פרויקט סיכום במעבדת

VLSI דיגיטלי

מגיש: דניאל רם

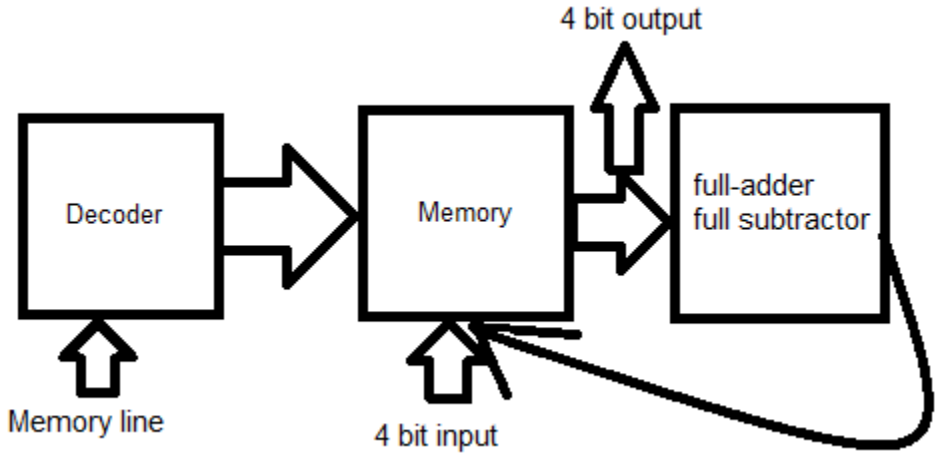
ת.ז: 208958322

תאריך הגשה: 20.08.2024

תוכן עניינים

חלק א: תכנון מחשבון בעזרת זיכרון מסוג SRAM

אנו רוצים לחבר/לחסר שני מספרים בעלי 4 ביט כל אחד באמצעות (full-adder) המשמש גם כ full subtractor כאשר המספרים (גם התוצאות של החיבור/חיסור) צריכים לעבור דרך הזיכרון.
יש לבנות מטריצת זיכרון 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.
צריכים לדאוג לכך שלמטריצה ניתן יהיה לכתוב בשני אופנים: דרך המשתמש (שורות קוד) או דרך לוגיקה (מוצא ה full adder).
סכמת בלוקים של המערכת:



איור 1 סכמת בלוקים

לסכמה ראשונית זו יש להוסיף את הרכיב שבוחר באיזה אופן יש לכתוב לזיכרון, דרך המשתמש או מוצא המחשבון. בנוסף יש להוסיף רכיב שיאפשר הכנסת 2 מספרים במקביל ל- Full adder וה- Full subtractor , יש גם להוסיף רכיב שישמור את יציאת ה- Full adder וה- Full subtractor זאת על מנת לאכסנו בזיכרון (מכיוון ששינוי של שורת הזיכרון יפגע בקלט הרכיב וישנה את מוצאו), לכן נשתמש ברכיב D-FF שישמור את הכניסה והמוצא של המחשבון.

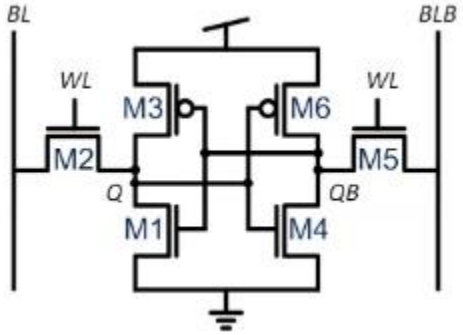
1. תא SRAM בודד

Static random Access Memory היינו תא זיכרון שמורכב מ-6 טרנזיסטורים אשר למעבד יש יכולת לגשת אליו ישירות ולכתוב אליו או לקרוא ממנו.

זיכרון זה נקרא סטטי מכיוון שאין צורך לרענן אותו על מנת לשמור את ערכו, אך אם אספקת החשמל אליו תופסק אזי יאבד את ערכו,

SRAM צורך הספק כל הזמן, אך נחשב כזיכרון מהיר.

SRAM היינו רכיב זיכרון יחסית גדול.



איור 2 סכמה של תא SRAM

ל- SRAM יש שלושה מצבים אפשריים:

- מוכן לפעולה (Standby): במצב זה התא שומר על התוכן האחרון שנכתב אליו.
- קריאה: המידע נקרא מהתא.
- כתיבה: מידע חדש (שיכול להיות זהה או שונה מהתוכן הנוכחי) נכתב לתא.

<u>השהייה</u>

כאשר כניסת Word Line (WL) היא ב-0 אז השערים M5 ו- M2 מנתקים את התא מקווי הביטים (BL ו-BLB) שני המהפכים המורכבים כל אחד משני טרנזיסטורים המחוברים בצורה הופכית במרכז M1-3 ו-M3-4, טכנולוגיה הנקראת CMOS, ימשיכו להפעיל אחד את השני וישמרו על המצב הסטטי כל עוד הם מנותקים מקווי הביטים.

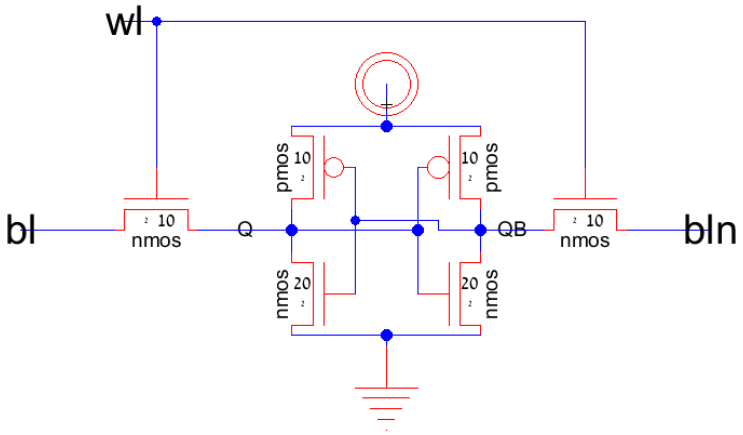
<u>קריאה</u>

אם התוכן של הזיכרון הוא 1, ערכו שווה לרמה הלוגית בנקודה Q. התהליך מתחיל בכך שקווי הביט מקבלים ערך 1. לאחר מכן קו WL מקבל גם ערך 1. הטרנזיסטורים 5 ו-2 נפתחים. קו BL נשאר כמו שהיה מכיוון ש- Q הוא 1. לעומת זאת קו BL שהוא טעון ב-1 נפרק כולו דרך טרנזיסטור M1 לאדמה. מה שמשאיר את BL ב-0. מכיוון שהמעגל סימטרי מה שקורה אם המעגל היה טעון ב-0 הוא בדיוק אותו דבר רק ש-BL מתחלף עם BLB.

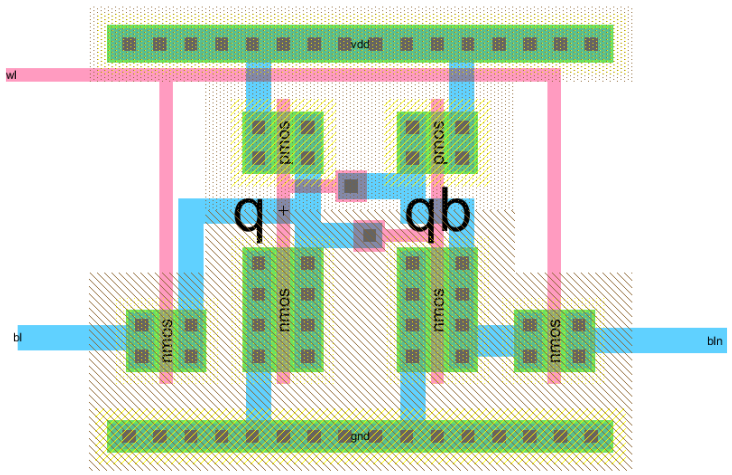
<u>כתיבה</u>

תחילת תהליך הכתיבה מתחיל על ידי הכנסת המידע ל-BL ו- BLB כלומר אם רוצים לכתוב 0 הופכים את BL ל-0 ואת BLB ל-1 (ואם רוצים לכתוב 1 אז להפך). לאחר מכן מפעילים את WL ואז המידע נכנס פנימה לתוך שני המהפכים. הסיבה לכך שהמידע הקודם נדרס הוא כיוון שהדוחפים של קווי הביטים הם הרבה יותר חזקים מהמהפכים החלשים שבתוך התא ולכן אם היה במוצא 1 הוא נפרק ומקבל את הערך 0.

לזיכרון מסוג SRAM יש צורך במעגל שנקרא PRECHARG זאת על מנת לאפשר טעינה של הקווים לצורך קריאה מן התא זיכרון או העברה של ערך חיצוני לקווי הטעינה של התא.
בנוסף נוסף מגבר משווה ביציאה מתא זיכרון על מנת שלא נצטרך להמתין עד להתפרקות הקווים BL או BLB, אלא ישר נשווה ביניהם ונראה מי גבוה יותר ובכך לקבל קריאה עוד יותר מהירה (ומדויקת יותר) מן הזיכרון.

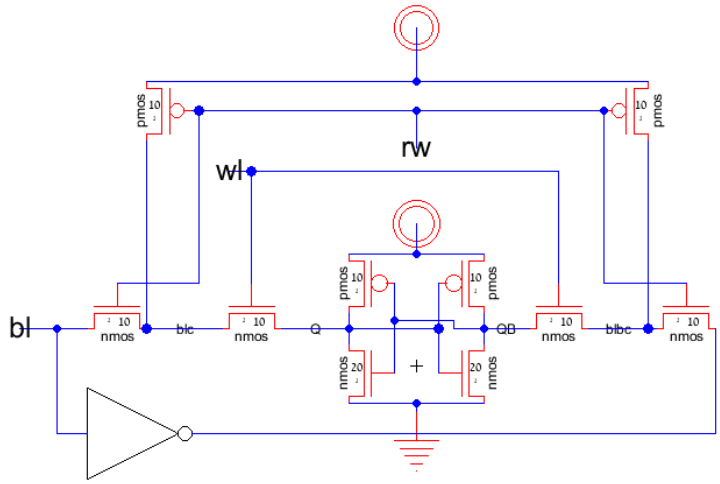


איור 3 סכמה של תא SRAM שנבנה בפרויקט



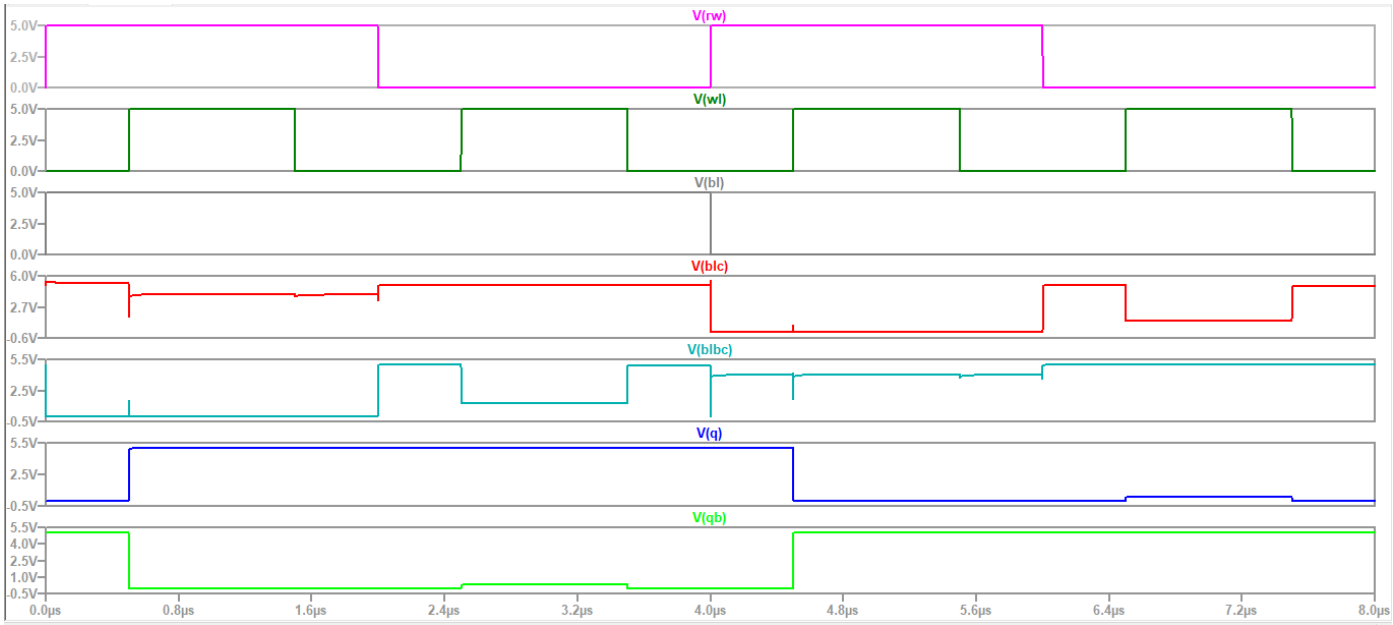
איור 4 LAYOUT של תא SRAM בודד שנבנה בפרויקט

בשני האיורים ניתן לראות סכמה של תא SRAM בודד, ללא PRECHARGE וללא המגבר משווה.
בכדי לבדוק את פעולת התא יש צורך לחבר מעגל טעינה (PRECHARGE).



איור 5 סכמה של תא SRAM עם מעגל טעינה

על מעגל זה הורצה סימולציה לבדיקת תקינות המעגל.



איור 6 סימולציה של תא SRAM בודד עם מעגל PRECHARGE

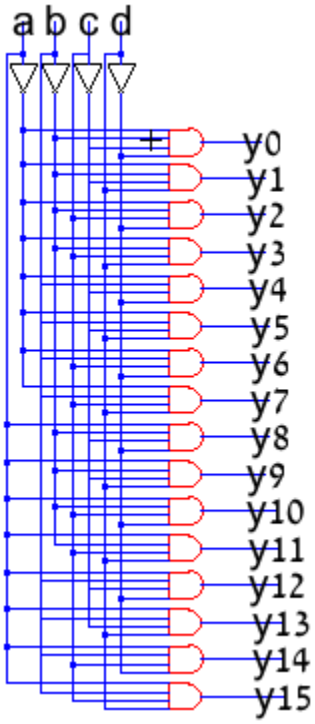
<u>פירוט הזמנים:</u>

תחילה אנו מבצעים כתיבה (rw נמצא ב-1) אך אנו לא מבצעים כלום עד אשר WL עולה ל-1, בזמן זה BL (0.5µs) נמצא ב-1, כלומר אנו כותבים ל-BLC 1 (וגם ל-Q) ול-BLBC 0 (וגם ל-QB), ואז מורידים את WL ל-0 והמצב נשמר עד אשר WL שוב עולה ל-1 (2.5µs) שבו RW נמצא במצב קריאה (0), בזמן זה ניתן לראות BLC גבוה מ-BLBC, כלומר אנו קוראים 1 לוגי.

לאחר מכן אנו מעלים את WL ל-1 שוב (4.5μs), בזמן זה אנו כותבים (rw ב-1) ל-BLC 0 (וגם ל-Q) ול-BLBC 1 (וגם ל-QB), ואז מורידים את WL ל-0 והמצב נשמר עד אשר WL שוב עולה ל-1 (6.5μs) שבו rw נמצא במצב קריאה (0), בזמן זה ניתן לראות BLBC גבוה מ-BLC, כלומר אנו קוראים 0 לוגי.

2. Decoder

על מנת לבחור שורת זיכרון ישנו צורך ברכיב המפענח, לו יהיו 4 כניסות על מנת לבחור אחת מ-16 שורות זיכרונות במטריצת הזיכרון.

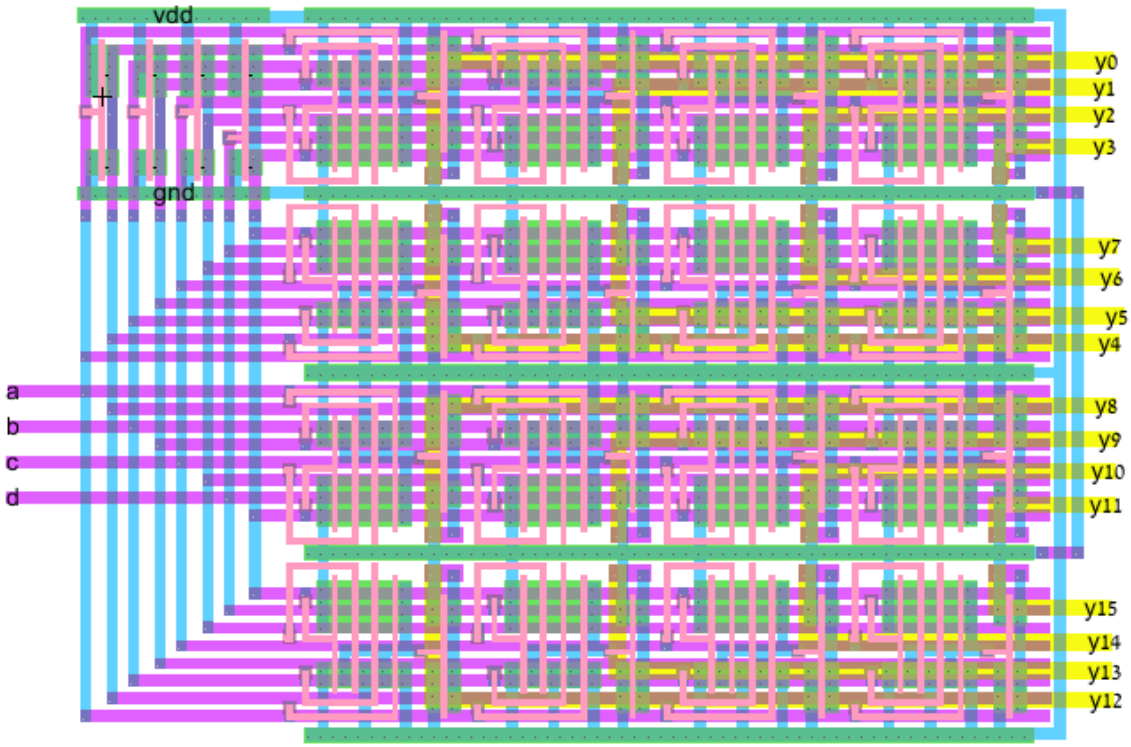


איור 7 סכמה של חלק מהמפענח שנבנה בפרויקט

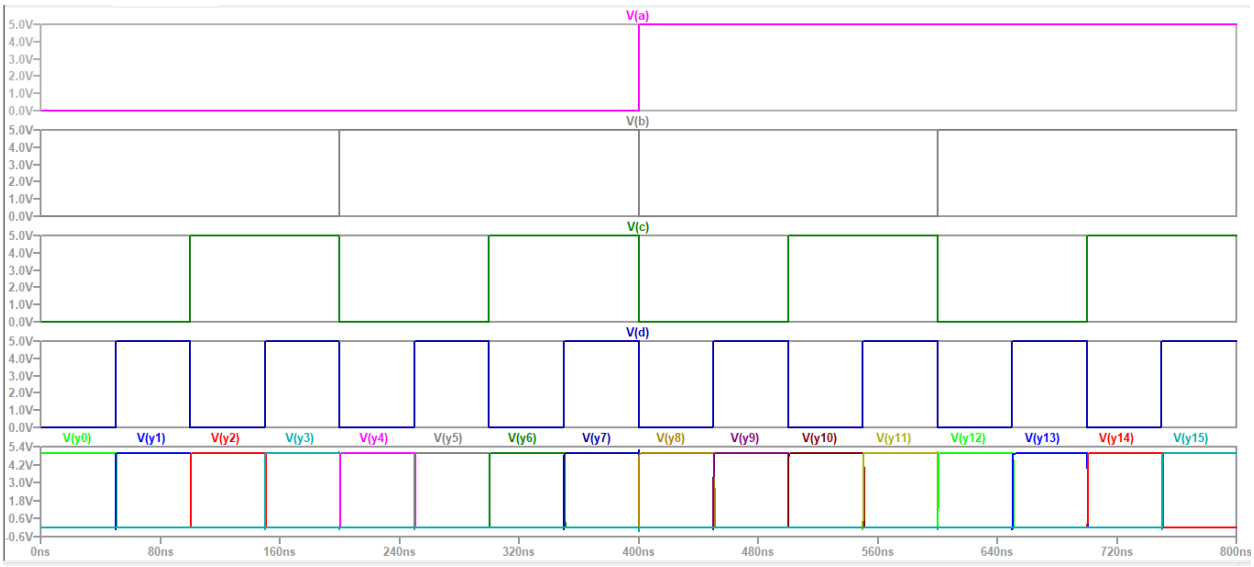
כאשר הכניסות הם כולם אפסים ("0000") רק היציאה Y0 תהיה 1, וכל האחרות יהיו 0.
כאשר הכניסות הם כולם אפסים חוץ מ-D ("0001") רק היציאה Y1 תהיה 1, וכל האחרות יהיו 0.
כאשר הכניסות הם כולם אפסים חוץ מ-C ("0010") רק היציאה Y2 תהיה 1, וכל האחרות יהיו 0.
כאשר הכניסות A ו-B אפסים ו-C ו-D הם אחדות ("0011") רק היציאה Y3 תהיה 1, וכל האחרות יהיו 0.
וכן הלאה.
כך ניתן לבחור שורה ספציפית של מטריצת הזיכרון בה ה-WL יעלה ל-1 ובכל השאר WL יהיה 0, וכך ניתן לבחור שורה אחת של 4 תאי SRAM לכתוב אליה או לקרוא ממנה.

A	B	C	D	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13	Y14	Y15
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

איור 8 טבלת אמת של מפענח 4 ל-16



איור 9 LAYOUT של המפענח 4 ל-16 שנבנה בפרויקט

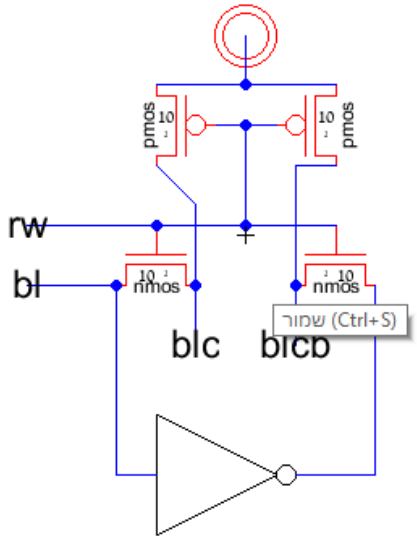


איור 10 סימולציה של המפענח 4 ל-16 שנבנה בפרויקט

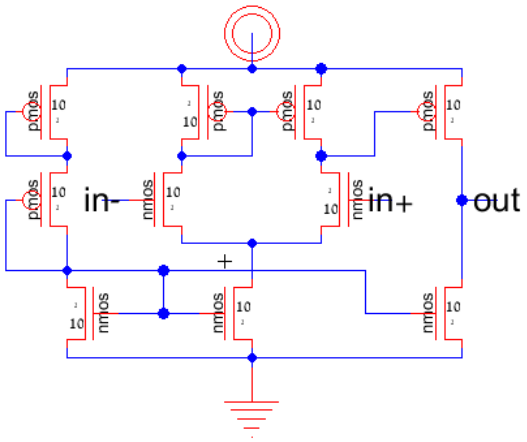
ניתן לראות כי טבלת אמת של המפענח באיור 8 מתקיימת.

3. מטריצת זיכרון

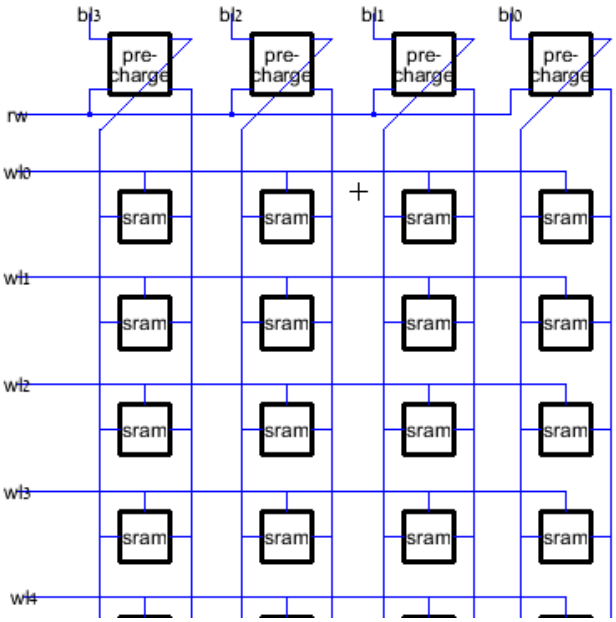
מטריצת הזיכרון בנויה מ-16 שורות שלכל שורה ארבעה תאי SRAM, כלומר 64 תאי SRAM מכיוון שבכל פעם נבחר שורת זיכרון אחת, נוכל לבנות לכל טור בזיכרון מעגל PRECHARGE אחד ומגבר משווה אחד, כלומר 4 מעגלי PRECHARGE ו-4 מגברים משווים.



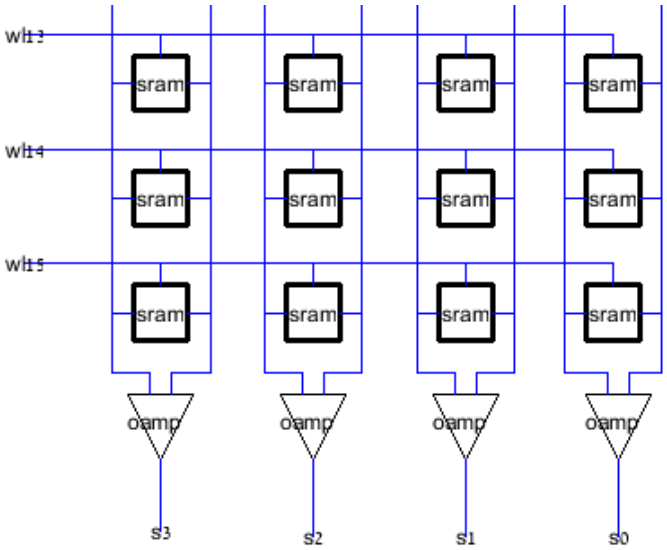
איור 11 סכמה של מעגל PRECHARGE שנבנה בפרויקט



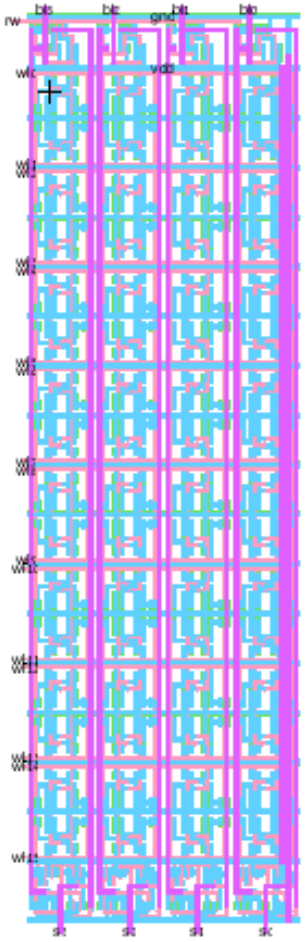
איור 12 סכמה של מגבר משווה שנבנה בפרויקט



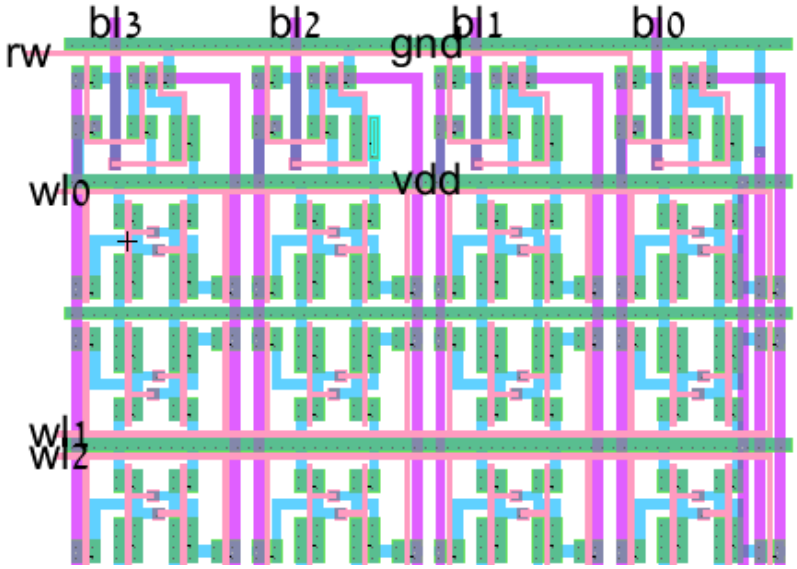
איור 13 סכמה של חלק ממטריצת הזיכרון שנבנה בפרויקט



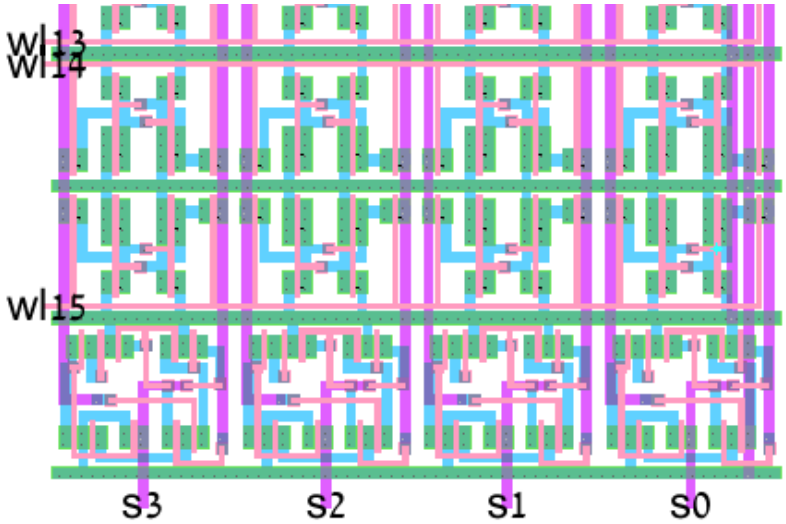
איור 14 סכמה של עוד חלק ממטריצת הזיכרון שנבנה בפרויקט



איור 15 LAYOUT של מטריצת הזיכרון שנבנתה בפרויקט

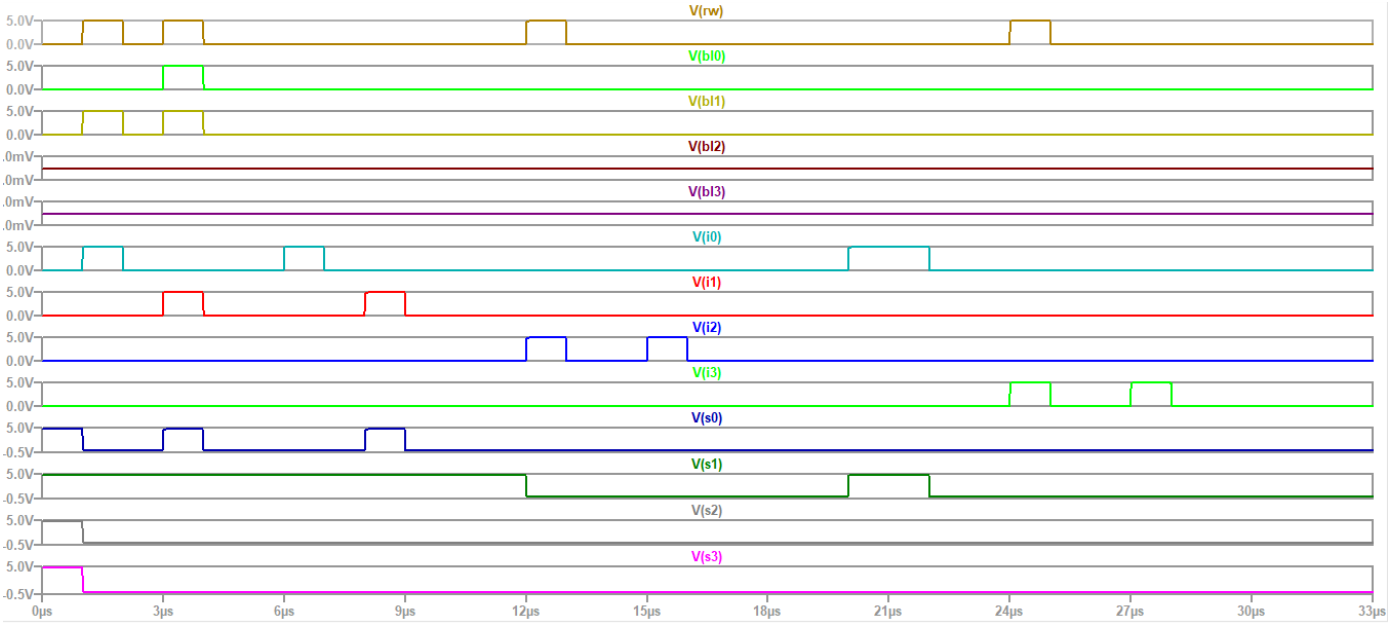


איור 16 תקריב של החלק העליון ב-LAYOUT של מטריצת הזיכרון הכולל את מעגלי ה-PRECHARGE ותאי SRAM



איור 17 תקריב של החלק התחתון ב-LAYOUT של מטריצת הזיכרון הכולל את המגברים משווים ותאי SRAM

על מנת להריץ סימולציה של מטריצת הזיכרון חובר לכניסות WL את יציאות המפענח 4 ל-16 שנבנה קודם על מנת להקל בכתיבת קוד ה- SPICE.



איור 18 סימולציה של מטריצת הזיכרון אשר מחוברת למפענח

תיאור זמנים:

- 0-1µs קריאה של כתובת "0000" (כתובת "זבל" אשר אותה תמיד נפעיל כשלא נרצה לבצע קריאה או כתיבה מהזיכרון, מכיוון שלא הוספנו רגל הפעלה למפענח), מתקבל מידע לא רלוונטי במוצא "1111".
- 1-2µs כתיבה לכתובת "0001" של הערכים "0010".
- 3-4µs כתיבה לכתובת "0010" של הערכים "0011".
- 6-7µs קריאה של כתובת "0001" מתקבלים הערכים "0010", כמצופה.
- 8-9µs קריאה של כתובת "0101" מתקבלים הערכים "0011", כמצופה.
- 12-13µs כתיבה לכתובת "0100" של הערכים "0000".
- 15-16µs קריאה של כתובת "0100" מתקבלים הערכים "0000", כמצופה.
- 20-22µs קריאה של כתובת "0001" מתקבלים הערכים "0010", כמצופה.
- 24-25µs כתיבה לכתובת "1000" של הערכים "0000".
- 27-28µs קריאה של כתובת "1000" מתקבלים הערכים "0000", כמצופה.

4. Full adder and Full subtractor

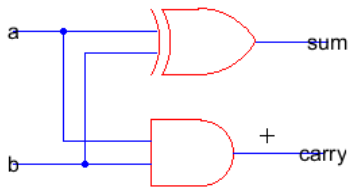
רכיב זה יודע לבצע פעולת חיבור או חיסור בין 2 מספרים באורכים של 4 ביטים.

חיבור היא פעולה פשוטה בבינארית: 0+0=0 , 0+1=1 , 1+1=10 כאשר ישנו CARRY לחיבור של הביט האחרון.

חיסור בינארי באמצעות שיטת המשלים ל-2 נעשה על ידי המרת פעולת החיסור לפעולת חיבור. ראשית, מייצגים את המספרים בבינארי. לדוגמה, עבור 2-5 (0010 ו-0101 בבינארי בהתאמה), מחשבים את המשלים ל-2 של המספר השני על ידי הפיכת כל הביטים (משלים ל-1) והוספת 1. המשלים פלוס 1 של 5 הוא 1011. כעת, מבצעים חיבור של המספר הראשון עם המשלים של המספר השני:

0010 + 1011 = 1101. מאחר והביט המוביל הוא 1, התוצאה היא שלילית. כדי למצוא את הערך המוחלט, הופכים שוב את כל הביטים ומוסיפים 1, מתקבל 0011, כלומר 3. לכן, 2-5 = 3-.

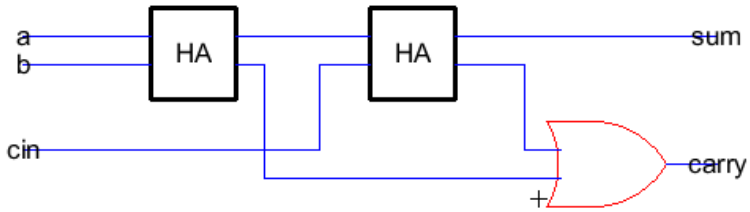
הרכיב הבסיסי של חיבור נקרא HALF-ADDER שהוא מורכב משער XOR ושער AND כך שנוכל לחבר 2 ביטים:



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

איור 19 סכמה וטבלת אמת של רכיב ה-HALF-ADDER שנבנה בפרויקט

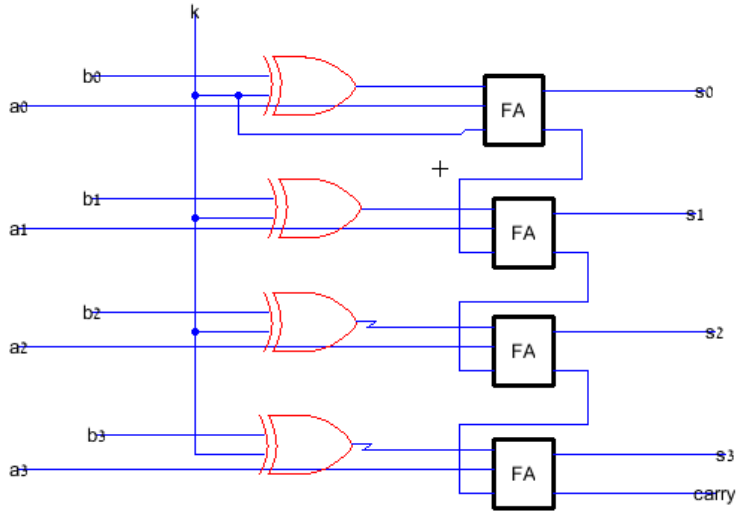
בעזרת רכיב זה ושער OR ניתן לבנות את הרכיב FULL-ADDER שבעזרתו ניתן לחבר 2 ביטים יחד עם CARRY מהדרגה הקודמת.



C_{in}	A	B	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

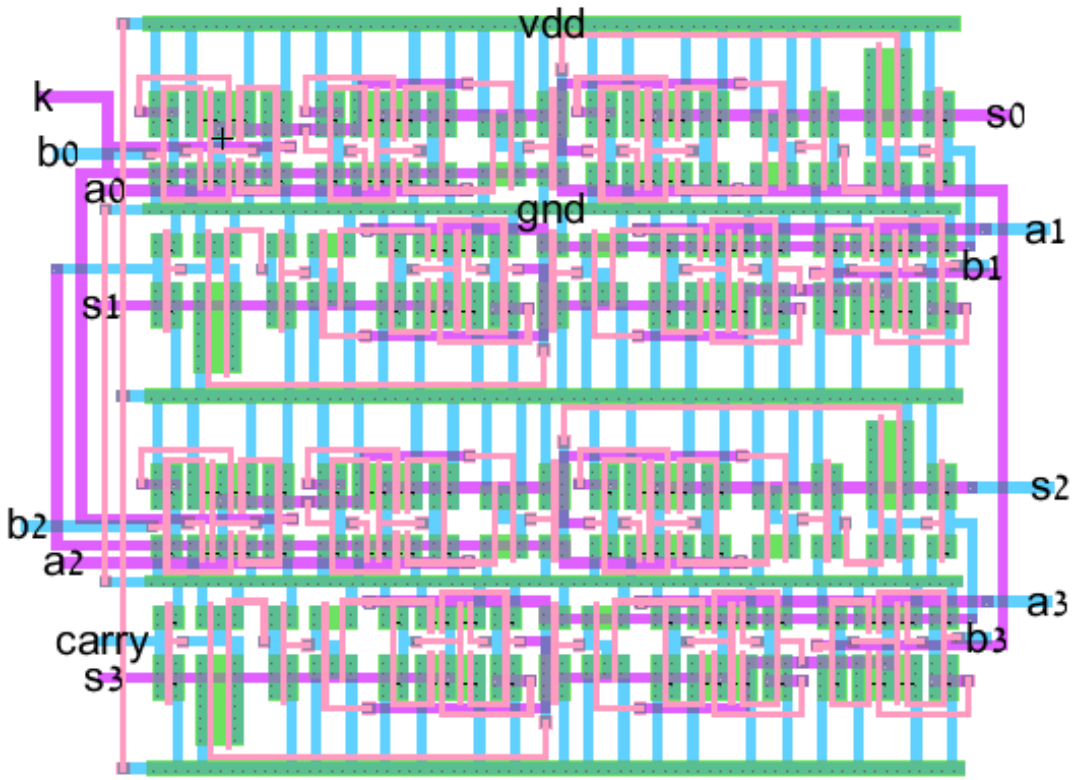
איור 20 סכמה וטבלת אמת של רכיב ה-FULL-ADDER שנבנה בפרויקט

בעזרת ארבעה FULL-ADDER ובעזרת 4 שערי XOR ניתן לבנות את רכיב שיודע לחבר או לחסר 2 מספרים בעלי 4 ביטים:

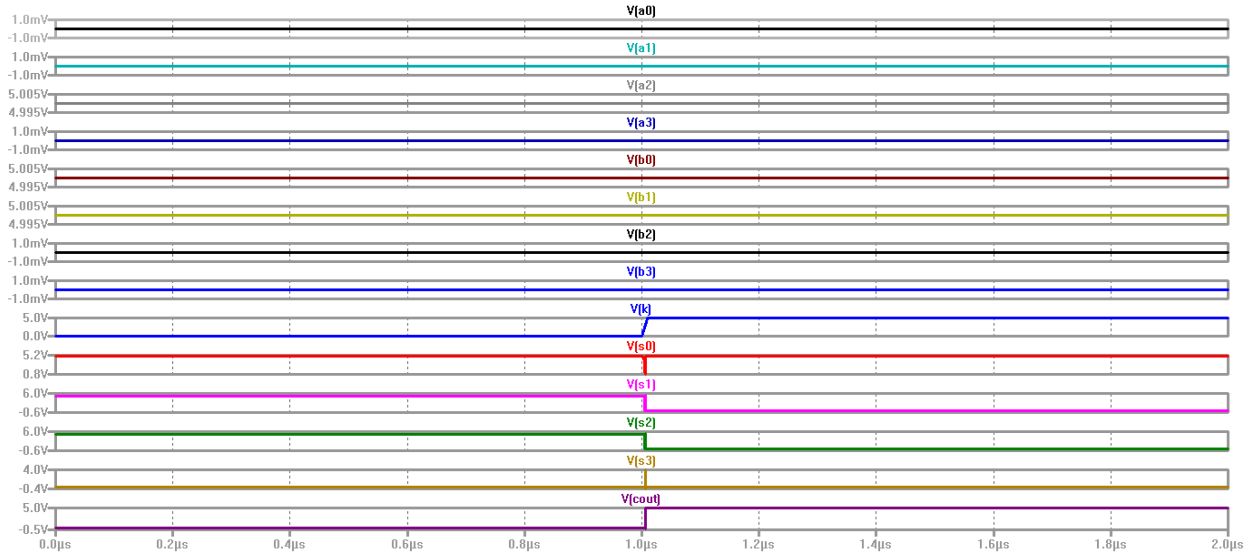


איור 21 סכמה של הרכיב PARALLEL ADDER/SUBTRACTOR אשר נבנה בפרויקט

בעזרת רכיב הכניסה K ניתן לבחור האם להפוך את הכניסות B ולהוסיף 1 דרך ה-CARRY IN של הדרגה הראשונה ובכך לבצע חיסור. כאשר K=1 נקבל A-B, לעומת זאת כאשר K=0 אז נקבל A+B במוצא.



איור 22 LAYOUT של הרכיב PARALLEL ADDER/SUBTRACTOR אשר נבנה בפרויקט



איור 23 סימולציה של הרכיב PARALLEL ADDER/SUBTRACTOR שנבנה בפרויקט

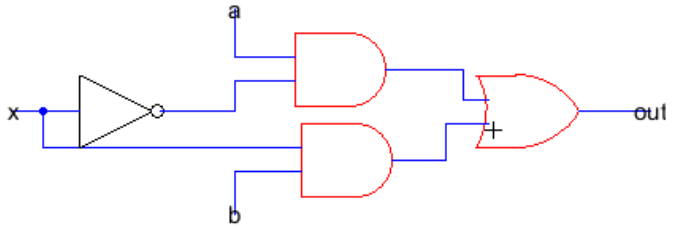
בחצי הסימולציה הראשון אנו מבצעים חיבור של המספרים 3 ו-4 ומתקבל 7,
בחצי השני של הסימולציה אנו מבצעים חיסור (K עולה ל-1) של המספר 4 ב-3 ומתקבל 1 (וה-COUT עולה ל-1) בזמן זה.

5. רכיבים נוספים

בכדי לחבר את הרכיבים שנבנו למעגל אחד שידע לקלוט מספרים למטריצת זיכרון ואחר כך להכניס במקביל לרכיב המחשב וממנו התוצאה של פעולת החישוב תיכנס חזרה למטריצת הזיכרון יש להוסיף כמה רכיבים:

5.1 - MUX-2X1

רכיב ה-MUX נועד לתת מענה לאופציה שישנן 2 כניסות (של 4 ביטים כל אחת) שצריכות להגיע למטריצת הזיכרון (כניסת משתמש ויציאת הרכיב המחשב).
לרכיב כניסה אשר בוחרת איזו אחת משתי הכניסות האחרות שלו תעבור למוצא שלו.
הרכיב מורכב משער מהפך שני שערי AND ושער OR אחד.

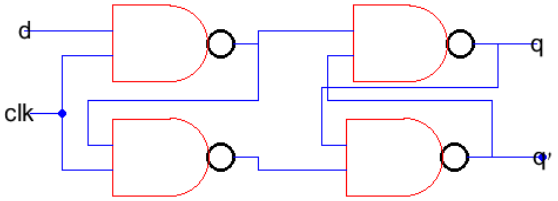


איור 24 סכמה של הרכיב MUX 2X1 אשר נבנה בפרויקט

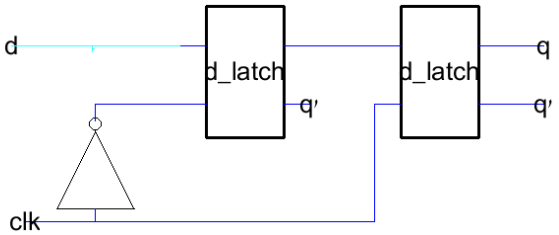
נצטרך 4 רכיבים כאלו (אחד לכל ביט).

D-Flip-flop - 5.2

רכיב זה נועד לאפשר שמירה של ערך מסוים, כך נוכל לשמור ערך שיצא ממטריצה הזיכרון ואז נוכל להכניס במקביל שני ערכים למחשבון.
בנוסף גם במוצא הרכיב המחשב נצטרך רכיב זה, זאת בגלל שנרצה לשנות את הכתובת שניגש אליה במטריצה הזיכרון בכדי להתכונן לשמירת ערך חדש, אם לא נעשה זה ישנה את הערך שנכנס למחשבון וכתוצאה מכך הערך במוצאו ישתנה.
רכיב זה מורכב מ- 2 רכיבי D-Latch ומהפך, כאשר רכיב D-Latch מורכב מארבעה NAND.



איור 25 סכמה של הרכיב D-Latch אשר נבנה בפרויקט

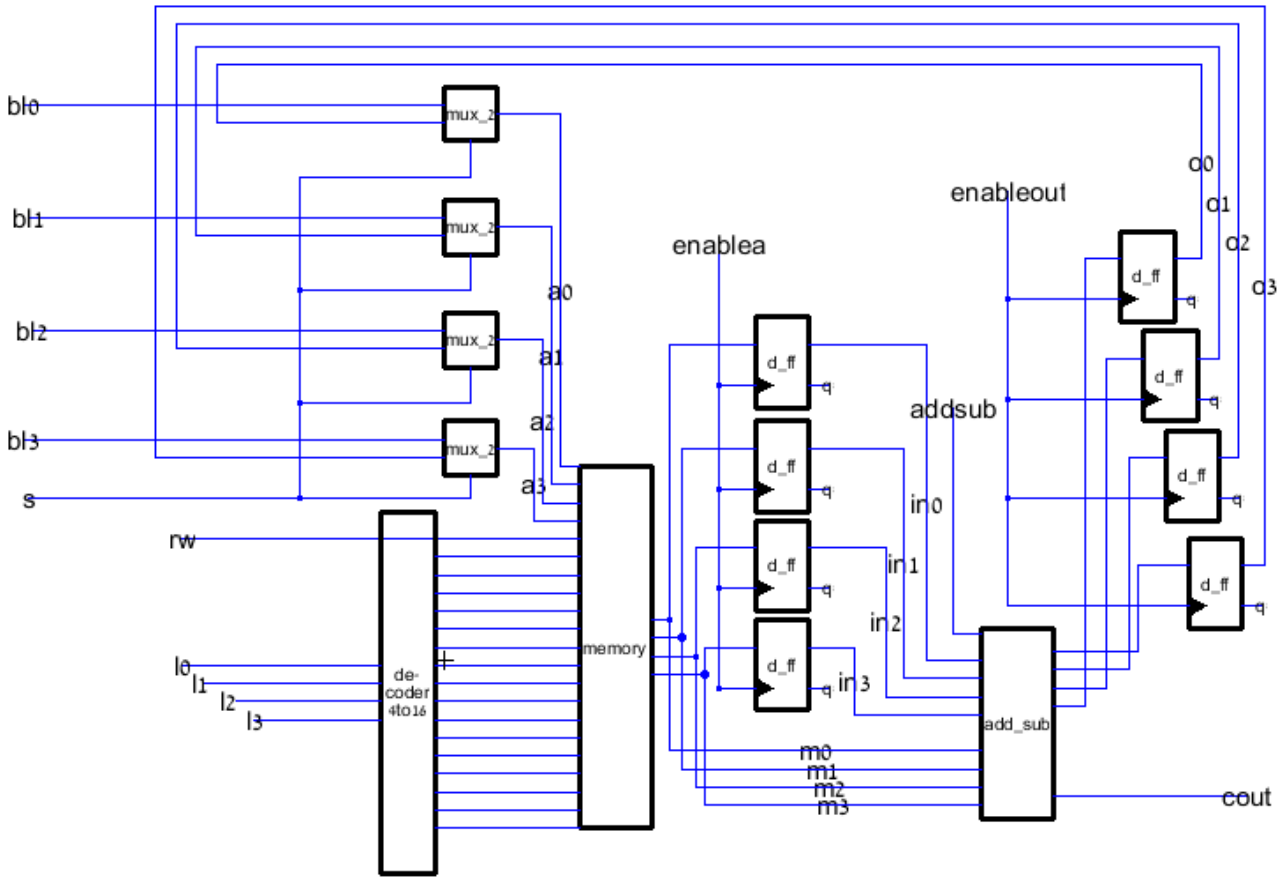


איור 26 סכמה של הרכיב D-flip-flop אשר נבנה בפרויקט

נצטרך 8 רכיבים כאלו (אחד לכל ביט אחרי הרכיב המחשב, ואחד לכל ביט של המספר הראשון לפני הרכיב המחשב).

6. בניית מחשבון בעזרת זיכרון מסוג SRAM

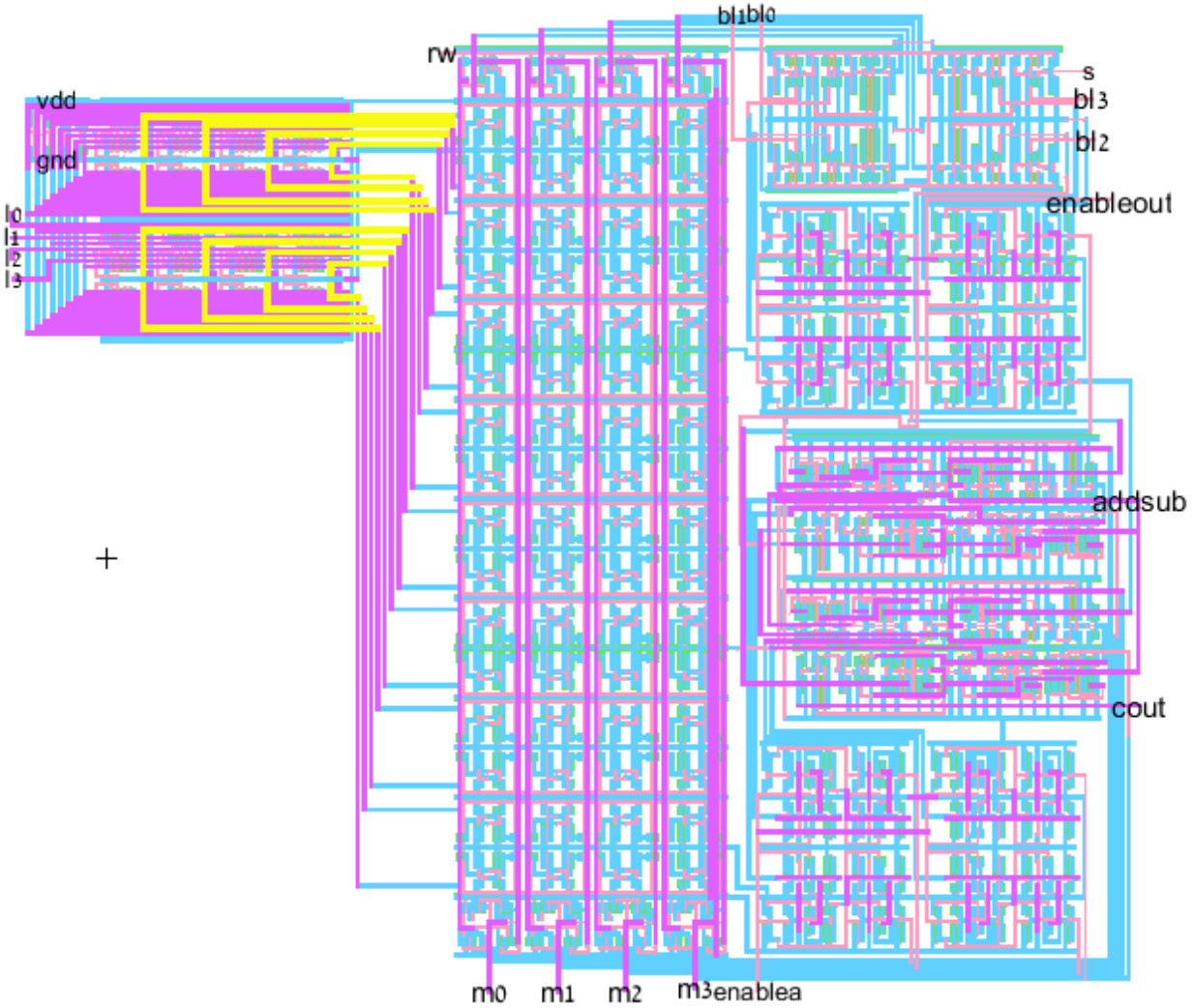
לבסוף כל הרכיבים חוברו להם יחדיו על מנת לממש את המחשבון:



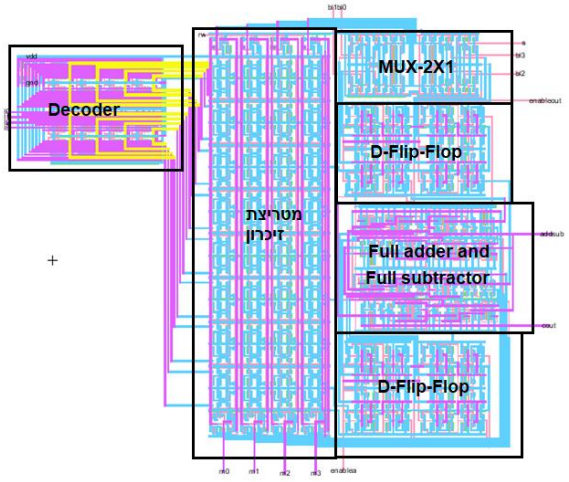
איור 27 סכמה של המעגל הסופי של הפרויקט

ניתן לראות כי תוצאות המחשבון ובחירת המשתמש כולם עוברים דרך הזיכרון, בנוסף ניתן לראות את יכולת הבחירה לתוך הזיכרון בין אם זה הכנסת ערך של המחשבון או הכנסת ערך מהמשתמש.

ניתן לראות את יכולת השמירה בכניסה ובמוצא של המחשבון כדי להכניס בזמן הנכון את המידע למחשבון ולהוציא בזמן הנכון לזיכרון את החישוב שנעשה.

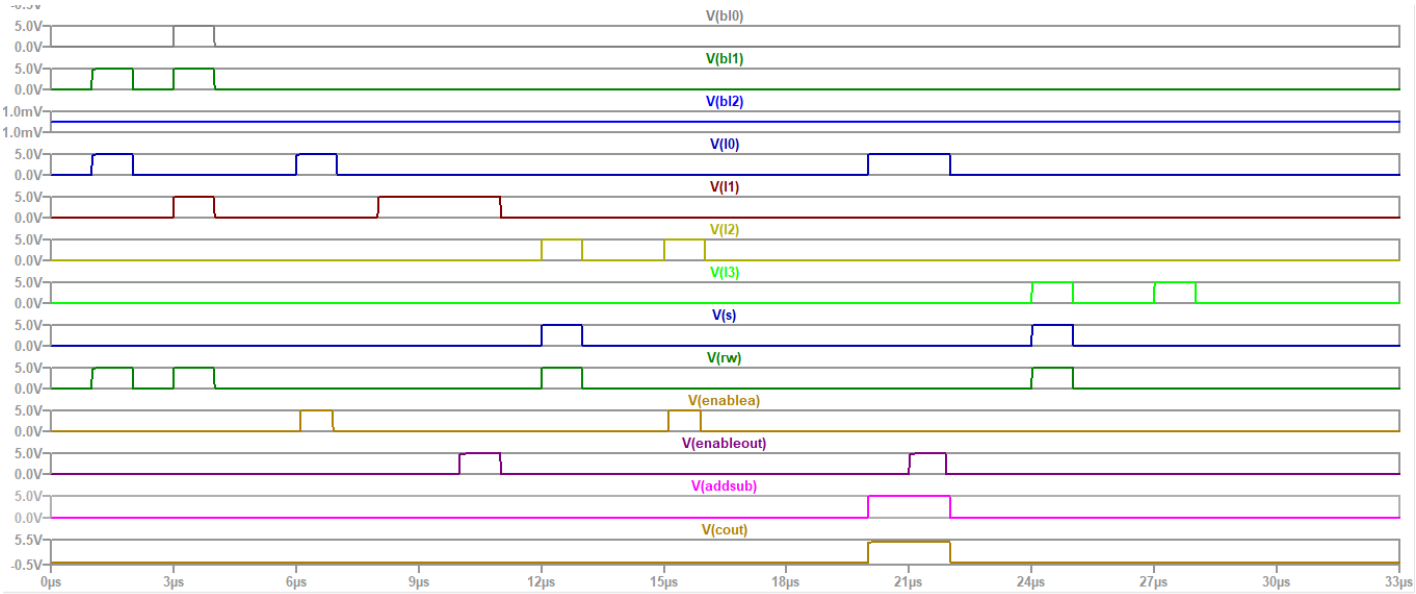


איור 28 LAYOUT של המעגל הסופי של הפרויקט

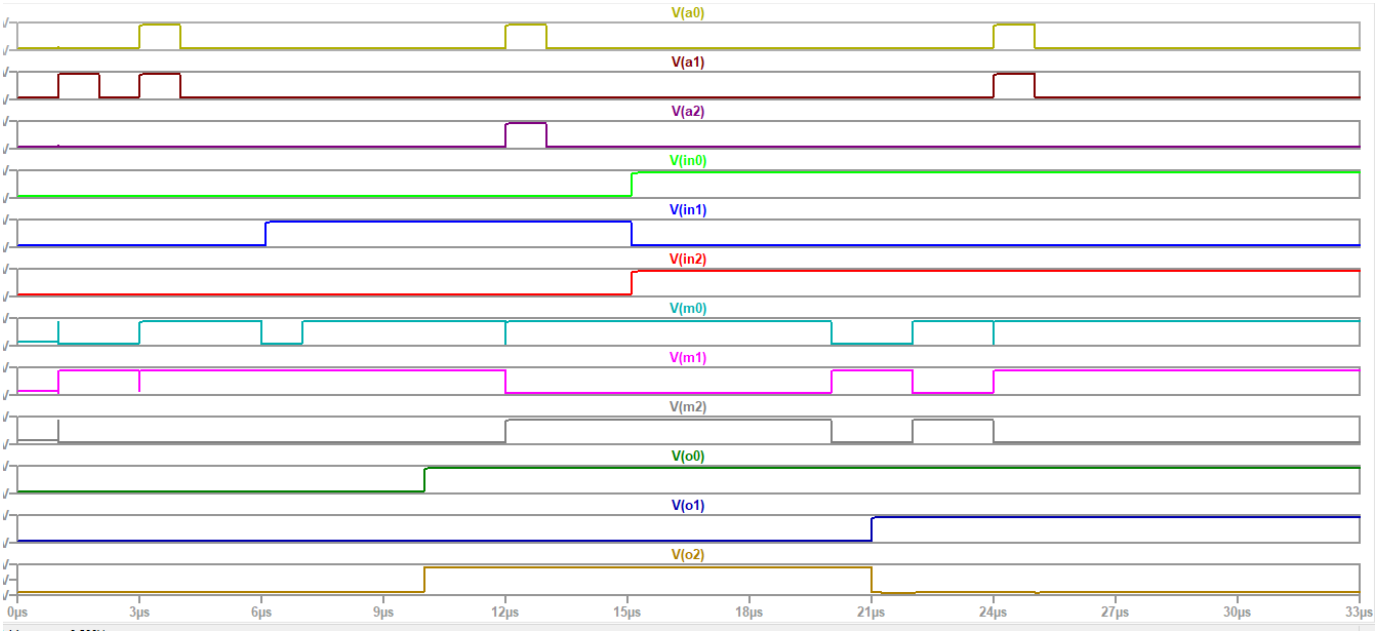


איור 29 חלוקה לרכיבים של ה-LAYOUT

סימולציה:



איור 30 סימולציה של המעגל הסופי של הפרויקט



איור 31 המשך אותות סימולציה של המעגל הסופי של הפרויקט

פירוט על נקודות הדגימה:

- Vbl – ערכים אותם נרצה להניס לרכיב הזיכרון (bl0-bl3 כאשר bl2 ו-bl3 תמיד 0 במקרה שלנו).
- Vl – בחירת שורת הזיכרון אותה נרצה להפעיל (Vl0-Vl3).
- Vs – רגל הכניסה לרכיבי ה-MUX אשר יבחרו לנו אילו נתונים להכניס למטריצת הזיכרון, כאשר 's='0 יוצא בחירת המשתמש (bl) וכאשר 's='1 יוצא המוצא של המחשבון.
- Vrw – כאשר rw הוא 0 אנו קוראים מן שורת הזיכרון, כאשר rw הוא 1 אנו כותבים אל שורת הזיכרון.
- Venablea – מאפשר טעינת הכניסה למוצא הזיכרון של ה- D-Flip-Flop (Vin) שבכניסה לרכיב המחשבון.
- Venableout – מאפשר טעינת הכניסה למוצא הזיכרון של ה- D-Flip-Flop (Vo) שביציאה של הרכיב המחשבון.
- Vaddsub – כאשר 0 אנו מבצעים חיבור של המספרים שבכניסה לרכיב המחשב, כאשר 1 אנו מבצעים חיסור של המספרים שבכניסה לרכיב המחשבון.
- Vm – מוצאי רכיב הזיכרון בעת קריאה.
- Vcout – מוצא ה-CARRY OUT של הרכיב המחשבון.
- Va – מוצא רכיבי הMUX כאשר מתאר לנו את כניסה לזיכרון מהמשתמש או ממוצא המחשבון.

הסבר זמנים:

- 0-1μs קריאה של כתובת "0000" (כתובת "זבל" אשר אותה תמיד נפעיל כשלא נרצה לבצע קריאה או כתיבה מהזיכרון, מכיוון שלא הוספנו רגל הפעלה למפענח), מתקבל מידע לא רלוונטי במוצא "0000".
- 1-2μs כתיבה לכתובת "0001" של הערכים "0010" $(2)_{10}$ מהמשתמש, נראה כי s=0 במקרה זה וכך מתקבל מידע מהמשתמש לזיכרון.
- 3-4μs כתיבה לכתובת "0010" של הערכים "0011" $(3)_{10}$ מהמשתמש ('s='0).
- 6-7μs קריאה של כתובת "0001" מתקבל בm הערכים "0010" $(2)_{10}$, כמצופה, ובמקביל טעינת
  D-Flip-Flop שבכניסה לרכיב הזיכרון וכך מתקבל בin הערכים "0010" בכדי לשמור את הכניסה בa.
- 8-11μs קריאה של כתובת "0010" מתקבלים הערכים "0011" $(3)_{10}$, כמצופה בm, לאחר מכן (10-11μs) אנו טוענים את ה-D-Flip-Flop במוצא הרכיב המחשב את תוצאת החיבור בין m ל-in ומוציא אותה לאות o כמו שניתן לראות מתקבל "0101" $(5)_{10}$.
- 12-13μs כתיבה לכתובת "0100" =i של הערכים שהתקבלו במוצא הרכיב המחשבון (מוצא רכיב
  D-Flip-Flop) ניתן לראות כי 's='1 וכן "a="0101 .

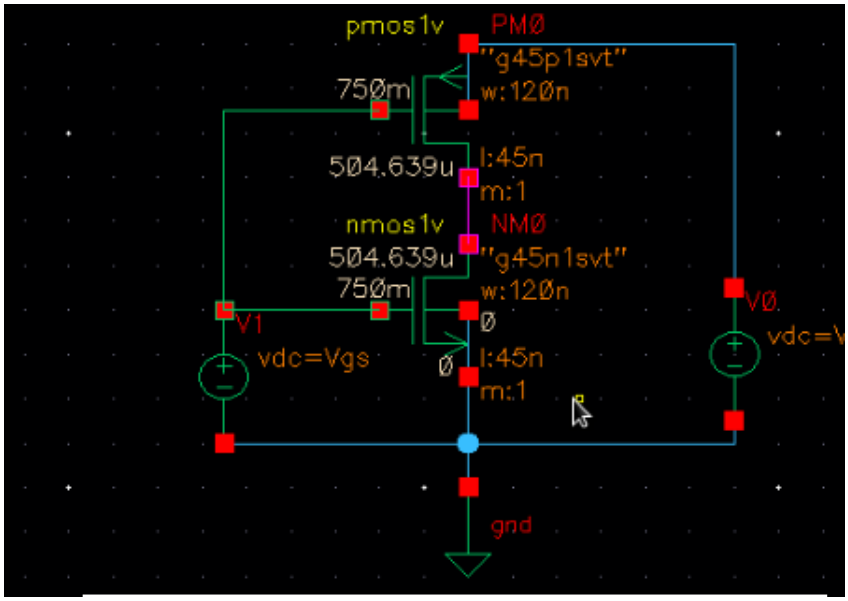
- 15-16μs קריאה של כתובת "0100" מתקבלים הערכים "0101" $(5)_{10}$, כמצופה (חיברנו 2 עם 3 וקיבלנו 5) בקו m, במקביל אנו טוענים את ה-D-Flip-Flop ('enablea='1) כך שin מקבל גם את הערכים "0101" $(5)_{10}$.
- 20-22μs קריאה של כתובת "0001" מתקבלים הערכים "0010" (2) בקו m, כמצופה, במקביל אנו מחסרים (addsub עולה ל-1) ולאחר מכן (21-22μs) אנו טוענים את ה-D-Flip-Flop במוצא הרכיב המחשבון ומתקבלת התוצאה בקו o. נוכל לראות שקו Cout עלה ל'1' בעת החיסור כמצופה.
- 24-25μs כתיבה לכתובת "1000" של הערכים שהתקבלו במוצא הרכיב המחשבון (מוצא רכיב D-Flip-Flop) לתוך הזיכרון. נראה כי "a="0011 כמצופה כמו קו o אחרי חישוב התוצאה.
- 27-28μs קריאה של כתובת "1000" מתקבלים הערכים "0011" (3) בקו m, כמצופה (חיסרנו 5 ב-2 וקיבלנו 3).

טבלה 1 - רכיב הזיכרון לאחר סיום הכתיבה וקריאה של המערכת

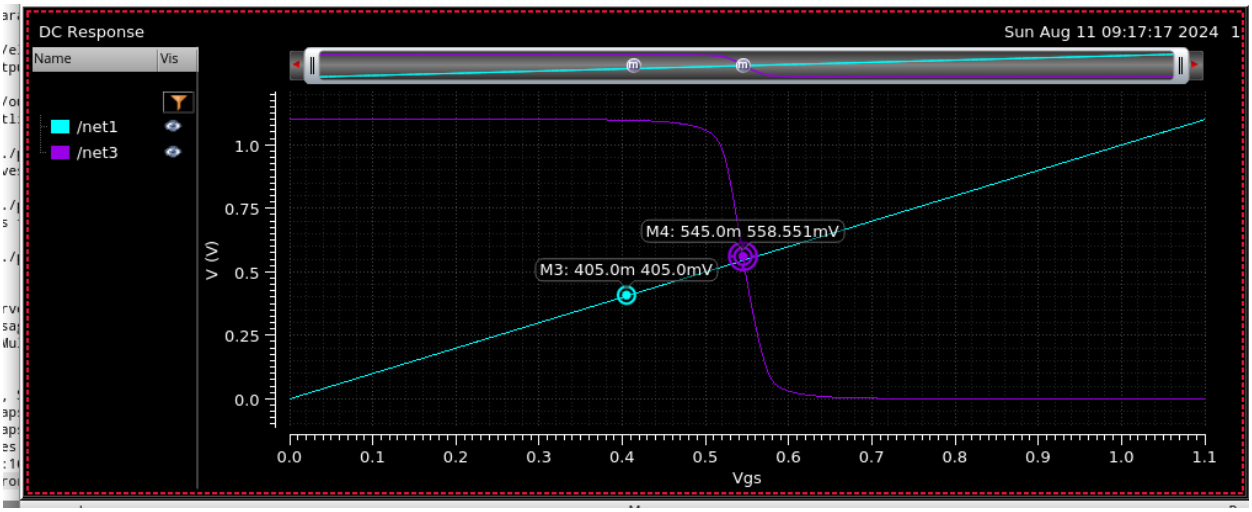
	3	2	1	0
0				
1	0	0	1	0
2	0	0	1	1
3				
4	0	1	0	1
5				
6				
7				
8	0	0	1	1
9				
. . 15				

חלק ב: Inverter ב- cadence

תחילה נוצר סכמה inverter בDC:



איור 32 סכמת inverter DC



איור 33 סימולצית inverter DC

ניתן לראות כי הסימולציה הזו בוחנת את תגובת המתח של המהפך במוצא (סגול) כתלות במתח הכניסה(תכלת) (Vgs).על פי הגרף ניתן להסיק את הערכים הבאים:

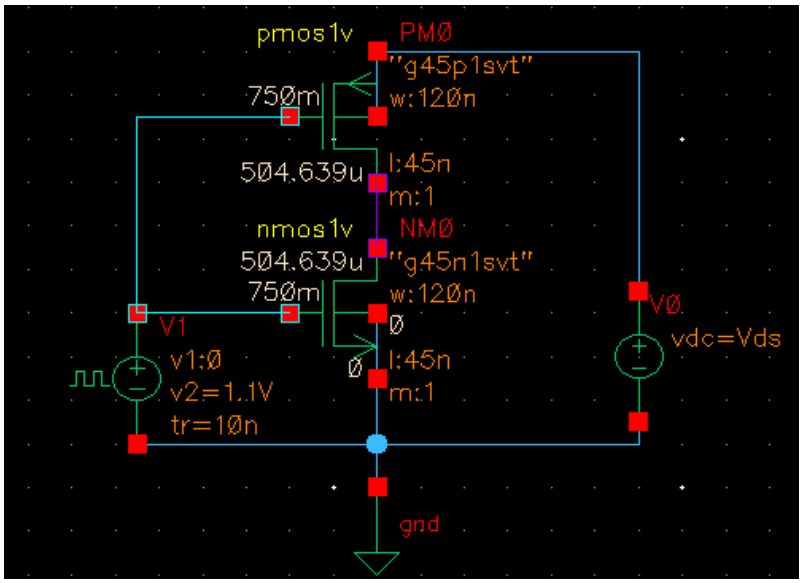
1. **Vhigh:** זה הוא המתח שמייצג את הלוגיקה "1" ביציאה של המהפך. לפי הגרף, Vhigh=1.1v כמצופה.
2. **Vlow:** זה הוא המתח שמייצג את הלוגיקה "0" ביציאה של המהפך. לפי הגרף, Vhigh=0v כמצופה.

3. **Vm**: זה הוא המתח שבו מתרחש המעבר בין "1" ל-"0" וזה הוא נקודת המפגש בין הכניסה למוצא (נקודה M4). לפי הגרף, Vm=0.55v. בדיוק באמצע מה שמעיד על פעולה תקינה ומעולה של מהפך סימטרי.

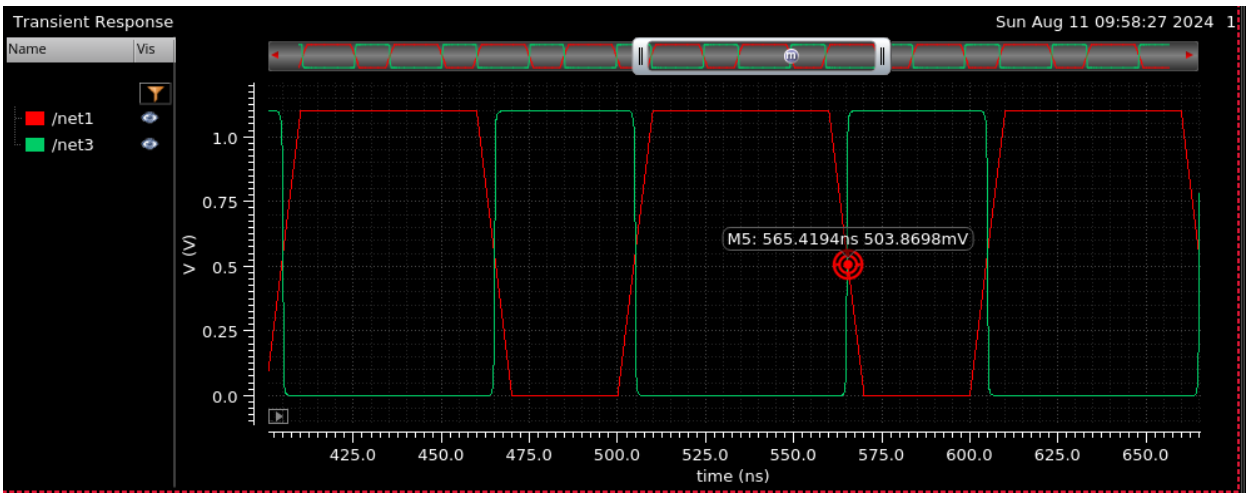
ניתן לראות שהגרף מראה מעבר חד בין המתח הנמוך לגבוה, מה שמעיד על פעולה תקינה של המהפך, שבו מתרחשת הפיכה של מצב הלוגיקה HIGH ל LOW-ולהפך.

כעת חובר בכניסה pulse ללא עומס (קבל) במקום כניסת DC עם הערכים הבאים:

$$period = 100ns, Delay = 0s, rise_{time} = 10ns, fall_{time} = 10ns, pulse\ widte = 50ns$$



איור 34 סכמת inverter tarn ללא עומס



איור 35 סימולציית inverter tarn ללא עומס

ניתן לראות כי המהפך הופך כנדרש באופן סימטרי כאשר הכניסה ב"0" לוגי (קו אדום) המוצא (קו ירוק) נמצא ב"1" לוגי, בנוסף ניתן לראות כי מכיוון שאין עומס נראה כי TPD מאוד קטן,

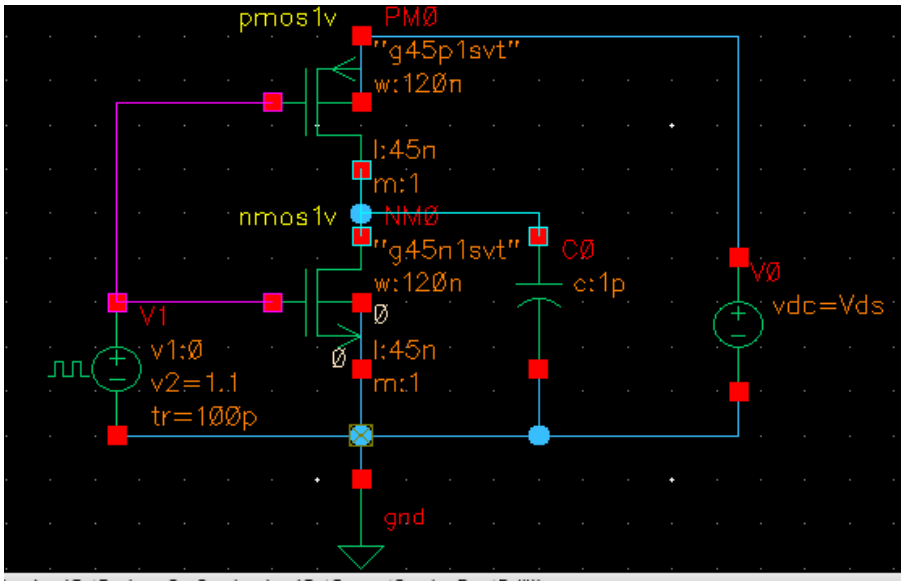
TPD הוא הזמן שעובר מרגע שהכניסה משתנה עד שהיציאה מגיעה למתח סף מסוים כאשר שבו היציאה משנה את מצבה מ-"0" ל-"1" או להיפך.

כאשר אין עומס במוצא, TPD קטן מכיוון שהשער הלוגי יכול לשנות את המצב של היציאה במהירות יחסית.

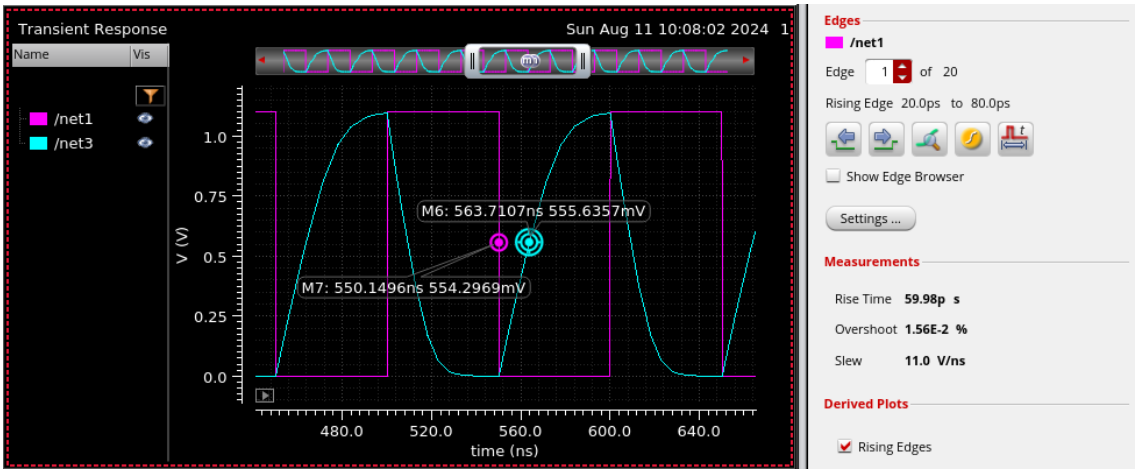
הוספת עומס במוצא מכבידה על המעגל, ולכן יכולה להגדיל את TPD.

נוסף עומס במוצא עם הערכים הבאים:

$$period = 100ns, Delay = 0s, rise_{time} = 0.1ns, fall_{time} = 0.1ns\ , pulse\ widte = 50ns$$



איור 36 סכמת inverter tarn עם עומס



איור 37 סימולצית inverter tarn עם עומס

ניתן לראות כי כאשר יש קבל במוצא, המתח ביציאה לא משתנה באופן מיידי כמו באיור 35 אלא עובר דרך תהליך של טעינה ופריקה של הקבל. זה מסביר את הצורה המעוגלת של העקומות במוצא(קו תכלת). הקבל במוצא גורם לכך שהמעבר בין המתח הגבוה לנמוך (ולהפך) ביציאה לא יהיה חד ומהיר כמו בכניסה (קו סגול) . המעבר הוא מתון יותר, הוא יוצר אפקט של השהיה, כלומר, ישנו עיכוב במעבר בין רמות המתח בכניסה לרמות המתח ביציאה. זמן זה נקרא TPD נראה כך שTPD הוגדל.

הנקודות M6 ו M7-מציינות את הזמן והמתח בנקודת מתח שווה על הגרף ניתן לראות שזמן התגובה(TPD) הוא הפרש בין הזמנים TPD=13ns.

תופעה זו יכולה להיות רצויה או לא רצויה, בהתאם ליישום. למשל, במעגלים מסוימים זה עשוי לשפר את יציבות המערכת על ידי סינון רעשים, אבל במעגלים אחרים זה עשוי לגרום לעיכוב לא רצוי בתגובה.

בנוסף ניתן לראות חישובים נוספים שעולים מהגרף והם:

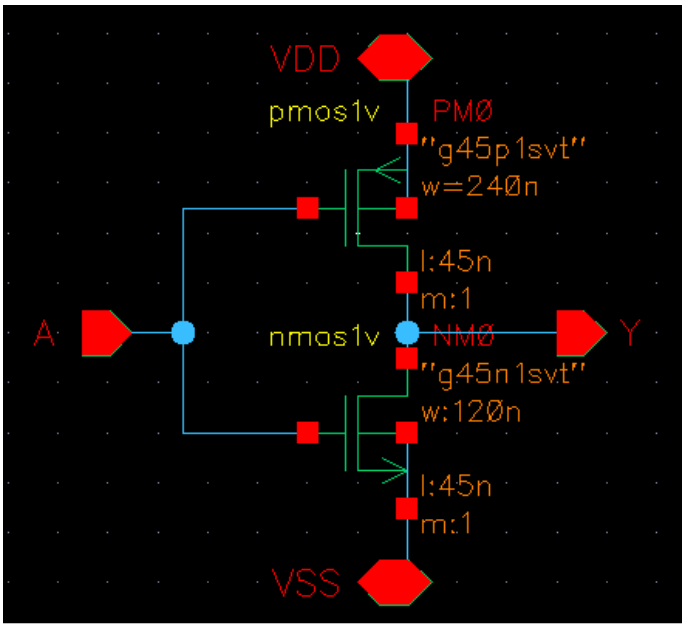
Rise Time (זמן עלייה): זמן העלייה נמדד כ-59.98ps זמן העלייה הוא הזמן שלוקח למתח בכניסה לעלות ממתח נמוך (לוגיקה 0) למתח גבוה (לוגיקה 1). ערך זה חשוב לבחינת מהירות התגובה של המעגל.

Overshoot (חריגה): החריגה נמדדה ב- 1.56E-2% זה מראה עד כמה המתח עבר את הערך הרצוי שלו לפני שהתייצב. חריגה גבוהה יכולה להעיד על בעיות ביציבות המעגל.

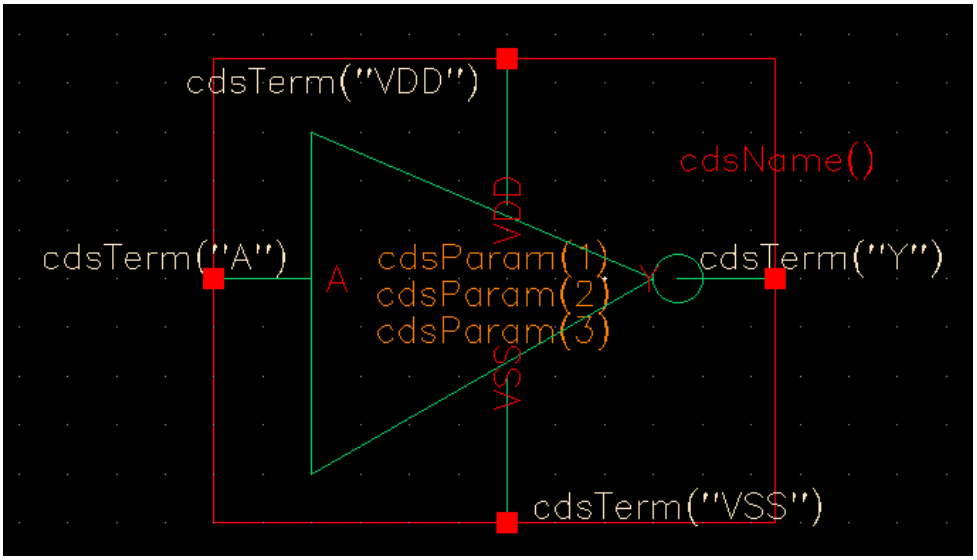
Slew Rate (קצב שינוי): קצב השינוי נמדד ב- 11.0V/ns קצב שינוי מייצג את מהירות השינוי של המתח בזמן המעבר בין הרמות הלוגיות, כלומר כמה מהר המתח עולה או יורד במהלך המעבר. ערך זה משפיע על תכנון המעגלים, במיוחד כשנדרש תזמון מדויק.

נוצר symbol לשער inverter

לשם כך נוצר סכמה חדשה הפעם עם ייצוג כניסה ומוצא עם פינים

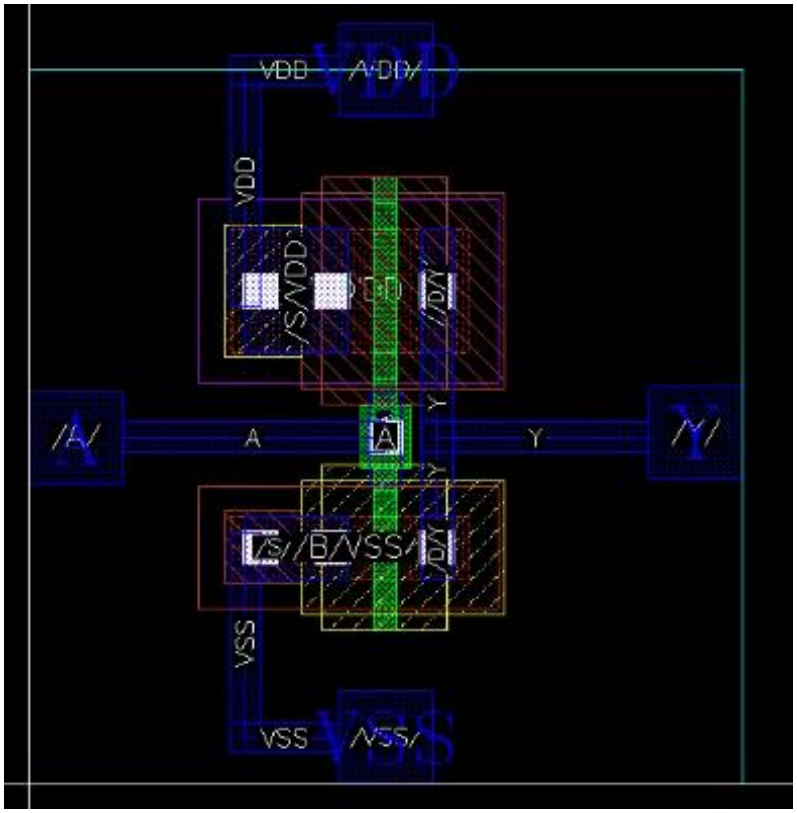


איור 38 סכמת inverter עם פינים



inverter symbol 39 איור

נוצר LAYOUT לשער invreter



inverter layout 40 איור