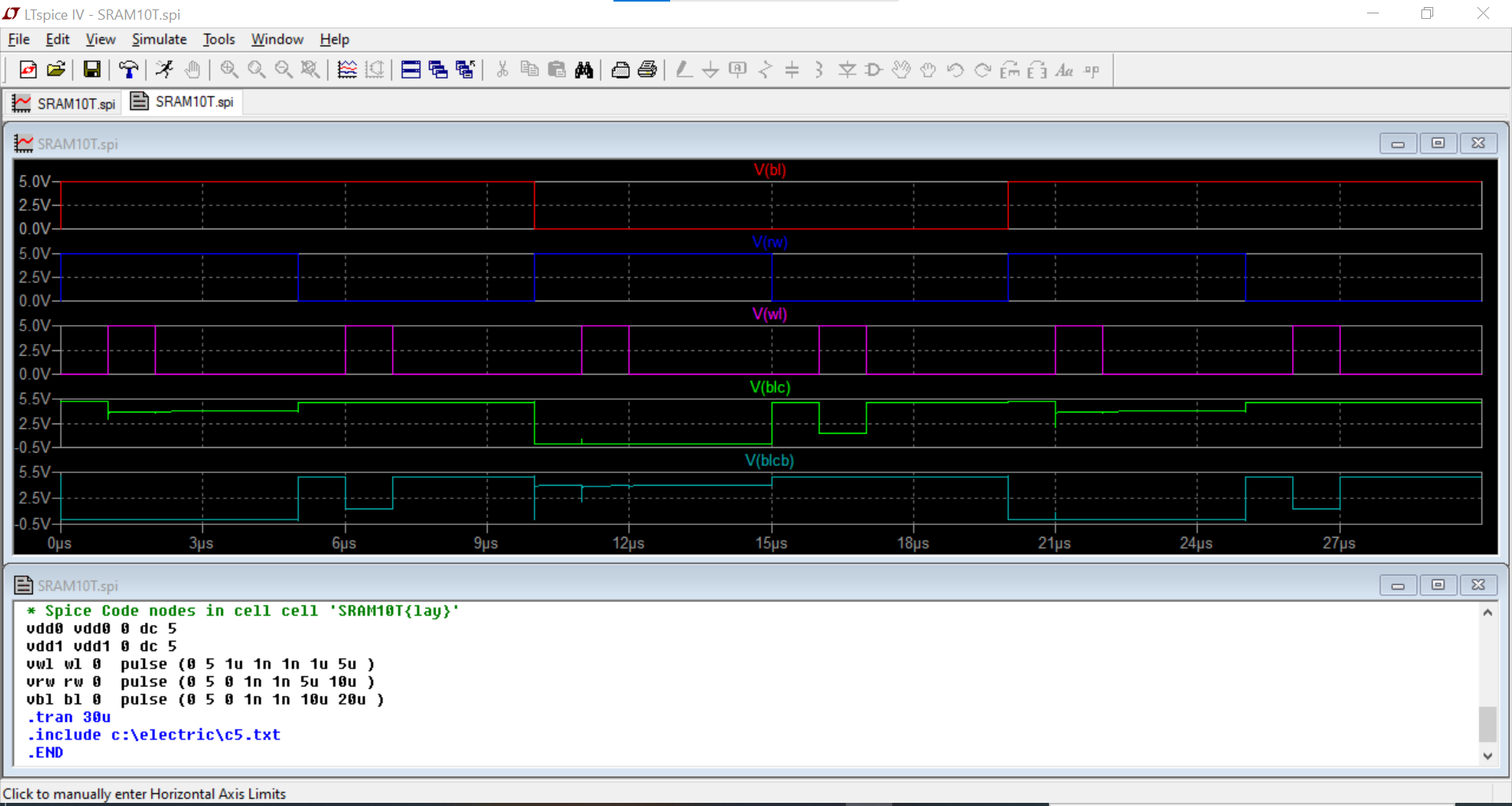


איור 14 המעגל של המגבר



איור 15 מעגל מטריצת הזיכרון

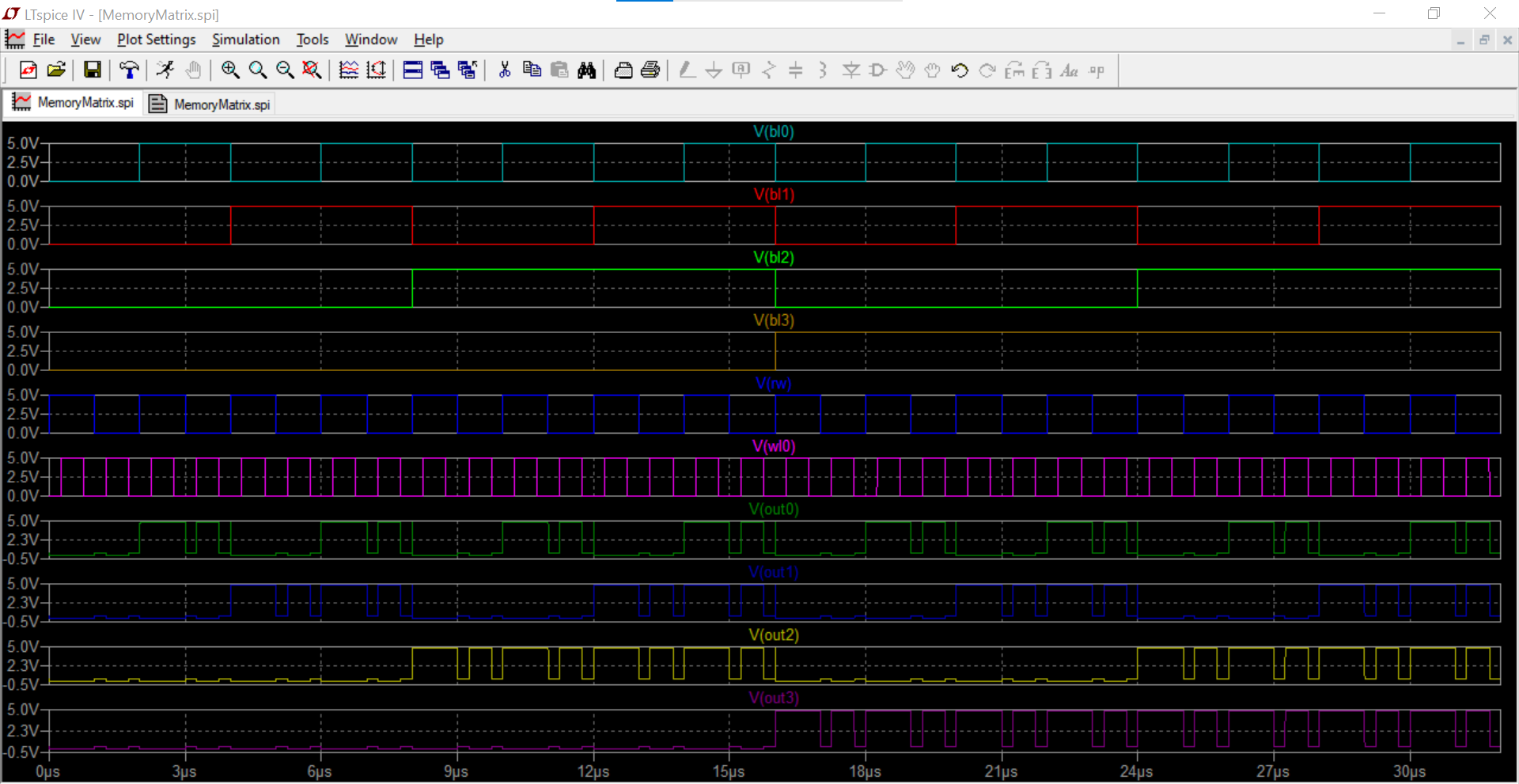
### סימולציה



איור 16 סימולצית ה- pre charge היחיד עם תא ה-SRAM

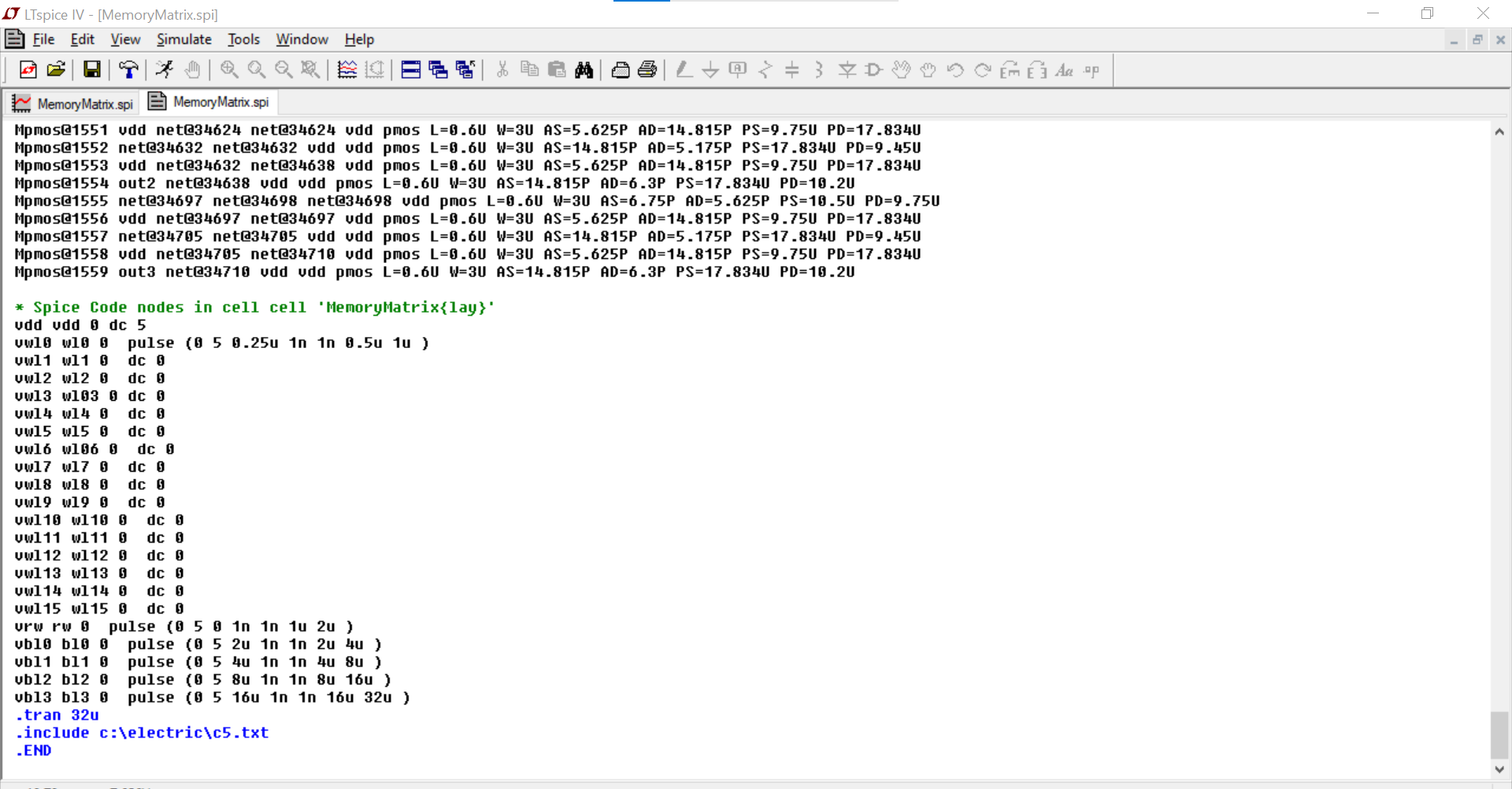
ניתן לראות את פעולת הקריאה והכתיבה של כל הווריאציות ( קריאה וכתיבה של '0' ו- '1').

כאשר מתבצעת פעולת הקריאה ה- blc,blcb נטענים על פי מעגל ה- pre charge וניתן לראות את ירידת המתח (פריקת הקבל) ב- blcb בעקבות הניסיון של מוצא ה- SRAM q' לתת '0' וירידת המתח ב- blc כאשר q מנסה להיות '0' .



איור 17 סימולצית מטריצת הזיכרון

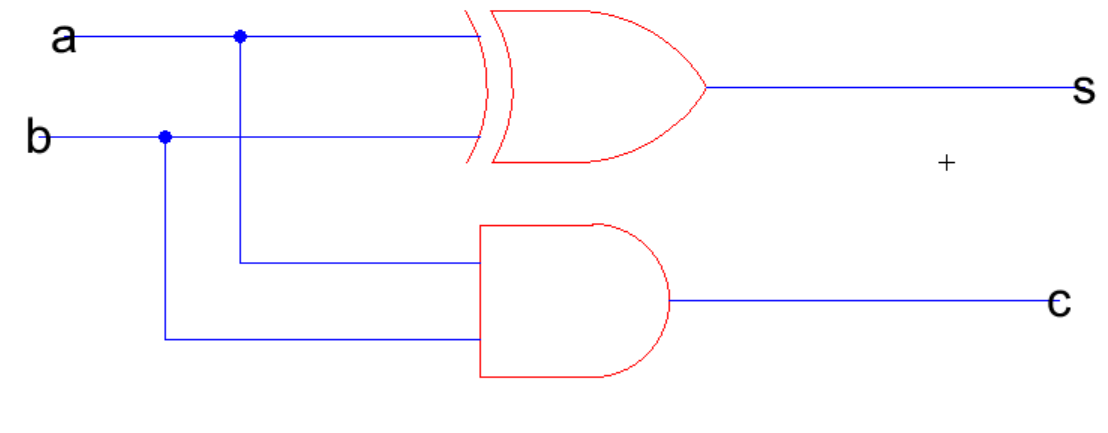
בסימולציית מטריצת הזיכרון המספרים bl0-bl3 המשמשים כאותות הכניסה, עולים לפי סדר הספירה הבינארית. בכל מספר מתבצעת פעולת כתיבה וקריאה המאופשרות לשורה w0.



איור 18 קוד סימולצית מטריצת הזיכרון

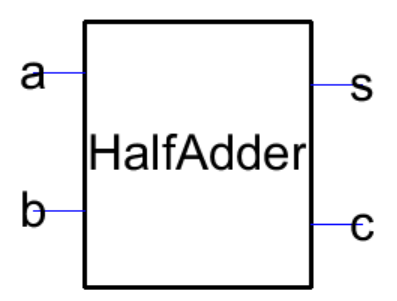
## Full adder and full subtractor

### Schematic

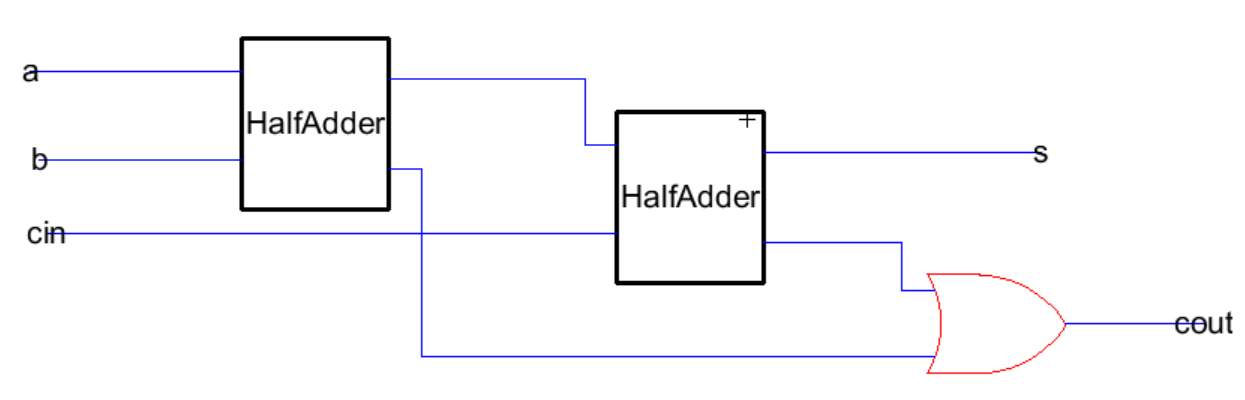


איור 19 Half Adder ב-schematic

יחידת ה- Half Adder היא היחידה הבסיסית של ה- Full Adder. יחידה זו משתמשת בשערים הלוגיים xorו- and. האותות a,b הם הכניסות של המספר, האות c הוא הסכום והאות c זה ה- carry.

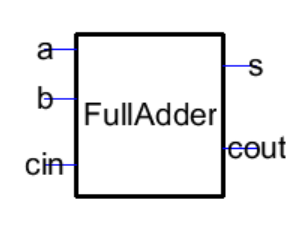


איור 20 ה- icon של ה-Half Adder

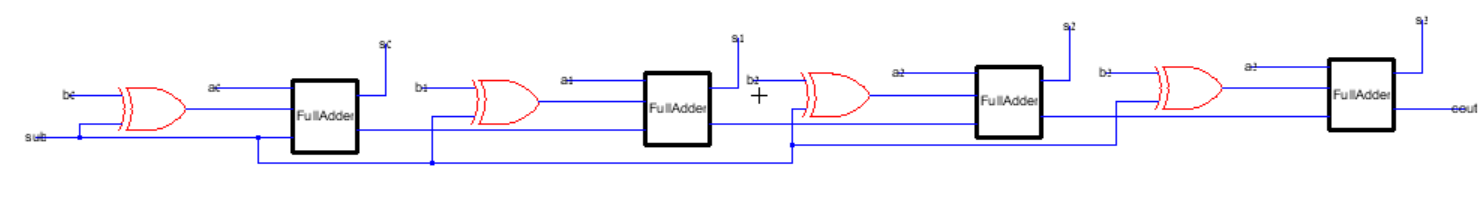


איור 21 Full dder

התוספת של מעגל ה-Full Adder על פני ה-Half Adder היא אות הכניסה של ה- carry, תוספת זו מאפשרת חיבור של מספר גדול יותר של ביטים על ידי שרשור של הרכיב.

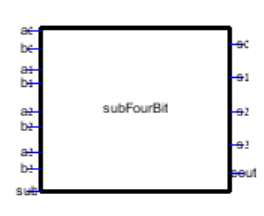


איור 22 ה- icon של ה- Full Adder



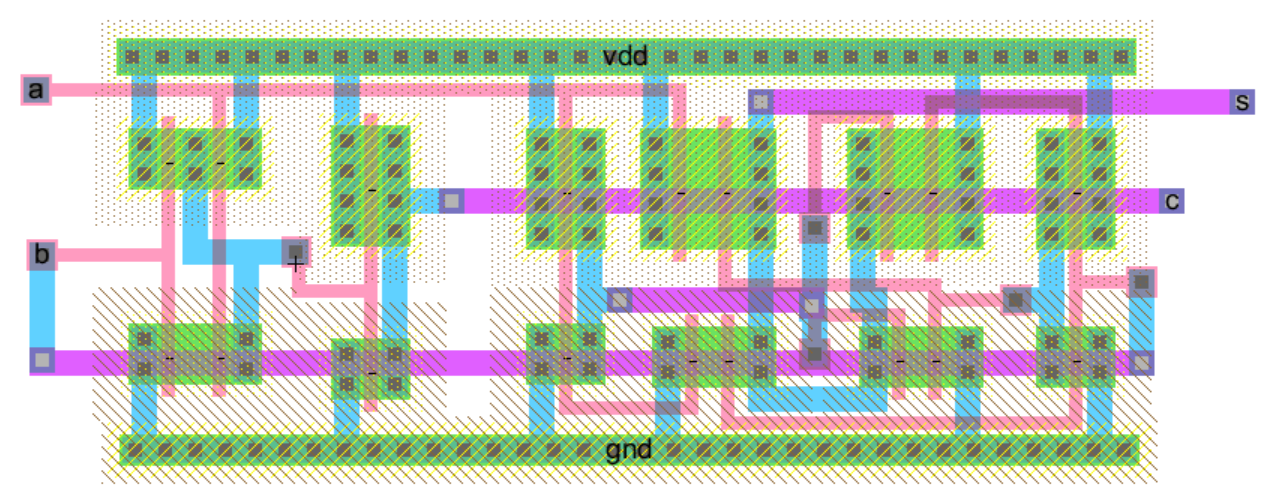
איור 23 Full adder and full subtractor

כפי שניתן לראות ממעגל זה, פעולות החיבור והחיסור מסווגות על פי ערך ה- sub, כלומר כאשר ה- sub במצב של '0' אז שער ה- xor אינו משנה את ערכי המספר b0-b3 ואז הפעולה המתבצעת בין המספר a0-a3 ל- b0-b3 היא פעולת חיבור, אבל כאשר הערך של ה- sub הוא '1' אז ה- xor הופך את המספר b0-b3 המתרחשת פעולת חיסור בין המספרים.

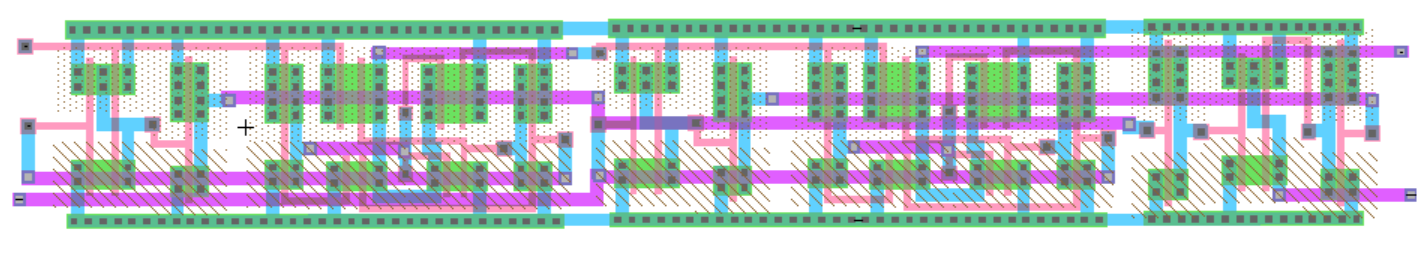


איור 24 Full adder and full subtractor icon

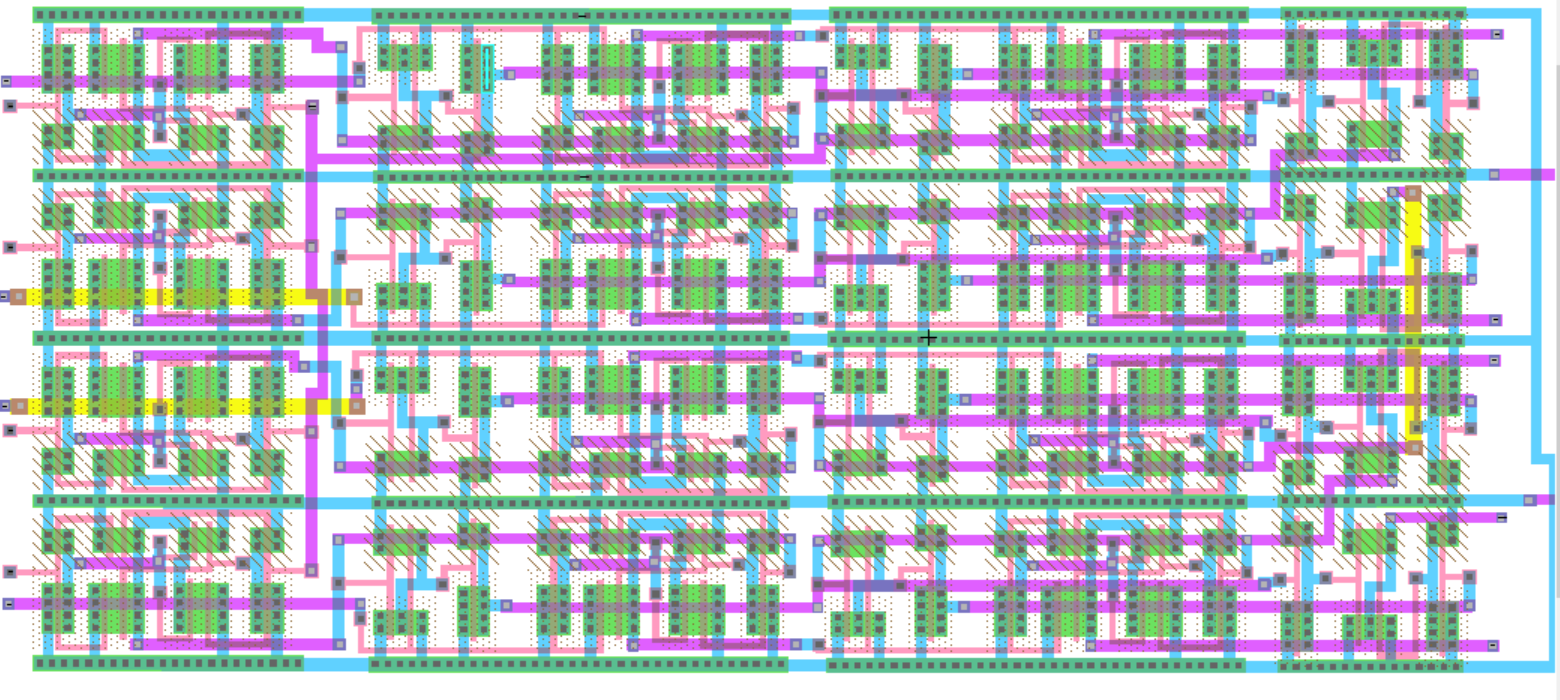
### Layout



איור 25 Half Adder Layout

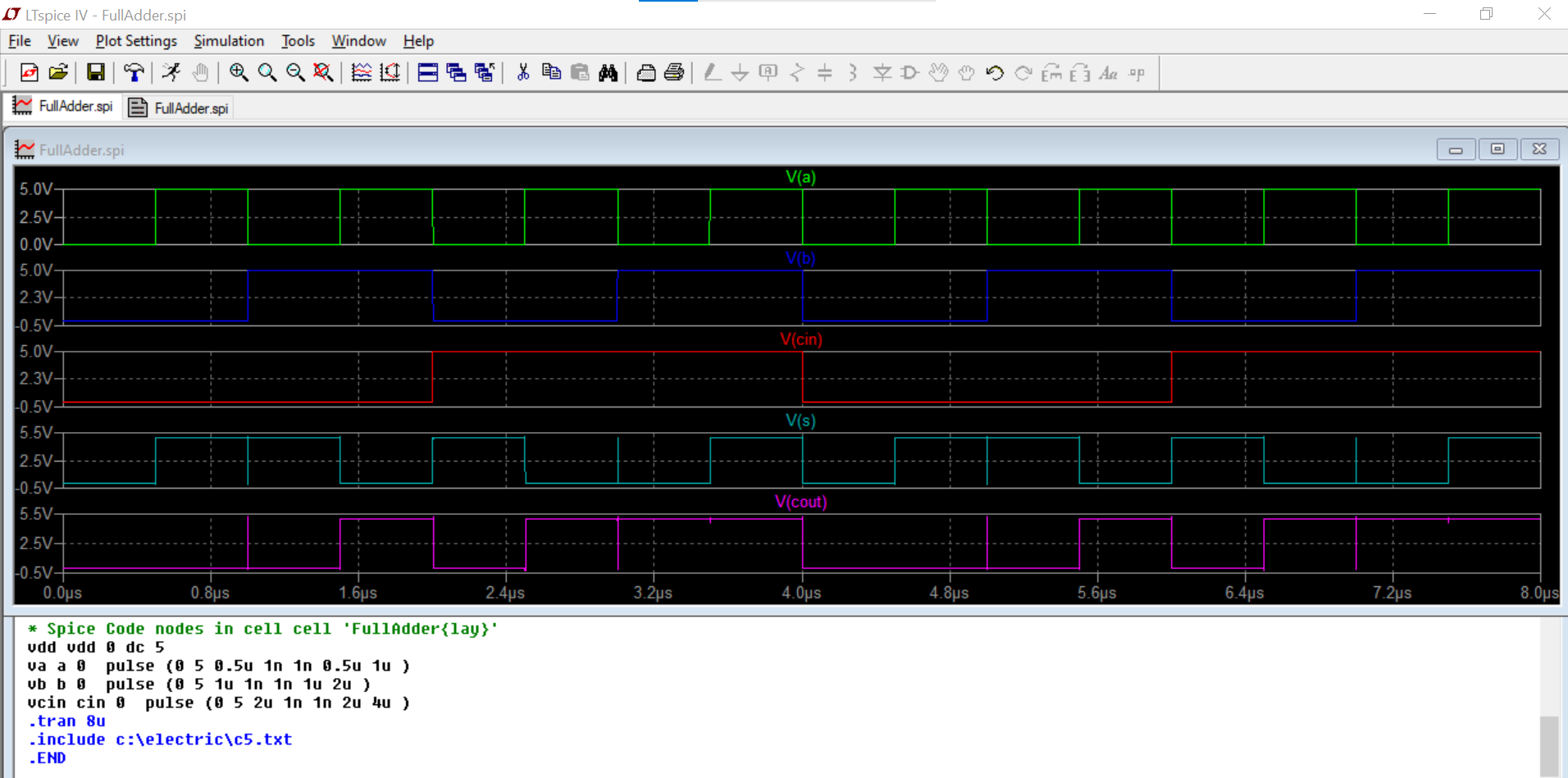


איור 26 Full adder Layout



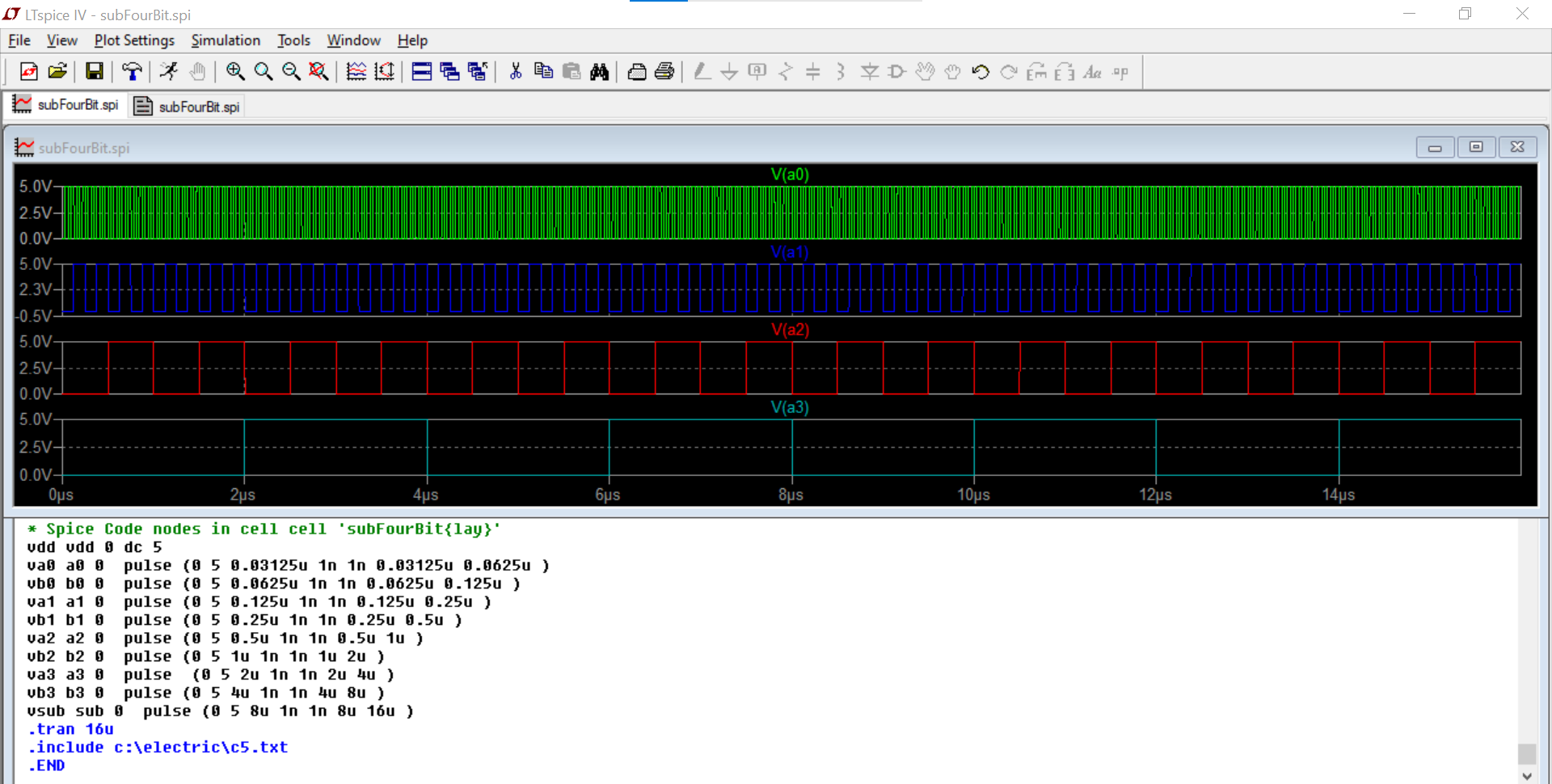
איור 27 Full adder and full subtractor Layout

### סימולציה



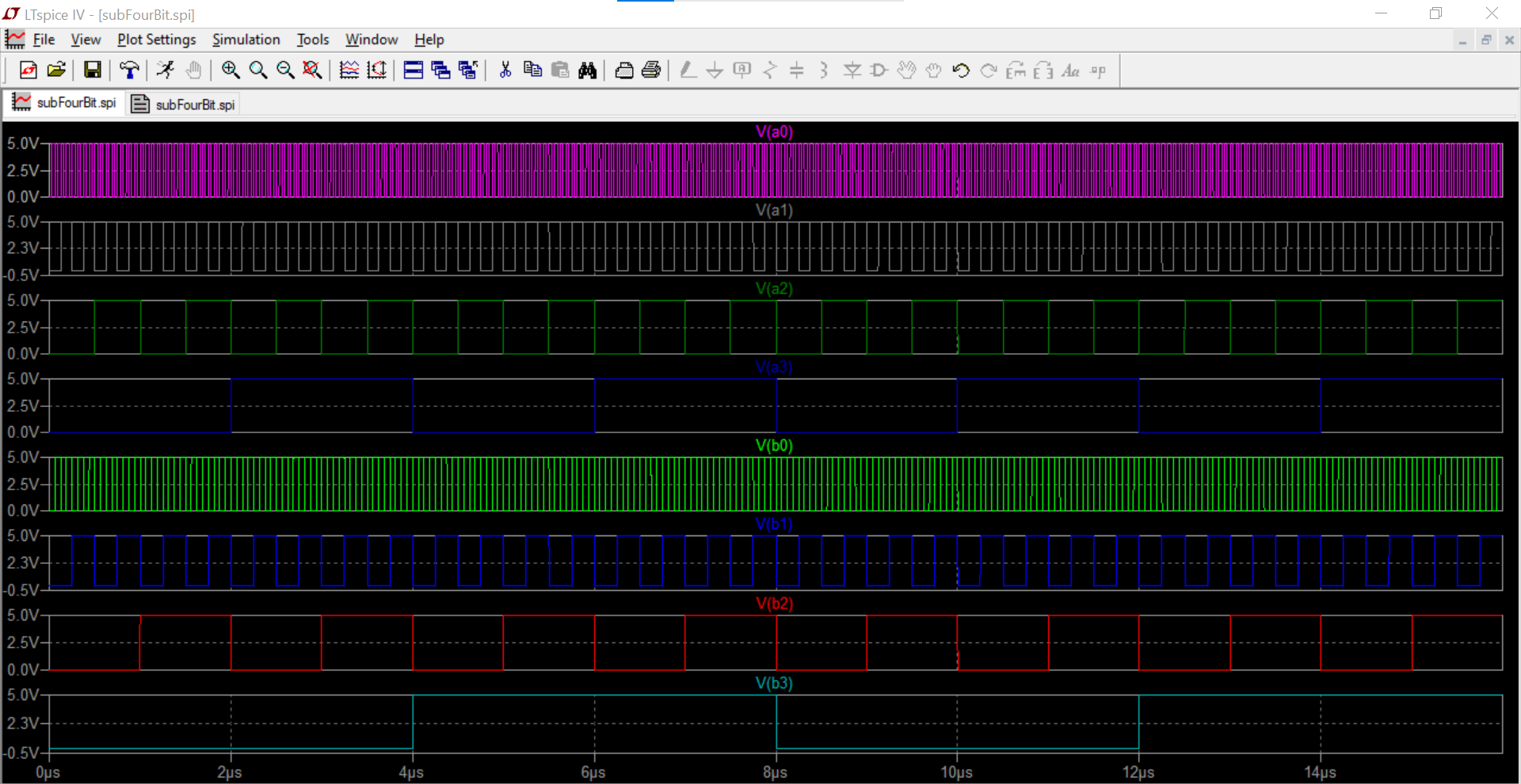
איור 28 סימולציה של ה-Full Adder עם שורת הקוד

הסימולציה מבטאת את טבלת האמת של רכיב ה- Full Adder ,כלומר המספרים a,b סופרים בסדר בינארי עולה וכך ניתן לראות את כל האופציות של המוצאים : cin, s, cout.

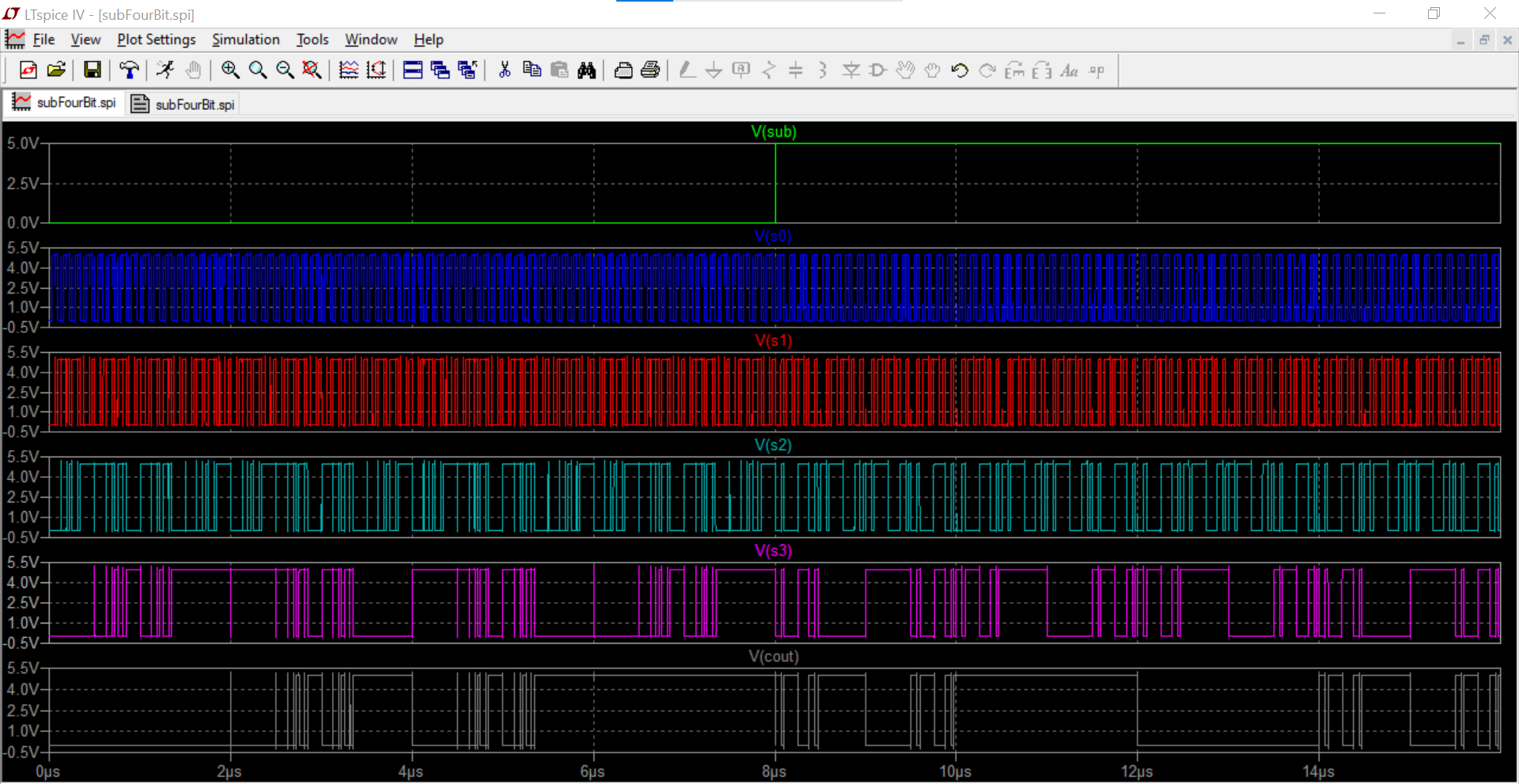


איור 29 הסימולציה של המספר הראשון ב- Full adder and full subtractor עם שורת הקוד

הספירה של מספרי הכניסות a0-a3 הם בקפיצות של 4 מחזורים אחד מהשני בסדר ספירה עולה וכך גם מספרי הכניסות b0-b3 כפי שמוצג באיור 30, כך שמתקבלת ספירה על פי סדר עולה באופן הבא בכל מחזור שלם של המספר מתבצע 2 מחזורים של המספר .

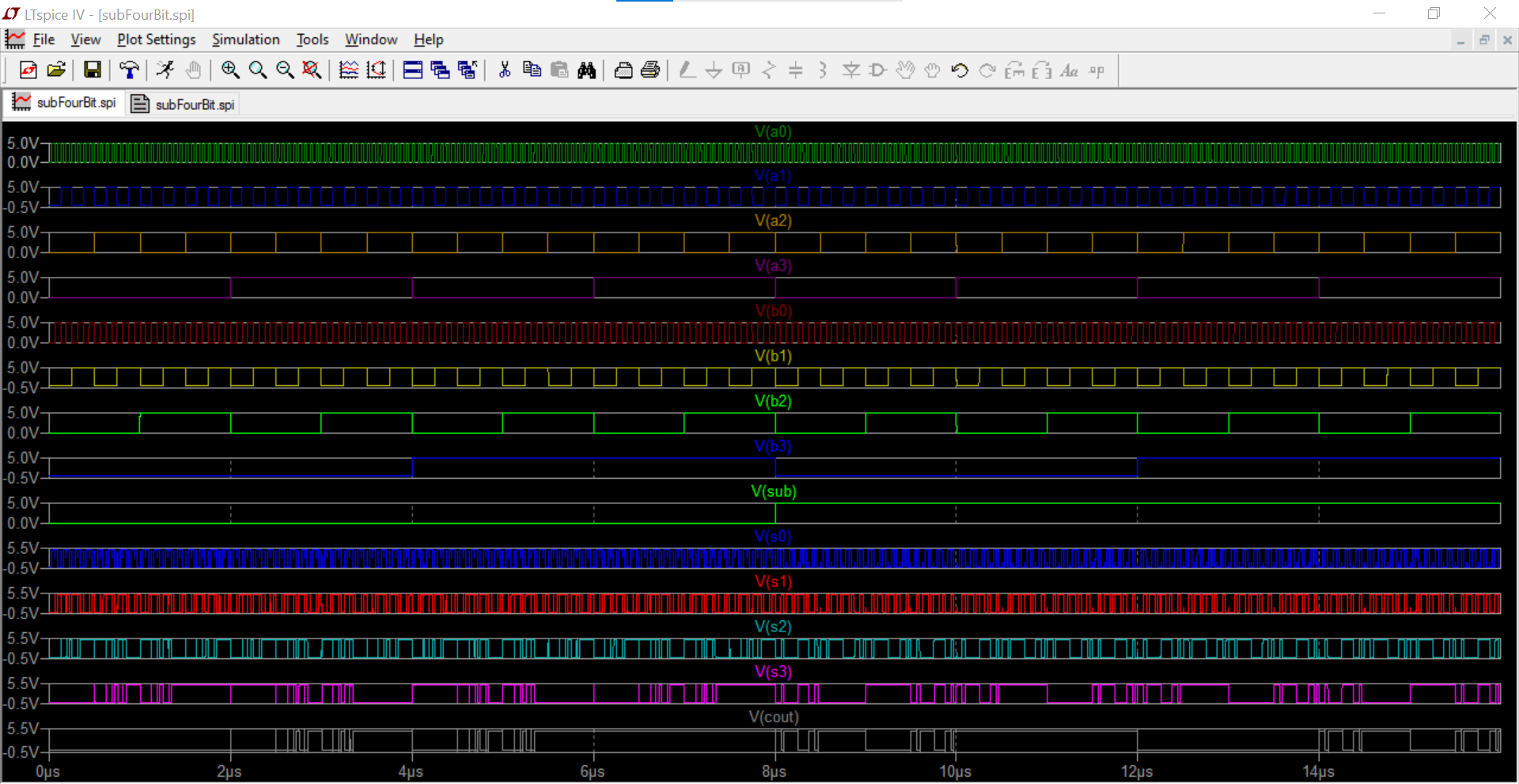


איור 30 הסימולציה של המספר הראשון והשני ב- Full adder and full subtractor

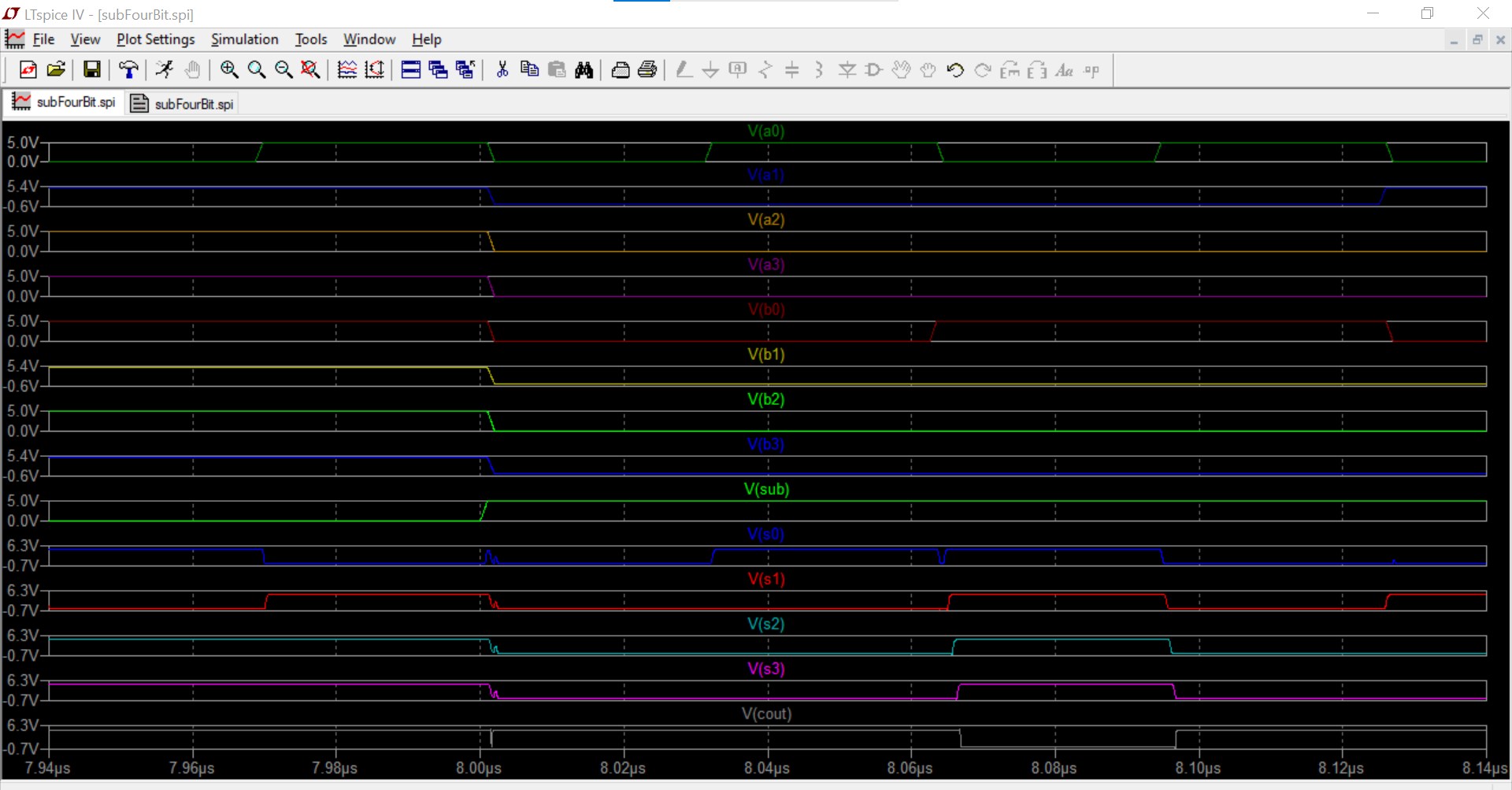


איור 31 הסימולציה של תוצאות החיבור והחיסור ב- Full adder and full subtractor

בסימולציה זו מופיעים אותות המוצא של הרכיבs0-s3,cout עם אות הכניסה sub האחראי לבחירת הפעולה החשבונית על פי אותות כניסת המספרים המתוארים באיור 30.



איור 32 סימולציה של המספרים והתוצאות ב- Full adder and full subtractor

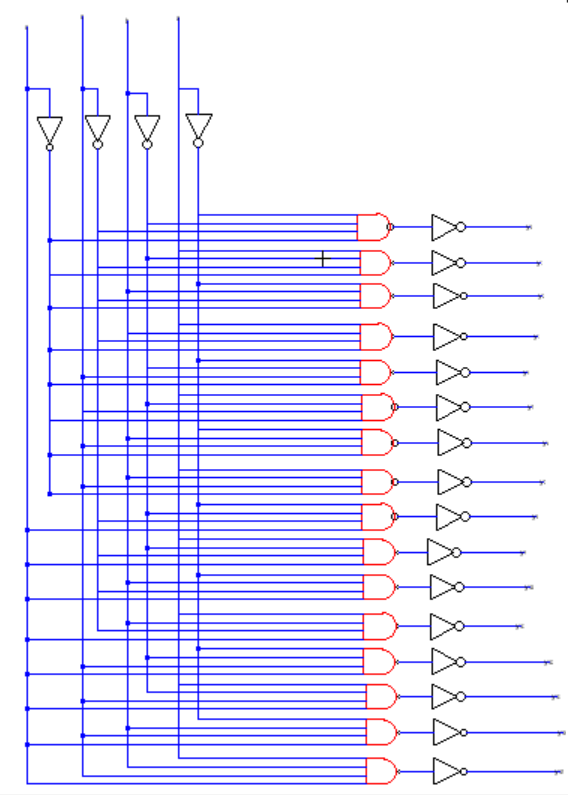


איור 33 סימולציה של המספרים והתוצאות ב- Full adder and full subtractor בתצורה מוגדלת

על מנת לראות בתצורה יותר ברורה את תוצאות הסימולציה בוצע הגדלה של מקטע ספציפי (החלק של המעבר מפעולת חיבור לפעולת חיסור). ניתן לראות בסימולציה זו את חיבור המספרים 1111a= עם 1111b= ואת התוצאה s=1110 עם carry של '1' ואת פעולת החיסור בין המספרים a=0001 ו- b=0000 עם התוצאה s=0001 ואת החיסור בין a=0000 ל- b=0001 אשר מוביל לתוצאה s=1111.

## Decoder

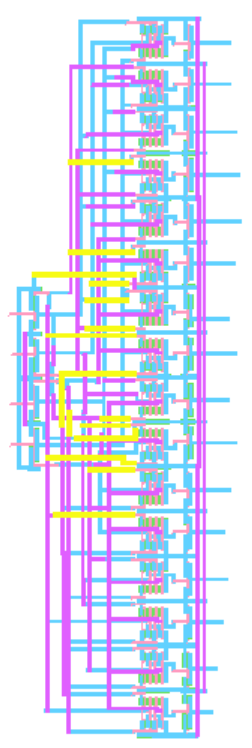
### Schematic



איור 34 Decoder 4 כניסות ו- 16 יציאות

על פי הסכמה ניתן להסיק שהמעגל מבצע פעולת פענוח של המספר בקלט, כלומר היציאה שתוציא בתור פלט של '1' לוגי מבין 16 המוצאים זאת היציאה שמבטאת את המספר שנכנס בתור קלט, כל שאר היציאות יוציאו '0' לוגי.

### Layout



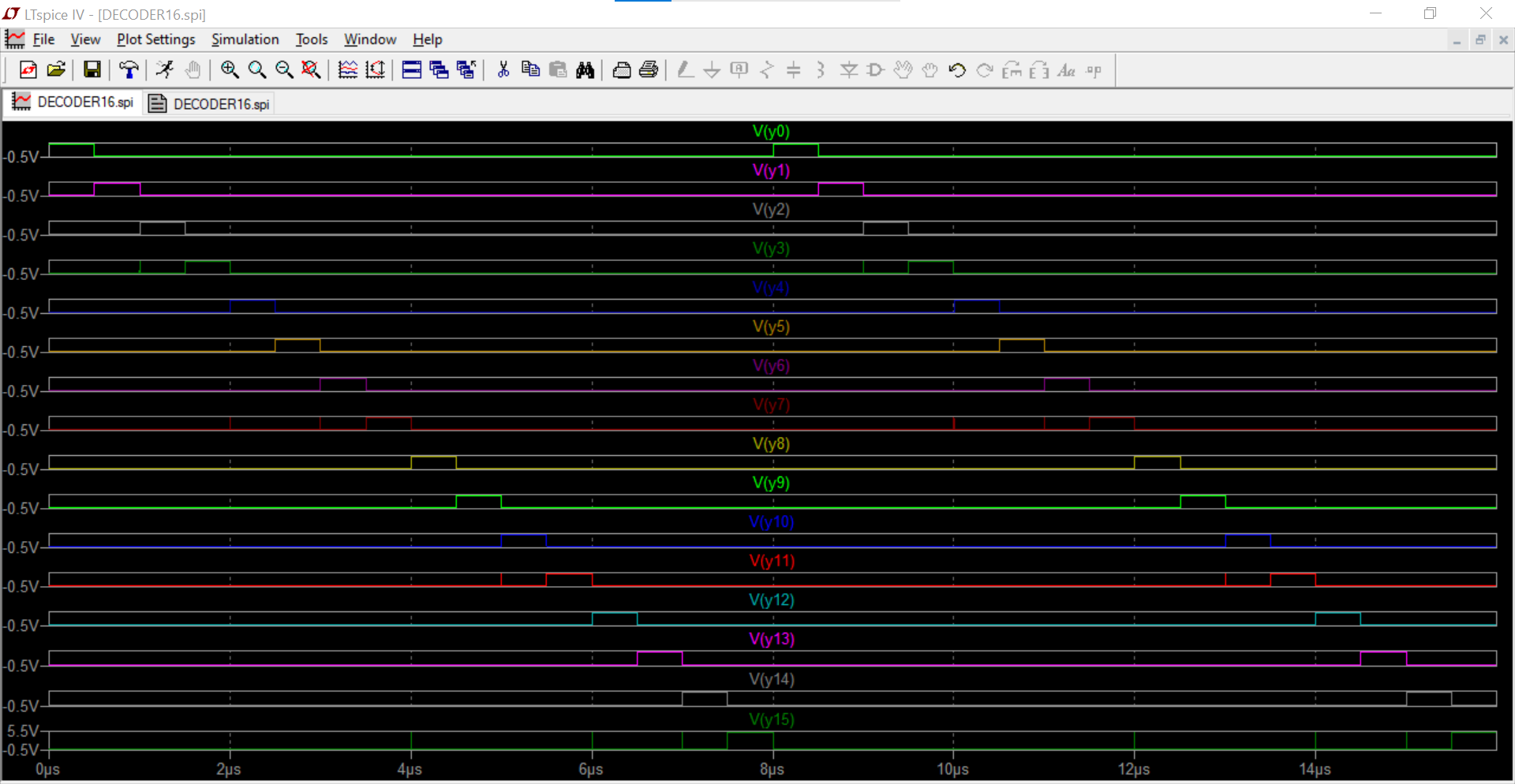
איור 35 decoder 4 כניסות ו- 16 יציאות

### סימולציה



איור 36 הסימולציה של כניסות ה- decoder עם שורות הקוד

הסימולציה מבטאת את טבלת האמת של הרכיב על פי ספירה בינארית בסדר עולה.



איור 37 תוצאות הסימולציה של מוצא ה- Decoder

כפי שניתן לראות בסימולציה, בכל פעם שנבחר מספר מסויים על פי כניסות הרכיב בהתאם לסדר הספירה המוצא הנבחר עולה ל-'1' וכל השאר ב-'0'.

## חיסור בשיטת המשלים ל-2

### משלים ל-2

שיטת המשלים ל־2 היא שיטה לייצוג מספרים עם סימן בבסיס בינארי. בשיטה זאת הסיבית הגבוהה ביותר (MSB - Most Significant Bit) מייצגת את הסימן של המספר (חיובי או שלילי) ושאר הספרות מייצגות את ערך המספר (בצורה שונה מייצוג רגיל אם הוא שלילי). שיטה זו מקובלת במחשבים לייצוג בינארי של מספרים שעשויים להיות שליליים, כיוון שייצוג זה מאפשר להשתמש בדרך חישוב אחת לכל סוגי המספרים הבינאריים, כלומר אין צורך בעוד יחידות חישוב במיוחד למספרים שלילים.

בשיטה משתמשים בייצוג הרגיל (ללא סימן) של מספרים בתחום מסוים, למשל מספרים בני 4 ביטים, "חותכים" את החלק העליון של התחום, ומדביקים אותו אל החלק התחתון, באותו סדר. כלומר, אם בייצוג בינארי רגיל המספרים הגבוהים ביותר הם 8–15, בשיטת המשלים ל-2 הייצוג שמשמש את המספרים האלה ישמש עבור המספרים שבין מינוס 8 לבין מינוס 1, באותו סדר. הקידוד עבור 8 (1000) ישמש עבור קידוד מינוס שמונה (16-8=8), הקידוד עבור 9 (1001) ישמש עבור מינוס 7 (16-9=7), וכן הלאה עד הקידוד עבור 15 (1111) שישמש עבור מינוס 1 (16-15=1). כלומר הקידוד עבור 2 בחזקת מספר הביטים פחות X, הוא הקידוד עבור מינוס X.

על מנת לייצג מספר בן n סיביות בשיטת המשלים ל-2 יש להפריד את הסיבית השמאלית ביותר משאר המספר.

### חיסור

על מנת לבצע חיסור, נבצע חיבור של המספר המחוסר עם הנגדי של המספר המחסר כאשר שניהם מיוצגים בשיטת המשלים ל-2 (חיבור מספר שלילי זהה לחיסור מספר חיובי)

לדוגמה: (5-) = 12 – 7

ראשית נמיר את 12- בשיטת המשלים ל-2 (0100 1111) ולאחר מכן נבצע:

-5

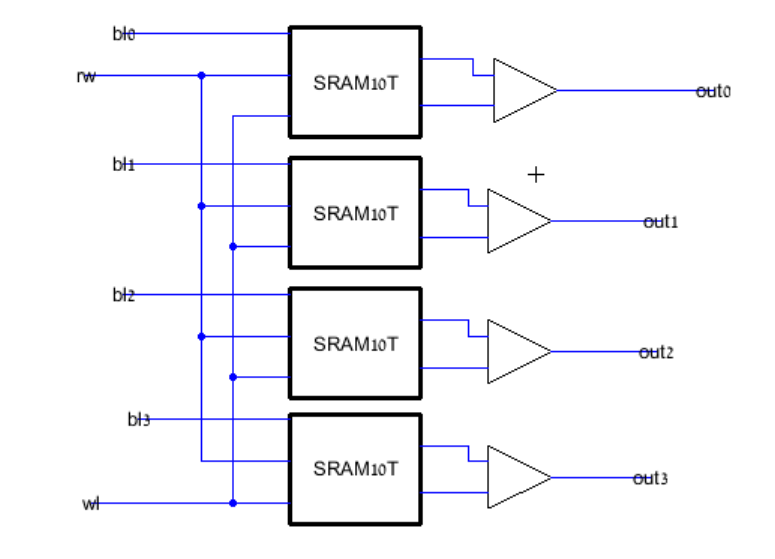
-12

+7

0000‎ 0111+1111‎ 0100=1111‎ 1011

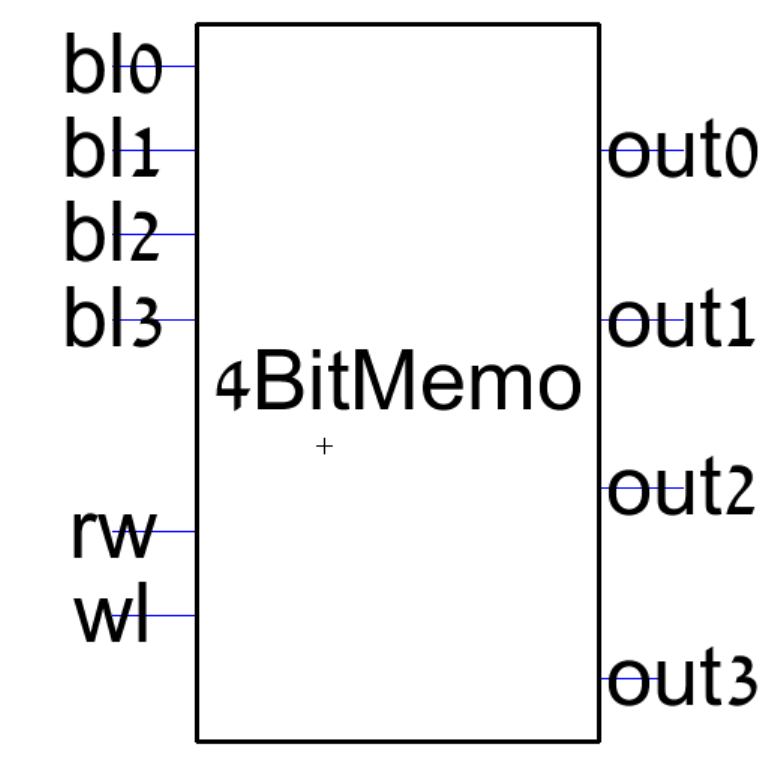
# מחשבון

## Schematic

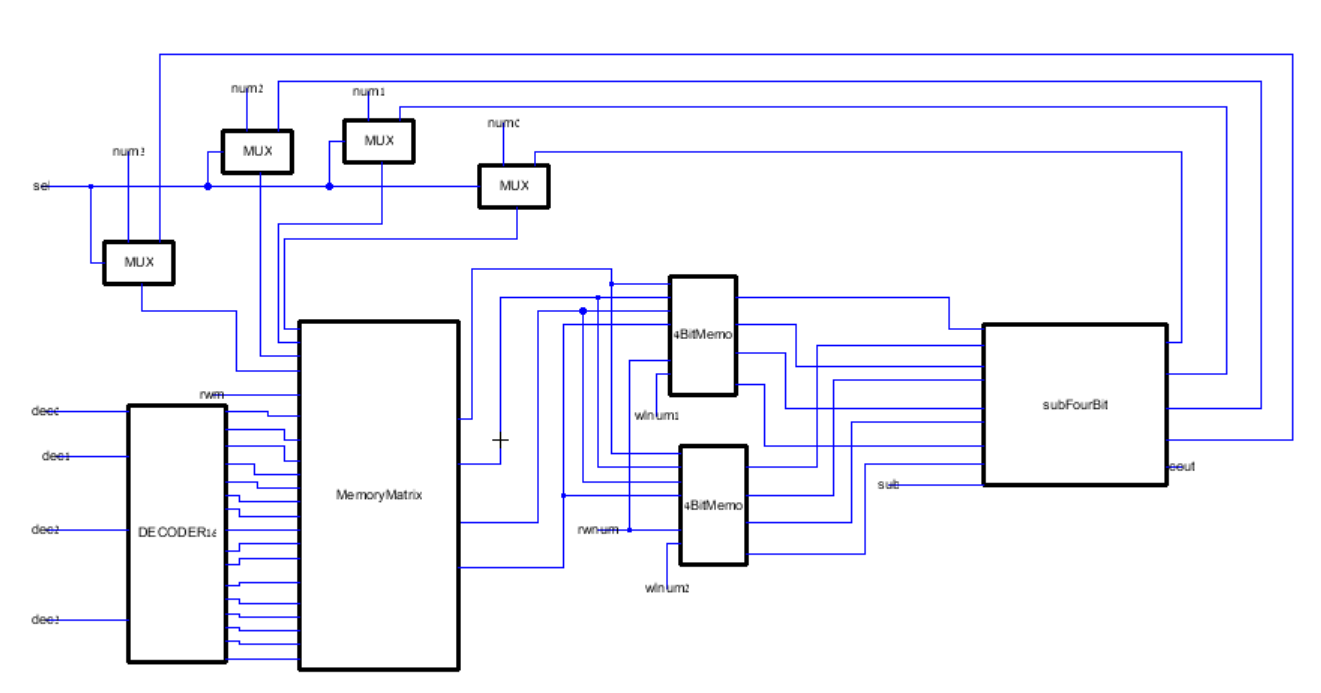


איור 38 רכיב שורת הזיכרון הזמינה המיועדת לכתיבת המספרים בו זמנית

רכיב זה נועד לאפשר אחסון מספר בעל 4 ביטים ממטריצת הזיכרון על מנת שיהיה ניתן לכתוב 2 מספרים בו זמנית לרכיב ה- Full adder and full subtractor כי מטריצת הזיכרון מאפשרת קריאה של שורה בודדת כל פעם ועל מנת לחבר 2 מספרים ב-Full adder and full subtractor יש צורך שהם יכנסו אותו הזמן.

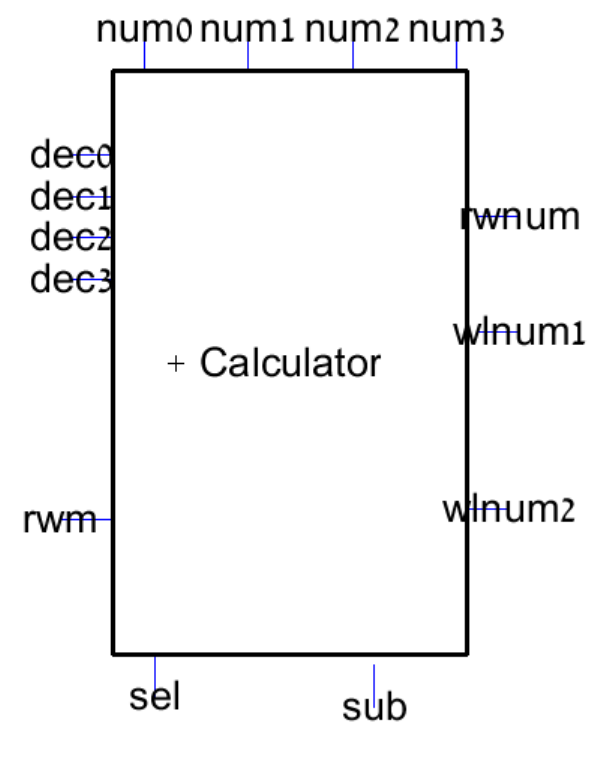


איור 39 סימון רכיב יחידת הזיכרון הזמינה



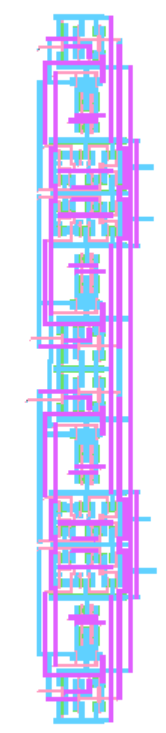
איור 40 סכמת המעגל הכולל של המחשבון

מעגל המחשבון בנוי באופן הבא: האותות dec0-dec3 מחוברים לרכיב decoder של 4 כניסות ו- 16 יציאות שתפקידן לבחור את שורת הזיכרון. במטריצת הזיכרון יש את אות הכניסה rwm שאחראי על הכתיבה לרכיב כאשר הוא ב-'1' או על קריאה מהרכיב כאשר הוא ב-'0' , האותות שאחראיים על התוכן שנכנס למטריצת הזיכרון במקרה של קריאה הם היציאות של ה- MUX. במקרה של קריאה ממטריצת הזיכרון ישנם שני שורות זיכרון שאחראיות על האחסון של המספרים שעליהם מבצעים את פעולת החיבור או החיסור, האות rwnum אחראי על הקריאה או הכתיבה של המספרים והאותות wlnum1,wlnum2 אחראיים על אפשור פעולת הקריאה או הכתיבה של כל מספר בנפרד. המספרים מחוברים לרכיב Full adder and full subtractor שמבצע את פעולת החיבור או החיסור בהתאם לאות sub, כלומר כאשר sub מקבל '1' אז מתבצעת פעולת חיסור וכאשר הוא מקבל '0' אז מתבצעת פעולת חיבור. מוצא ה- Full adder and full subtractor מחובר לכניסה של ה- MUX כך שעל ידי האות sel ניתן לבחור איזה מספר יכנס למטריצת הזיכרון בעת פעולת הקריאה, כלומר כאשר sel ב- '0' אז המספר מהקוד של הסימולציה נבחר בתור האותות num0-num3 וכאשר sel הוא '1' אז המספר שבמוצא ה- Full adder and full subtractor נבחר.

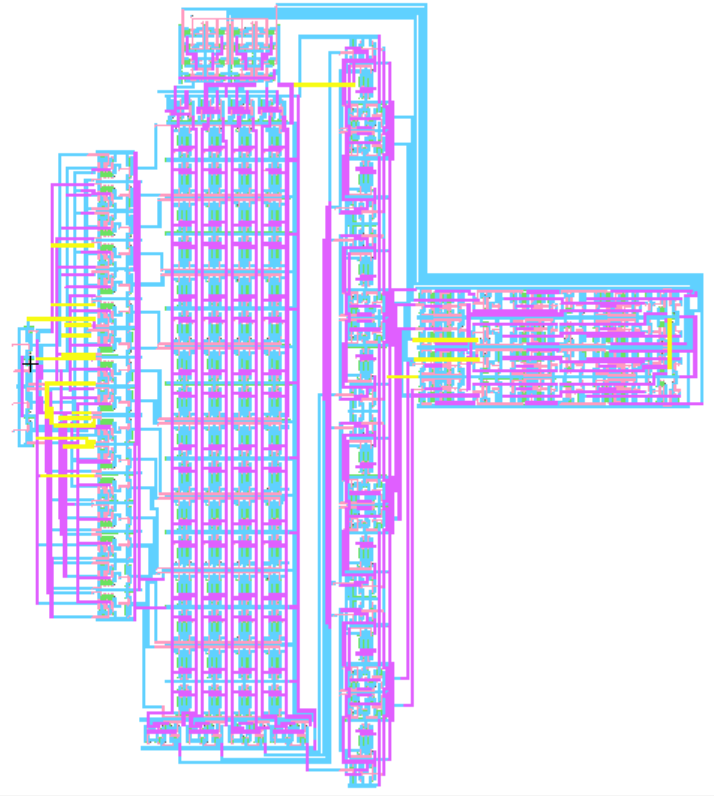


איור 41 icon של מערכת המחשבון

## Layout

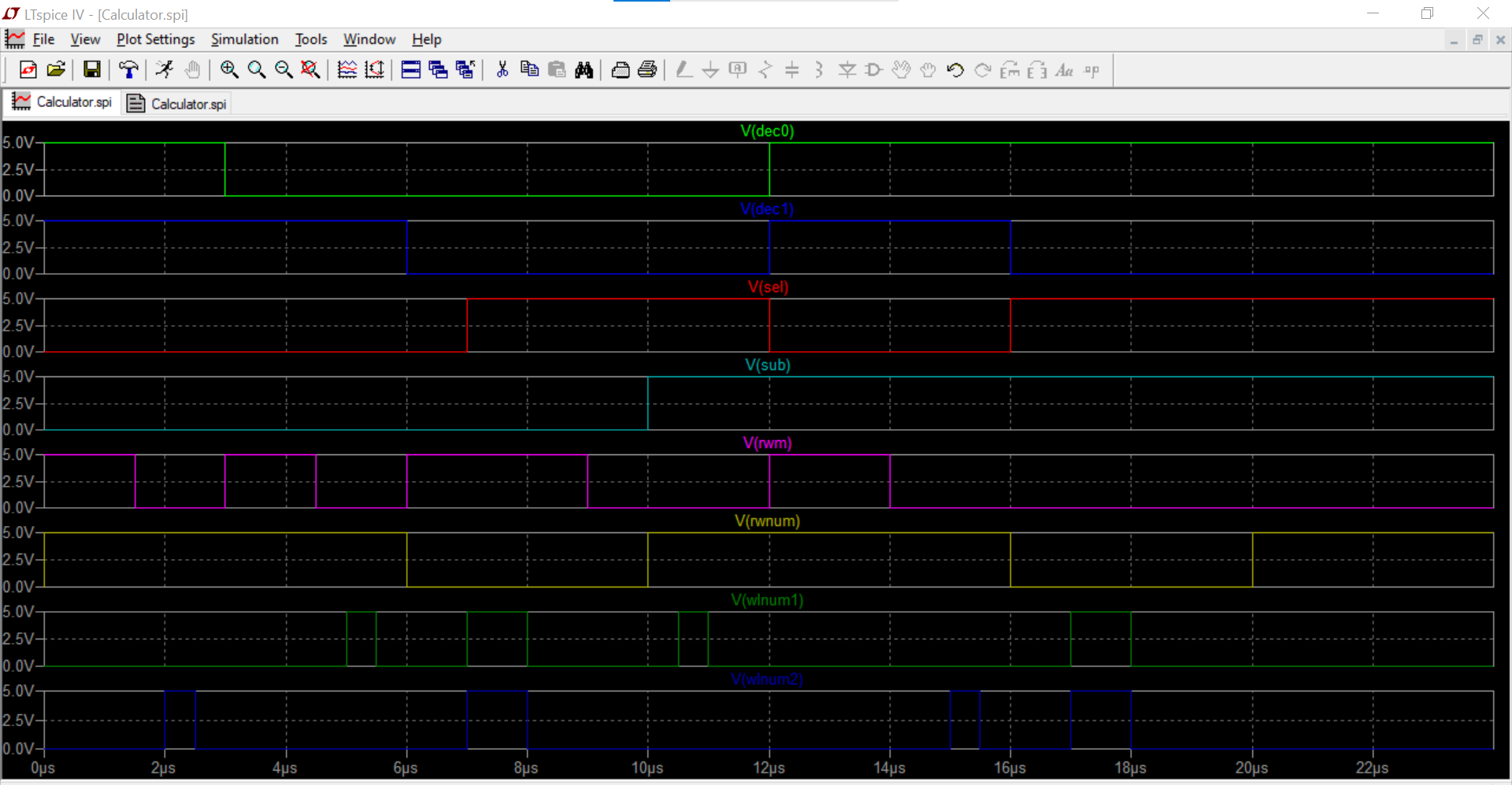


איור 42 רכיב שורת הזיכרון הזמינה



איור 43 המעגל הכללי של מערכת המחשבון

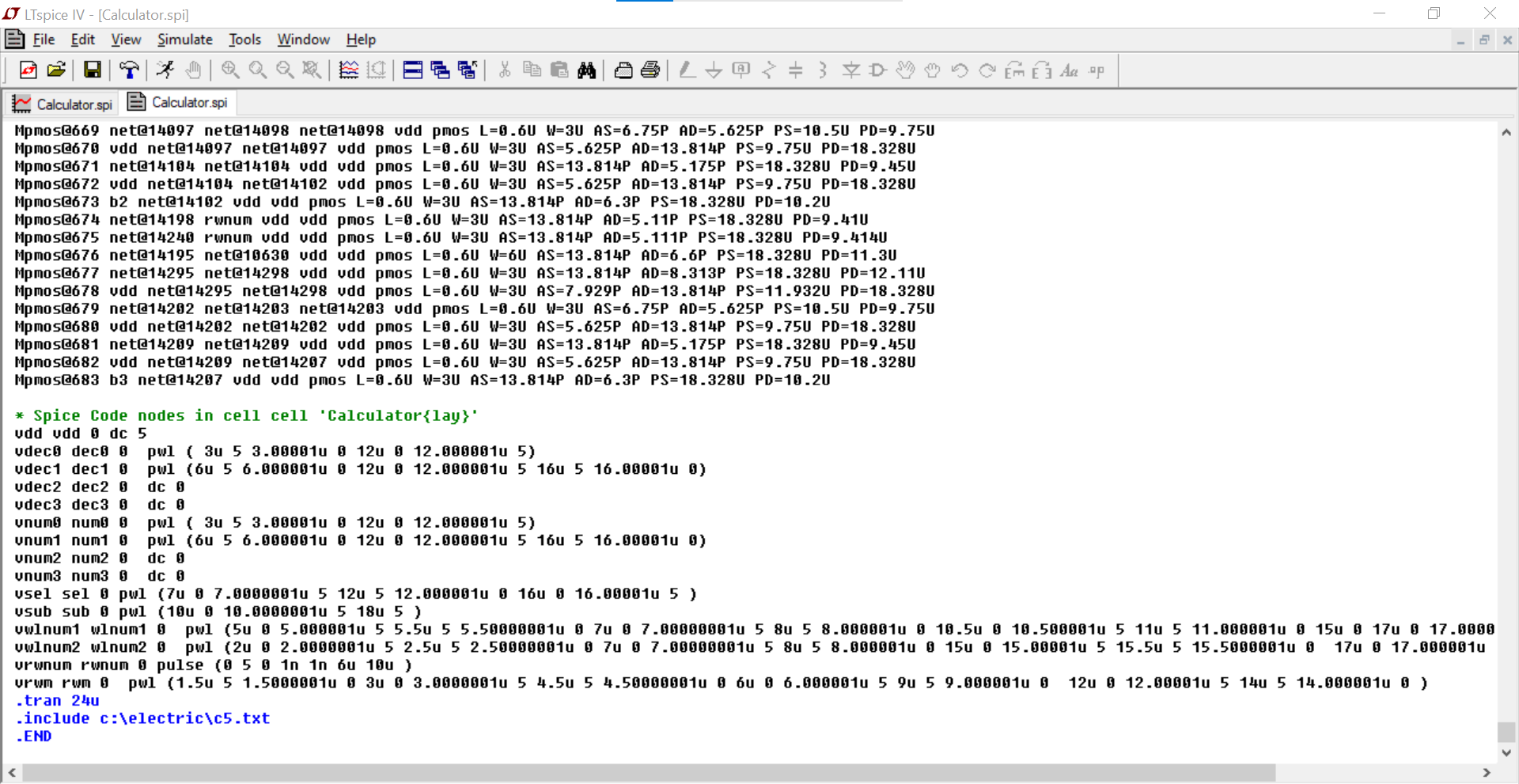
## סימולציה



איור 44 פעולות חיבור וחיסור במערכת המחשבון הכוללות אכסון במטריצת הזיכרון

בסימולציה זו ניתן לראות את פעולת חיבור של המספרים 0011 המאוחסן בשורה 0011 במטריצת הזיכרון עם המספר 0010 המאוחסן בשורה 0010 במטריצת הזיכרון, תוצאת החיבור מאוחסנת בשורה 0000. בנוסף מתבצעת פעולת חיסור בין תוצאת החיבור הנ"ל לבין המספר 0011 המאוחסן בשורה 0011 ותוצאת החיסור מאוחסנת בשורה 0001.

### תהליך הסימולציה של המחשבון



איור 45 קוד התוכנית של המחשבון

בחירת מספר שורת הזיכרון במטריצת הזיכרון על ידי ה- decoder (מוצג בתור dec) והמספר שנכנס אליה מהקוד דרך ה- MUX (מוצג בתור num) הוגדרו בתור אותם המספרים והם מסונכרנים באותם הזמנים על מנת להציג פחות אותות בסימולציה, כמו כן המספרים dec2,dec3,num2,num3 קבועים על 0V מהסיבה הנ"ל.

בתחילת הסימולציה מתבצעת כתיבת המספרים 0011 ו-0010 מקוד הסימולציה למטריצת הזיכרון (sel מקבל '0') לשורות נפרדות (המספרים 0011 ו- 0010 נרשמים כאמור לשורות הזיכרון בהתאם למספרם) והם נכתבים לשורות הזיכרון המוכנות לקריאת המספרים בו זמנית לרכיב ה- Full adder and full subtractor. לאחר כתיבת שני המספרים בנפרד לכל שורת הזיכרון הזמינות אות ה-sel של בחירת ה- MUX עולה ל-'1' ומטריצת הזיכרון מבצעת קריאה לשורה 0000 המיועדת לאחסון תוצאת פעולת החיבור ואז מתבצעת על שורות הזיכרון הזמינות פעולת קריאה במקביל לרכיב ה- Full adder and full subtractor כאשר אות ה- sub במצב '0' על מנת לבצע פעולת חיבור. תוצאת החיבור שהיא 0101 המאוחסנת במטריצת הזיכרון ונרשמת לשרות הזיכרון הזמינה ובשורה השנייה נרשם מספר 0011 חדש מהספקים של שורת הקוד על ידי שינוי ה- sel בחזרה ל-'0'. אות ה-sub שונה ל-'1' על מנת לבצע פעולת חיסור בין תוצאת החיבור 0101 לבין המספר 0011 והתוצאה 0010 נרשמת בשורה 0001 במטריצת הזיכרון.

# רשימת איורים

[איור 1 תא SRAM בודד ב- schematic 3](#_Toc78488379)

[איור 2 מראה ה- icon של ה- SRAM 4](#_Toc78488380)

[איור 3 תא SRAM בודד 4](#_Toc78488381)

[איור 4 סימולציית ה- SRAM עם שורת הקוד 5](#_Toc78488382)

[איור 5 מעגל ה-pre charge 5](#_Toc78488383)

[איור 6 הסימון של ה- pre charge 6](#_Toc78488384)

[איור 7 מעגל המגבר והסימון שלו 6](#_Toc78488385)

[איור 8 מטריצת הזיכרון השלמה ב- schematic 7](#_Toc78488386)

[איור 9 מראה ה- icon של מטריצת הזכרון 7](#_Toc78488387)

[איור 10 החלק העליון של מטריצת הזיכרון 8](#_Toc78488388)

[איור 11 החלק התחתון של מטריצת הזיכרון 8](#_Toc78488389)

[איור 12 מעגל ה- Pre Charge 9](#_Toc78488390)

[איור 13 מעגל pre charge יחיד עם תא SRAM בודד 9](#_Toc78488391)

[איור 14 המעגל של המגבר 10](#_Toc78488392)

[איור 15 מעגל מטריצת הזיכרון 10](#_Toc78488393)

[איור 16 סימולצית ה- pre charge היחיד עם תא ה-SRAM 11](#_Toc78488394)

[איור 17 סימולצית מטריצת הזיכרון 11](#_Toc78488395)

[איור 18 קוד סימולצית מטריצת הזיכרון 12](#_Toc78488396)

[איור 19 Half Adder ב-schematic 12](#_Toc78488397)

[איור 20 ה- icon של ה-Half Adder 13](#_Toc78488398)

[איור 21 Full dder 13](#_Toc78488399)

[איור 22 ה- icon של ה- Full Adder 13](#_Toc78488400)

[איור 23 Full adder and full subtractor 13](#_Toc78488401)

[איור 24 Full adder and full subtractor icon 14](#_Toc78488402)

[איור 25 Half Adder Layout 14](#_Toc78488403)

[איור 26 Full adder Layout 14](#_Toc78488404)

[איור 27 Full adder and full subtractor Layout 14](#_Toc78488405)

[איור 28 סימולציה של ה-Full Adder עם שורת הקוד 15](#_Toc78488406)

[איור 29 הסימולציה של המספר הראשון ב- Full adder and full subtractor עם שורת הקוד 15](#_Toc78488407)

[איור 30 הסימולציה של המספר הראשון והשני ב- Full adder and full subtractor 16](#_Toc78488408)

[איור 31 הסימולציה של תוצאות החיבור והחיסור ב- Full adder and full subtractor 16](#_Toc78488409)

[איור 32 סימולציה של המספרים והתוצאות ב- Full adder and full subtractor 17](#_Toc78488410)

[איור 33 סימולציה של המספרים והתוצאות ב- Full adder and full subtractor בתצורה מוגדלת 17](#_Toc78488411)

[איור 34 Decoder 4 כניסות ו- 16 יציאות 18](#_Toc78488412)

[איור 35 decoder 4 כניסות ו- 16 יציאות 19](#_Toc78488413)

[איור 36 הסימולציה של כניסות ה- decoder עם שורות הקוד 19](#_Toc78488414)

[איור 37 תוצאות הסימולציה של מוצא ה- Decoder 20](#_Toc78488415)

[איור 38 רכיב שורת הזיכרון הזמינה המיועדת לכתיבת המספרים בו זמנית 21](#_Toc78488416)

[איור 39 סימון רכיב יחידת הזיכרון הזמינה 22](#_Toc78488417)

[איור 40 סכמת המעגל הכולל של המחשבון 22](#_Toc78488418)

[איור 41 icon של מערכת המחשבון 23](#_Toc78488419)

[איור 42 רכיב שורת הזיכרון הזמינה 24](#_Toc78488420)

[איור 43 המעגל הכללי של מערכת המחשבון 25](#_Toc78488421)

[איור 44 פעולות חיבור וחיסור במערכת המחשבון הכוללות אכסון במטריצת הזיכרון 26](#_Toc78488422)

[איור 45 קוד התוכנית של המחשבון 26](#_Toc78488423)