

הטכניון - מכון טכנולוגי לישראל  
הפקולטה להנדסת חשמל



מעבדה 1

חומר עזר לסטודנט בנושא

תכן סכמתי 1

גרסה 1.24

קיץ 2018

מחברים:

דודי בר-און ליאת שוורץ ואברהם קפלן  
ע"פ חוברת של עמוס זסלבסקי



1	מפענח בינארי - טבלת המרה	4
1.1	פעולות בסיסיות באלגברה בוליאנית	4
1.1.1	עקרונות פעולה וחיווט של שערים	5
1.2	חוקים בסיסיים באלגברה בוליאנית	10
1.3	מימוש פונקציות לוגיות - מפות קרנו	12
1.3.1	שערי NAND ו NOR	14
1.3.2	פעולת XOR	15
1.3.3	רכיבי ניתוב וקידוד מידע - הבורר MUX	17
1.3.4	רכיבי ניתוב וקידוד מידע - המפלג DEMUX	18
1.3.5	רכיבי ניתוב וקידוד מידע - המפענח	19
1.3.6	רכיבי ניתוב וקידוד מידע - המקדד	20
1.3.7	רכיבים בעלי יציאות מתנתקות - Open Drain	21
1.4	רכיבים בעלי יציאות מתנתקות - Tri-State	24
2	שיטות ייצוג של מספרים	26
2.1	שיטות ייצוג של מספרים - בעלי סימן SIGNED	30
3	רכיבים אריתמטיים	32
3.1	רכיבים אריתמטיים של Quartus	33
3.2	רכיבים גנריים	34
3.3	התקני זיכרון בסיסיים - Latch	35
3.4	התקני זיכרון בסיסיים - Gated Latch	38
3.5	התקני זיכרון בסיסיים - Flip-Flop	39
4	מונים	43
4.1	רגיסטרים (אוגרים)	45
4.2	מונים אסינכרוניים	47
4.3	מונה סינכרוני 74160 - BCD - 74161 עשרוני	48
5	ארכיטקטורות של רכיבים מיתכנתים -	50
5.1	Simple Programmable Logic Devices	50
5.2	Complex PLDs	56
5.3	דוגמאות למשאבי חמרה בארכיטקטורות LUT של Altera	60
6	מבנה קובץ TCL	66
7	חומר רקע הדרוש לניסוי במעבדה	66

# 1 מפענח בינארי - טבלת המרה

## 1.1 פעולות בסיסיות באלגברה בוליאנית

טבלת האמת הבאה מגדירה פעולת OR בין שני משתנים.

$A$	$B$	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

טבלת האמת הבאה מגדירה פעולת AND בין שני משתנים.

$A$	$B$	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

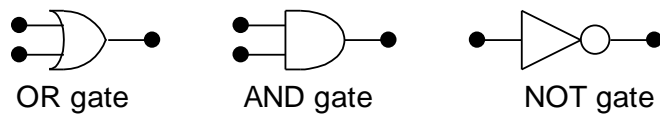
טבלת האמת הבאה מגדירה פעולת NOT על משתנה אחד.

$A$	$\bar{A}$
0	1
1	0

סדר ביצוע הפעולות באלגברה בוליאנית הוא הסדר המקובל הבא :

- סוגריים ( )
- פעולת "גג" (פעולת NOT)
- פעולת כפל לוגי (פעולת AND)
- פעול חיבור לוגי (פעולת OR)

באיור הבא מתוארים שערים הממשים את שלושת הפעולות הנ"ל.

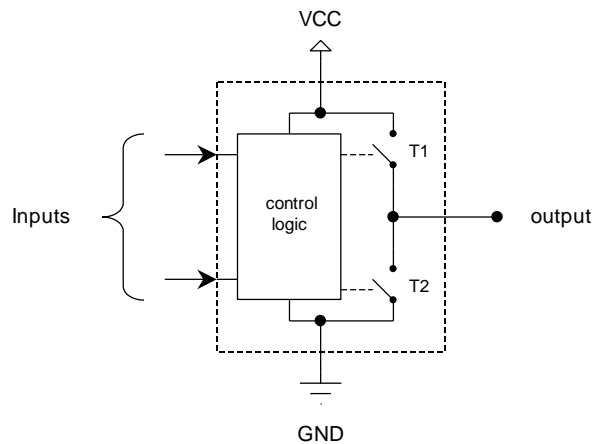


ניתן ליצור גם שערי OR ו AND עם מספר גדול יותר של כניסות. להלן דוגמה לשערים עם שלוש כניסות.



### 1.1.1 עקרונות פעולה וחיווט של שערים

האיור הבא מתאר מבנה חשמלי עקרוני של שער טיפוסי.



השער מחובר למתח ספק שמסומן ב- VCC. נקודת הייחוס שיחסית אליה מודדים את מתח הספק ואת מתחי הכניסות והיציאות במערכת נקראת GND. נניח שמתחים נמוכים, שהם קרובים למתח GND נחשבים למצב '0' לוגי ונניח שמתחים גבוהים, שקרובים למתח הספק VCC נחשבים למצב '1' לוגי. מתחי כניסות הרכיב מוזנים למערכת טרנזיסטורים שמתוארת באיור כ- Control Logic. מערכת זו משפיעה על זוג הטרנזיסטורים שמצויים ביציאה ושסומנו כזוג מתגים T1 ו T2. כאשר נגזר על היציאה להימצא במצב של '1' לוגי המתג הטרנזיסטורי T1 מוליך ל- VCC והמתג הטרנזיסטורי T2 כמעט מנותק מ- GND. המתח שמופק ביציאה במקרה זה הוא מתח גבוה (קרוב ל VCC). כאשר נגזר על היציאה להימצא במצב של '0' לוגי המתג הטרנזיסטורי T2 מוליך ל- GND והמתג הטרנזיסטורי T1 כמעט מנותק מ- VCC. המתח שמופק ביציאה במקרה זה הוא מתח נמוך (קרוב ל- GND).

לאחר שהחלו להשתמש בשערים (החל משנות השישים) הגדירו היצרנים משפחות לוגיות (Logic Families). משפחה לוגית היא אוסף של רכיבים ספרתיים שמתקיים בהם אוסף של כללים משותף. מדובר בהגדרות של פרמטרים חשמליים כמו:

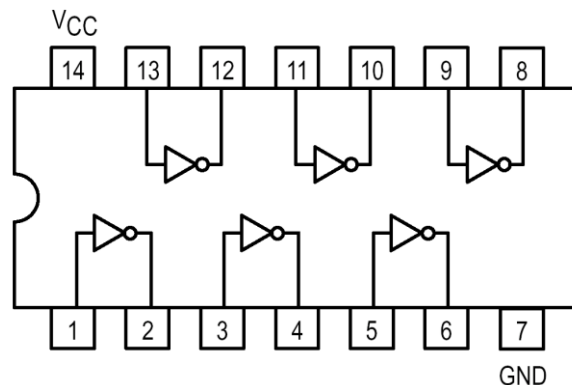
הגדרת מתח ספק אחיד  
הגדרה אחידה של מתחים בכניסות וביציאות  
הגדרה אחידה של זרמים בכניסות וביציאות

בנוסף לפרמטרים החשמליים הנ"ל, משפחה לוגית כללה גם הגדרות של אוסף סטנדרטי של רכיבים שיש להם:

מספור סטנדרטי  
מיקום הדקים סטנדרטי באריזה

קיום של משפחות לוגיות אפשר לחבר בין רכיבים מאותה משפחה וגם כאשר מדובר היה ברכיבים שיוצרו בחברות שונות. החיבור בין הרכיבים היה פשוט וקל ודמה במידה מסוימת לחיבור בין "אבני-לגו" אידיאליות, מבלי שהיה צורך להתעניין באופן מפורט בפרמטרים החשמליים או במבנה הפנימי של הרכיבים. סטנדרטיזציה של משפחות לוגיות גם יצרה תחרות בין יצרנים ועזרה לשמור על מחירים נמוכים.

אחת מהמשפחות הלוגיות הדומיננטיות ביותר שנוצרו בתעשייה ושהחלה להיות מיוצרת בשנות השישים היא משפחה לוגית שנקראת TTL. רכיבים אלו מוספרו בקידומת של הספרות 74. להלן דוגמא לרכיב מהמשפחה שנקרא 7404 ושכולל ארבעה שערי NOT (מהפכים):



סדרת רכיבי ה-74 הייתה מאוד פופולרית ויוצרה על ידי מספר רב של יצרנים. ייצורה הגיע לשיא בשנות השבעים והשמונים. רכיבי הסדרה עדיין מיוצרים בכמויות קטנות עד היום. משפחת רכיבי 74 עברה במשך השנים שיפורים רבים וקמו לה תת-משפחות רבות, בעלות ביצועים טובים יותר (מהירות גבוהה יותר וצריכת הספק נמוכה יותר). רכיבים אלו סומנו בספרות 74 בתוספת של אותיות כמו: 74H, 74L, 74S, 74LS, 74ALS, 74F. רכיבי הסדרה 74 היו עד כדי כך פופולריים, עד שמשפחה לוגית אחרת שנקראת CMOS החלה אף היא להיות מיוצרת כתואמת בצורת הסימון ואף במתחים לרכיבי הסדרה 7400. מדובר בתת משפחות שנקראות למשל בשמות הבאים: 74FCT, 74ACT, 74HCT.

במשפחות TTL משתמשים במתח ספק של +5 Volt. במלים אחרות המתח של VCC הוא +5 Volt (כמובן יחסית ל-GND). מתח הספק יכול לנוע בין 4.75 וולט ל-5.25 וולט וברכיבים מסוימים (צבאיים) יכול לנוע אף בין 4.5 וולט ל-5.5 וולט.

במשפחת TTL, מתחי הכניסה שנחשבים '0' לוגי או '1' לוגי הם המתחים הבאים בהתאמה:

מתח כניסה (בוולטים) שנחשב כ- '0' לוגי מקיים:  $V_{IL} < 0.8$

מתח כניסה (בוולטים) שנחשב כ- '1' לוגי מקיים:  $2 < V_{IH}$

הסימון  $V_{IL}$  בא מהמלים Low Input Voltage. הסימון  $V_{IH}$  בא מהמלים High Input Voltage. הזנה של כניסה של שער TTL במתח שקטן מ-0.8 Volt, מבטיחה שהרכיב יתייחס לכניסה, ככניסה שנמצאת ב- '0' לוגי (Low). הזנה של כניסה של שער TTL במתח שגדול מ-2 Volt, מבטיחה שהרכיב יתייחס לכניסה ככניסה שנמצאת ב- '1' לוגי (High). הזנת הכניסה במתחים שאינם שייכים לאחד מהתחומים הנ"ל (למשל במתח של 1.5 Volt) נחשבת כמצב לוגי שאינו מוגדר, ואין דרך לדעת כיצד הרכיב יתייחס לכניסה שנמצאת במצב זה.

המתחים של המצבים הלוגיים שהוזכרו קודם, התייחסו למתחים בכניסות של הרכיבים. מהם המתחים שמופקים ביציאות של הרכיבים? המתחים שמובטחים ביציאה של רכיבים בסדרת רכיבי ה-TTL הראשונה (שנקראה סדרה 74) במקרה של ב- '0' לוגי או '1' לוגי הם המתחים הבאים בהתאמה:

מתח יציאה (בוולטים) שמובטח במקרה של יציאה שנמצאת ב- '0' לוגי:  $V_{OL} < 0.4$

מתח יציאה (בוולטים) שמובטח במקרה של יציאה שנמצאת ב- '1' לוגי:  $2.4 < V_{OH}$

הסימון  $V_{OL}$  בא מהמלים Low Output Voltage. הסימון  $V_{OH}$  בא מהמלים High Output Voltage. כפי שאפשר לראות, תחומי המתחים שמובטחים לנו ביציאות (בהשוואה לתחומי המתח בכניסות) הם תמיד תחומי מתחים צרים יותר וקרובים יותר למתחי הספק. במלים אחרות מתקיימים הקשרים הבאים:

$$V_{OL} < V_{IL}$$

$$V_{IH} < V_{OH}$$

זוהי תכונה חשובה של המשפחה הלוגית שמאפשרת להתגבר על הפרעות ורעש חשמלי. להפרש בין מתחי היציאה ומתח הכניסה (שעומד ברכיבי TTL המקוריים על 0.4 Volt) קוראים Noise Margin.

רמות המתחים של '0' לוגי ו- '1' לוגי בכניסות של במשפחות CMOS הם בדרך כלל:

$$V_{IL} < \frac{1}{3}V_{CC} \quad \text{מתח כניסה שנחשב כ- '0' לוגי מקיים:}$$

$$V_{CC} \frac{2}{3} < V_{IH} \quad \text{מתח כניסה שנחשב כ- '1' לוגי מקיים:}$$

רמות המתחים ביציאה הן רמות מתחים שהן מאוד קרובות למתחי הספק (0 Volt ו- 5 Volt בהתאמה). היתרון של רכיבי CMOS הוא בהורדה משמעותית בצריכת ההספק. החל מאמצע שנות השמונים החלו להופיע משפחות CMOS שפועלות במתח נמוך יותר של 3.3 Volt.

מספר הרכיבים במשפחות הלוגיות הקלאסיות היה בשיאו בשנות השבעים והשמונים. באותן שנים נהגו אנשים שעסקו בתכן לוגי וספרתי לדפדף בקטלוגים עבי קרס ולבחור מתוכם את הרכיבים שבהם הם היו מעוניינים להשתמש. קטלוגים אלו הכילו מגוון גדול של רכיבים (מאות רכיבים) כמו: שערים מסוגים וגדלים שונים, רכיבי ניתוב מידע מסוגים שונים כמו בוררים מפענחים ומקדדים, רכיבים אריתמטיים כמו מחברים ומשווים, פליפ-פלופים, רגיסטרים, מונים, ורכיבים רבים נוספים.

החל משנות השמונים מספר הרכיבים הסטנדרטיים במשפחות הלוגיות המודרניות הלך ומצטמצם. הסיבה לכך היא פשוטה, בתכן מודרני משתמשים כיום בעיקר ברכיבים מיתכנתים. ברכיבים אלו הפונקציה הלוגית שאותה מבצע הרכיב אינה נקבעת על ידי היצרן, אלא נקבעת על ידי המשתמש. נאמר כבר קודם, שאוסף הכללים שהגדיר מהי משפחה לוגית, כלל בעבר גם אוסף של רכיבים סטנדרטי. כיום, בעידן שבו המשתמש קובע את הפונקציה שאותה מבצע הרכיב הספרתי, אין כבר משמעות להגדרה הרחבה הזו שניתנת למשפחה הלוגית. המושג הרחב של "משפחה לוגית" אינו נפוץ כיום והוא הוחלף בשם I/O Standard (תקן קלט/פלט) שמתאר אך ורק את אוסף הפרמטרים החשמליים של הדקי הכניסות והיציאות של הרכיב.

לרכיבים מיתכנתים רבים כיום יש הדקים שמסוגלים לפעול בתקני קלט/פלט של TTL ו- CMOS אך גם עם תקני קלט/פלט מודרניים יותר. תקני קלט/פלט ורכיבים מודרניים משתמשים כיום בדרך כלל במתחי ספק נמוכים יותר מ- 5 Volt. להלן כמה דוגמאות לתקני קלט/פלט כאלה.

LVTTTL (Low Voltage TTL - General purpose)

LVC MOS (Low Voltage CMOS - General purpose)

הדקים שפועלים בתקן קלט/פלט LVTTTL (Low Voltage TTL) הם הדקים שפועלים במתח ספק של 3.3 Volt ומסוגלים במקרים רבים להתחבר גם לרכיבי TTL ו- CMOS שפועלים במתח של 5 Volt. הדקים שפועלים בתקני קלט/פלט LVC MOS הם הדקי CMOS שפועלים במתח ספק נמוך יותר מ- 5 Volt. המתחים הנמוכים המקובלים כיום הם המתחים הבאים: 2.5 Volt, 3.3 Volt, 1.8 Volt ו- 1.5 Volt. המגמה של הירידה במתח ברכיבים מודרניים מבטיחה חיסכון בהספק. רכיבי CMOS שפועלים למשל במתח ספק של 2.5 Volt צורכים רבע מכמות ההספק שצורכת משפחה

דומה שפועלת במתח ספק של 5 Volt. צריכת ההספק הנמוכה חשובה לא רק בכדי להוריד את הדרישות החיצוניות מספק המתח. רכיבים בעלי צפיפות גבוהה חייבים לפעול בהספק נמוך גם על מנת למנוע חימום יתר של הרכיב.

תקני הקלט/פלט הנ"ל הם תקנים כלליים. ברכיבים מיתכנתים מודרניים קיימים גם תקני הקלט/פלט שמיועדים להתחברות עם התקנים מיוחדים כמו מעבדים, רכיבי זיכרון, התקני BUS מיוחדים של מחשב PC ורכיבים מיוחדים (כמו AGP, CTT, PCI, HSTL, SSTL, GTL ועוד).

הורדת המתח מקטינה את ההגנה כנגד רעש. כתוצאה מכך בשנים האחרונות אנו עדים לשימוש באמצעים הבאים:

רכיבים שפועלים בכמה מתחים (Multi-Volt Devices)  
שימוש בתקני קלט/פלט דיפרנציאליים (Differential I/O Standards).

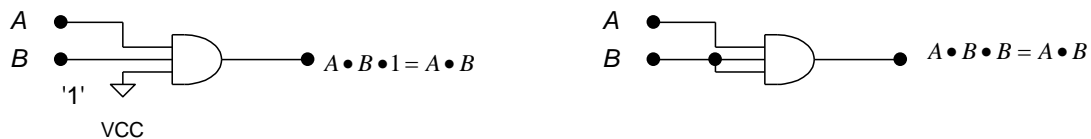
רכיבים שפועלים בכמה מתחים עשויים לעבוד באופן פנימי במתח נמוך ובאופן חיצוני במתח גבוה יותר. רכיב כזה יחובר לשני מתחי ספק. ספק אחד שיחובר למשל באמצעות הדקים שנקראים VCCINT יספק מתח נמוך למערכת הפנימית (Internal Core) של הרכיב. ספק מתח נוסף שיתחבר לרכיב למשל באמצעות הדקים שנקראים VCCIO יקבע את מתח העבודה החיצוני של ההדקים. מתח העבודה הגבוה יותר יאפשר לשמור על Noise Margin גבוה יותר מחוץ לרכיב.

השימוש בתקני קלט/פלט דיפרנציאליים (Differential I/O Standards) מאפשר לשמור על חסינות לרעש גם כאשר מתח העבודה הוא מתח נמוך. התכונה המשותפת של תקנים דיפרנציאליים היא שהשידור והקליטה בין שני רכיבים נעשית באמצעות זוג חוטים שהמתחים בהם הם הפוכים (לדוגמה: LVPECL, LVDS, PCML ועוד).

כאשר מחוטים מערכת עם שערים רגילים צריך להקפיד על כללי החיווט הבאים:

אסור להשאיר כניסה באוויר  
אסור לחוות בין שתי יציאות

כניסות TTL שאינן מחוברות (שנשארות באוויר) מתנהגות ברוב המקרים כאילו יש בהן '1' לוגי. למרות מה שנאמר כאן, מאוד לא כדאי להשאיר כניסות באוויר מכיוון שמדובר במתח של '1' לוגי גבולי שרגיש מאוד לרעש חשמלי, ושמהווה מקור בלתי אכזב לתקלות מרגיזות שקשה לאתרן. השארת כניסה של רכיב CMOS באוויר יוצרת מצב לוגי שעלול להיות בלתי מוגדר מבחינה לוגית ועלול אף להזיק לרכיב. לכן תמיד יש לחבר כניסות אלו למצב לוגי מוגדר כל שהוא. נשתמש בדוגמה. נניח שבידנו יש שער AND בעל שלוש כניסות, אך אנו מעונינים להשתמש רק בשתיים מכניסותיו. נוכל לבצע חיבור של הכניסה המיותרת באחד מהאופנים הבאים:

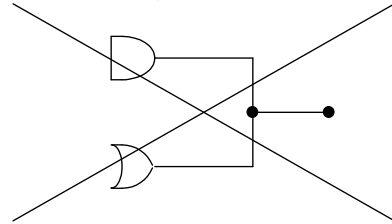


בכל מקרה של כניסות שאינן בשימוש, יש להחליט לאן לחברן כך שהן לא תפרענה. ראה דוגמאות נוספות באחת משאלות ההכנה.

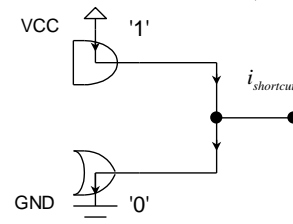
כל מה שנאמר כאן חל על כניסות בלבד. בהחלט מותר להשאיר יציאות שלא בשימוש באוויר.



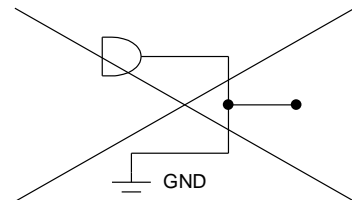
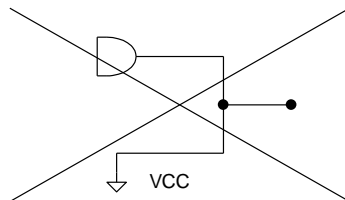
מדוע אסור לחוות בין שתי יציאות ?



חיבורים מסוג זה עלולים ליצור קונפליקטים, כאשר שער אחד מנסה למשל לאלץ בנקודת החיבור מצב של '0' לוגי ושער אחר מנסה למשל לאלץ בנקודת החיבור '1'. חיבורים מסוג זה עלולים ליצור בנקודת החיבור מתחים שאינם מוגדרים מבחינה לוגית ולעתים חיבורים מסוג זה אף עלולים להזיק לרכיבים. הנזק נגרם כתוצאה מכך שזרם גבוה זורם מ  $VCC$  ל  $GND$ . לדוגמה, נניח שהשער העליון באיור "מנסה" לאלץ מצב של '1' והשער התחתון מנסה לאלץ '0' לוגי.



בשער העליון נוצרת הולכה של כמעט קצר בין  $VCC$  והיציאה של השער. בשער התחתון נוצרת הולכה של כמעט קצר בין היציאה של השער ו  $GND$ . בסופו של דבר נוצרת זרימה חזקה בין  $VCC$  ל  $GND$  דרך טרנזיסטורים פנימיים שמוליכים במצב שהוא כמעט קצר. זרמים אלו גורמים להתחממותם של הרכיבים שיכולה כמובן לגרום לנזק. רק ברכיבים בעלי יציאות מיוחדות שנקראות Tri-State ו Open-Collector (נזכיר אותם בהמשך), ניתן לעתים לחבר חבורים מסוג זה. כמובן שגם אסור בשום פנים ואופן לחוות יציאה למתח ספק כמו  $VCC$  או  $GND$  ! זהו חיבור קטלני שעלול ברוב המקרים להזיק לשער.



אין כמובן איסור לחבר שתי כניסות לנקודה משותפת !

תכנת הפיתוח שלנו Quartus (שבה נשתמש בניסוי) מאפשרת להשתמש באופן גרפי בשערים. השערים של Quartus שייכים כולם למשפחת רכיבים שנקראים Primitives. דוגמאות לרכיבים מסוג זה הם למשל השערים הבאים :

שערי AND שנקראים : AND2, AND3, AND4, ...

שערי OR שנקראים : OR2, OR3, OR4, ...

שער NOT שנקרא : NOT

קיימים גם שערים דומים נוספים כמו NOR, NAND ו XOR. תכנת ה - Quartus תומכת גם בכל סדרת רכיבי ה - 74. בדרך כלל כאשר רוצים להשתמש בשערים לא כדאי להשתמש ברכיב 74 אלא ב Primitives הני"ל. השימוש ברכיבי 74 נפוץ יותר כאשר מדובר ברכיבים גדולים יותר כמו מחברים, משוים, מונים ורגיסטרים. קיימים גם רכיבי שערים גמישים שנקראים LPM\_OR ו LPM\_AND.

## 1.2 חוקים בסיסיים באלגברה בוליאנית

מתקיימים חוקי חילוף (Commutative Laws) על כפל וחיבור :

$$A + B = B + A$$

$$A \bullet B = B \bullet A$$

מתקיימים חוקי קיבוץ (Associative Laws) על כפל וחיבור :

$$(A + B) + C = A + (B + C)$$

$$(A \bullet B) \bullet C = A \bullet (B \bullet C)$$

בעצם בכתיבת חיבור של כמה אופרנדים או בכתיבת כפל של כמה אופרנדים, אין צורך להשתמש כלל בסוגריים.

$$(A + B) + C = A + (B + C) = A + B + C$$

$$(A \bullet B) \bullet C = A \bullet (B \bullet C) = A \bullet B \bullet C$$

הכלל הנ"ל חל גם במקרה של כתיבת מספר גדול יותר של אופרנדים.

החוק הבא מתאר תוצאה של חיבור או כפל של משתנים זהים (Idempotent Law).

$$A + A = A$$

$$A \bullet A = A$$

לשני החוקים הנ"ל קוראים לעתים גם בשם "חוקי ספיגה" (Absorption Laws). ניתן להרחיב את החוקים הנ"ל למספר גדול יותר של משתנים זהים.

להלן חוקים נוספים שקשורים לפעולת החיבור הלוגי והכפל הלוגי עם קבועים (שנקראים Identity Law ו Null Law):

$$A + 0 = A$$

$$A + 1 = 1$$

$$A \bullet 0 = 0$$

$$A \bullet 1 = A$$

חוקי הפילוג (Distributive Laws) משתמשים הן בפעולות החיבור הלוגי והן בפעולות הכפל הלוגי.

$$A \bullet (B + C) = A \bullet B + A \bullet C$$

$$A + B \bullet C = (A + B) \bullet (A + C)$$

חוק הפילוג הראשון הוא פילוג של כפל על חיבור והוא קיים גם באלגברה רגילה. חוק הפילוג השני הוא פילוג של חיבור על כפל, והוא אינו קיים באלגברה רגילה. חוקים אלו מתקיימים גם במקרה של פילוג פעולה על מספר גדול יותר של אפרנדים.

שני החוקים הבאים קשורים לפעולת ההיפוך (גג) והם נקראים חוקי החלקים משלימים (Complementary Parts Laws)

$$A + \bar{A} = 1$$

$$A \bullet \bar{A} = 0$$

גם החוקים הבאים קשורים לפעולת המשלים והם נקראים חוקי המשלים העצמי (Involution laws או Multiple negation Laws) וניתן להרחיבם למספר זוגי או איזוגי גדול יותר של גגות.

$$\overline{\overline{A}} = A$$

$$\overline{\overline{\overline{A}}} = \overline{A}$$

החוקים הבאים נקראים חוקי De-Morgan (או Dualization Laws) והם משתמשים בשלוש הפעולות הבסיסיות שקיימות באלגברה בוליאנית.

$$\overline{A + B} = \bar{A} \bullet \bar{B}$$

$$\overline{A \bullet B} = \bar{A} + \bar{B}$$

חוקים אלו מתקיימים גם במקרה של מספר גדול יותר של אפרנדים.

באלגברה בוליאנית קיים עקרון שנקרא "עקרון הדואליות" (The principle of duality). עקרון הדואליות הוא בעצם חוק שחל על החוקים של אלגברה בוליאנית. עקרון הדואליות טוען שכל חוק שנגזר מחוקי היסוד של האלגברה נשאר תקף גם אם מבצעים את ההחלפות הבאות:  
החלפה בין הפעולות  $\bullet$  ו  $+$

החלפה בין האלמנטים 0 ו 1

חוסר הסימטריה היחידה שקיים בין הפעולות  $\bullet$  ו  $+$  הוא רק בהסכם על סדר ביצוע הפעולות (פעולת  $\bullet$  מתבצעת לפני פעולת  $+$ ).

### 1.3 מימוש פונקציות לוגיות - מפות קרנו

רכיבים מיתכנתים פשוטים SPLDS (Simple Programmable Logic Devices) רבים, ושיוזכרו במשך, בנויים בארכיטקטורה של סכום מכפלות (Sum Of Products או בקצור SOP). ארכיטקטורות אלו מתאימות למימוש באמצעות פונקציות בעלות המבנה הבא :

$$f(A, B, C, \dots) = \overline{A}C.. + AB.. + \overline{A}\overline{B}C + \dots$$

הביטוי בצד ימין מכיל אוסף כל שהוא של מכפלות (Products) של אותיות (Literals) של הפונקציה. האותיות של הפונקציה יכולות להיות עם וללא היפוך לוגי. כאשר הביטוי בצד ימין מכיל את כל האותיות של הפונקציה, אנו אומרים שהפונקציה רשומה בצורת סכום מכפלות מלאות (או בצורה קנונית שלה). ניתן לרשום פונקציה בצורה קנונית ישירות מטבלת האמת שלה. להלן דוגמה.

A B C	$f(A, B, C) = ?$
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	1
1 1 1	1

$$f(ABC) = \sum(3,5,6,7) = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

המימוש הקנוני הנ"ל מקובל ברכים מיתכנתים מסוג ROM ובנגזרות הטכנולוגיות שלהם (PROM, EPROM, EEPROM .. וכו' שיוזכרו בהמשך).

בדרך כלל ברוב רכיבי ה- SPLD (בארכיטקטורות שמבוססות על רכיבי PLA ו PAL) שבהם לא קיימות כל המכפלות המלאות האפשריות, מעדיפים שלא להשתמש בצורת הרישום הקנונית של הפונקציה ומבצעים לפני המימוש צמצום (Reduction) או מינימוזציה (Minimization) לפונקציה. שיטות מינימוזציה (ידניות) מקובלות הן למשל: שיטת מפות קרנו ושיטת Quine-Mcclusky. אלגוריתם צמצום (ממוחשב) הוא למשל: ESPRESSO. להלן דוגמה לצמצום ידני הפונקציה הנ"ל באמצעות מפת קרנו :

BC \ A	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$f(ABC) = AB + AC + BC$$

צמצום יכול להיעשות על כל פונקציה בנפרד (מקובל בארכיטקטורות PAL) או באופן קבוצתי על כמה פונקציות (על מנת למצוא מאגר מכפלות מינימלי – שקיים ברכיבי PLA שיש להם מאגר מכפלות משותף).

ברכיבי PAL ו PLA רבים, קיימת אפשרות לממש פונקציה בקוטביות הפוכה. לשם כך הופכים את הפונקציה פעמיים: פעם אחת בטבלה שמגדירה את הפונקציה ופעם נוספת בפונקציה - כלומר בחמרה. להלן הדוגמה הקודמת במימוש בקוטביות הפוכה:

BC \ A	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$$f(ABC) = \overline{\overline{A \bullet B} + \overline{A \bullet C} + \overline{B \bullet C}}$$

נקרא למימוש בצורת סכום מכפלות  $SOP$  ולצורת המימוש ההפוכה בשם  $\overline{\overline{SOP}}$ . בדוגמה של הפונקציה הנ"ל, המימוש בקוטביות הפוכה לא היה פשוט יותר. לעתים המימוש בקוטביות הפוכה עשוי להיות מצומצם יותר. להלן דוגמה.

$$g(A,B,C,D,E,F)_{SOP} = AD + AE + AF + BD + BE + BF$$

$$g(A,B,C,D,E,F)_{\overline{\overline{SOP}}} = \overline{\overline{A \bullet B \bullet C} + \overline{D \bullet E \bullet F}}$$

הצורות הדואליות לצורות המימוש של  $SOP$  ו  $\overline{\overline{SOP}}$  הן צורות מימוש  $POS$  ו  $\overline{\overline{POS}}$  כאשר המשמעות של האותיות  $POS$  היא: מכפלת סכומים (Product Of Sums). כמובן שגם כאן קיימים מימושים קנוניים ומינימליים. להלן מימוש של הפונקציה הנ"ל בצורה הקנונית השנייה:

$$f(ABC)_{CANONICAL\_POS} = \Pi(0,1,2,4) = (A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+B+C)$$

להלן מימוש של הפונקציה הנ"ל בצורת  $POS$  מינימלי.

BC \ A	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$f(ABC)_{POS} = (A+B)(A+C)(B+C)$$

להלן מימוש של הפונקציה הנ"ל בצורת  $\overline{\overline{POS}}$  מינימלי.

BC \ A	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$$f(ABC)_{\overline{\overline{POS}}} = \overline{\overline{(\overline{A+B})(\overline{A+C})(\overline{B+C})}}$$

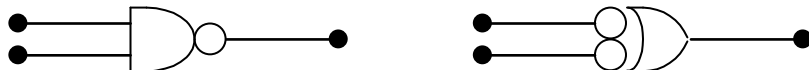
כמובן שניתן לממש פונקציות גם בצורות רבות נוספות שאינן שייכות לאחת מצורות המימוש הנ"ל ושמיכלול יותר משתי רמות לוגיות של שערים.

### 1.3.1 שערֵי NAND ו־NOR

הטבלה הבאה מגדירה פעולת NAND על שני אופרנדים:

$A$	$B$	$\overline{A \cdot B} = \overline{A} + \overline{B}$
0	0	1
0	1	1
1	0	1
1	1	0

באיור הבא מתוארות שתי צורות אפשריות לתיאור שער NAND.



הטבלה הבאה מגדירה פעולת NOR על שני אופרנדים:

$A$	$B$	$\overline{A + B} = \overline{A} \cdot \overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

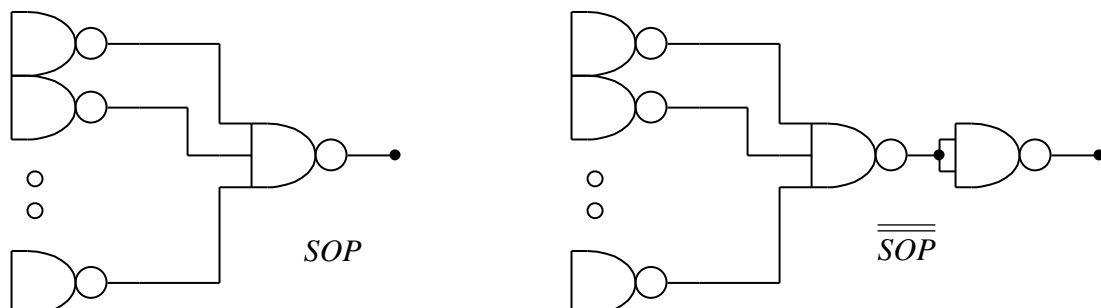
באיור הבא מתוארות שתי צורות אפשריות לתיאור שער NOR.



שערֵי NAND ו־NOR יכולים להיות בעלי מספר גדול יותר של כניסות.



שערֵי NAND ושערֵי NOR מהווים כל אחד בעצמו מערכת שלמה (כלומר אפשר ליצור מכל אחד מסוגי השערים הנ"ל את כל הפעולות של אלגברה בוליאנית). קל להמיר פונקציה בצורת סכום מכפלות - (Sum Of Products) או סכום מכפלות הפוך למימוש באמצעות שערֵי NAND.



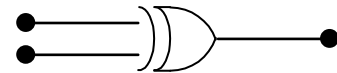
באופן דומה ניתן להמיר מימוש פונקציה שכתובה בצורת מכפלת סכומים (POS) למימוש באמצעות שערֵי NOR.

## 1.3.2 פעולת XOR

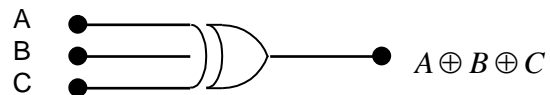
הטבלה הבאה מגדירה פעולת XOR על שני אופרנדים:

$A$	$B$	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

באיור הבא מתואר שער XOR של שתי כניסות.



ניתן להרחיב את ההגדרה של שער XOR למספר גדול יותר של כניסות כשער שמפיק ביציאה '1' לוגי כאשר מספר "האחדים" בכניסותיו הוא אי-זוגי (Odd). להלן דוגמה לשער XOR עם מספר גדול יותר של כניסות:



בשערי XOR מקיימים חוקי חילוף וקיבוץ ואין צורך להשתמש בסוגריים כאשר רושמים כמה פעולות XOR רצופות:

$$A \oplus B = B \oplus A$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

מתקיים חוק פילוג של פעולת כפל (AND) על XOR ומבחינת קביעת סדר הפעולות ל - OR ו XOR יש בדרך כלל עדיפות זהה (הנמוכה ביותר) כלומר מתקיים:

$$A \bullet (B \oplus C) = A \bullet B \oplus A \bullet C$$

פעולת XOR מקיימת את החוקים הבאים עם הקבועים '0' ו '1':

$$A \oplus 0 = A$$

$$A \oplus 1 = \bar{A}$$

בחוקים הנ"ל משתמשים באופן נרחב להפיכת קוטביות מתוכנתת של אותות ברכיבים מיתכנתים (נראה בהמשך).

ביצוע XOR על ביטויים זהים או הפוכים מקיים את שני החוקים הבאים:

$$A \oplus A = 0$$

$$A \oplus \bar{A} = 1$$

בפעולות XOR נשמרת הזוגיות או האי-זוגיות של מספר הגגות :

$$\overline{A \oplus B} = A \oplus \overline{B} = \overline{\overline{A \oplus B}} = \overline{\overline{A \oplus B}}$$

$$\overline{A \oplus \overline{B}} = A \oplus B = \overline{\overline{A \oplus B}} = \overline{A \oplus B}$$

תכונה זו מתקיימת גם כאשר מדובר על ביטויים שכוללים מספר גדול יותר של אופרנדים (למשל בדוגמה הבאה).

$$A \oplus \overline{B} \oplus C \oplus \overline{D} = \overline{A \oplus \overline{B} \oplus C \oplus \overline{D}} = A \oplus B \oplus C \oplus D$$

$$\overline{A \oplus \overline{B} \oplus C \oplus \overline{D}} = \overline{\overline{A \oplus \overline{B} \oplus C \oplus \overline{D}}} = A \oplus \overline{B} \oplus C \oplus D$$

מותר להעביר מאגף לאגף במשוואות עם ביטויים מסוג XOR (חוק ההצפנה) :

$$A = B \oplus C \leftrightarrow A \oplus C = B$$

חוק נוסף הוא : שאם הפונקציות  $f1$  ו  $f2$  זרות (אורתוגונליות) אחת לשניה (כלומר  $f1 \cdot f2 \equiv 0$ ) אז מתקיים הקשר הבא בין הפונקציות :  $f1 + f2 = f1 \oplus f2$ .

שילוב של שתי הפעולות XOR ו AND מהווה מערכת שלמה. שתי פעולות אלו יכול להוות בסיס לאלגברה (Reed Miler) שמתנהגת כשדה (כלומר מקיימת חוקים דומים לאלגברה רגילה).

פונקציית XOR אינה ניתנת לצמצום כסכום מכפלות או כמכפלת סכומים. ניתן לראות זאת למשל בדוגמה הבאה של מפת קרנו שמתארת פונקציית XOR של ארבע כניסות :

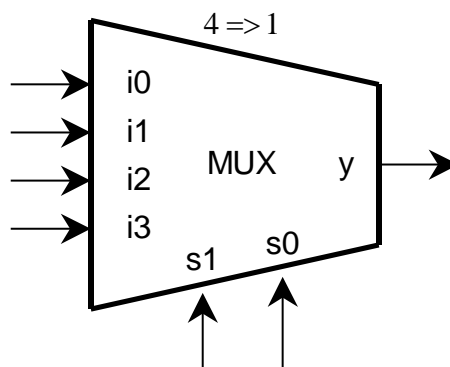
$$A \oplus B \oplus C \oplus D$$

CD \ AB	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0



### 1.3.3 רכיבי ניתוב וקידוד מידע – הבורר MUX

באיור הבא מתואר בורר (Multiplexer) בעל ממדים  $4 \Rightarrow 1$ .



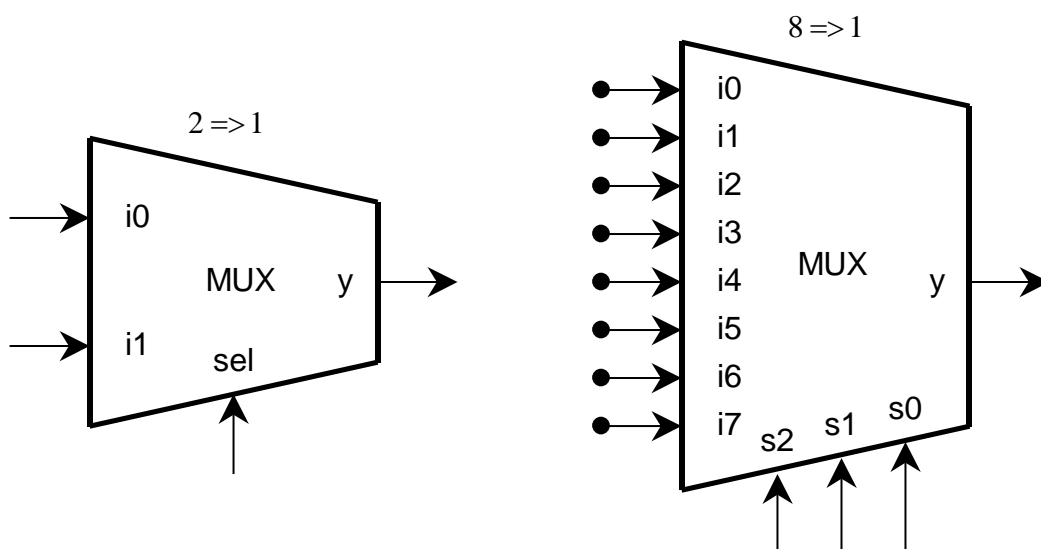
הבורר הנ"ל בוחר להעביר את אחת מכניסות המידע שלו ( $i0$ ,  $i1$ ,  $i2$  או  $i3$ ) ליציאת המידע  $y$ , בשליטה של הקוד הבינארי בכניסות הבקרה  $s1$  ו  $s0$ . להלן טבלת אמת שמתארת את התנהגות הרכיב:

s1	s0	y
0	0	$i0$
0	1	$i1$
1	0	$i2$
1	1	$i3$

המשוואה הבאה מתארת את הבורר:

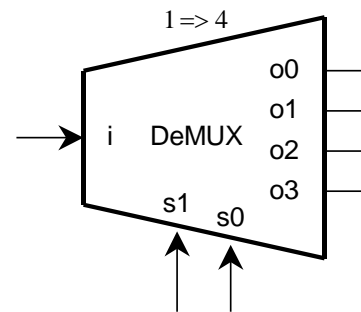
$$y = \overline{s1} \cdot \overline{s0} \cdot i0 + \overline{s1} \cdot s0 \cdot i1 + s1 \cdot \overline{s0} \cdot i2 + s1 \cdot s0 \cdot i3$$

בוררים יכולים להיות בעלי ממדים גדולים או קטנים יותר. להלן דוגמאות:



### 1.3.4 רכיבי ניתוב וקידוד מידע – המפלג DEMUX

המפלג (De-Multiplexer) מבצע פעולה הפוכה מזו של הבורר. באיור הבא מתואר מפלג בעל ממדים  $1 \Rightarrow 4$ .



המפלג הנ"ל בוחר להעביר את המצב הלוגי שבכניסת המידע שלו  $i$ , לאחת מיציאות המידע שלו ( $o0$ ,  $o1$ ,  $o2$  או  $o3$ ), בשליטה של הקוד הבינארי בכניסות הבקרה  $s1$  ו  $s0$ . יציאה שלא נבחרה מפיקה '0' לוגי. להלן טבלת אמת שמתארת את התנהגות הרכיב:

$s1$	$s0$	$o0$	$o1$	$o2$	$o3$
0	0	$i$	0	0	0
0	1	0	$i$	0	0
1	0	0	0	$i$	0
1	1	0	0	0	$i$

המשוואות הבאות מתארות את המפלג:

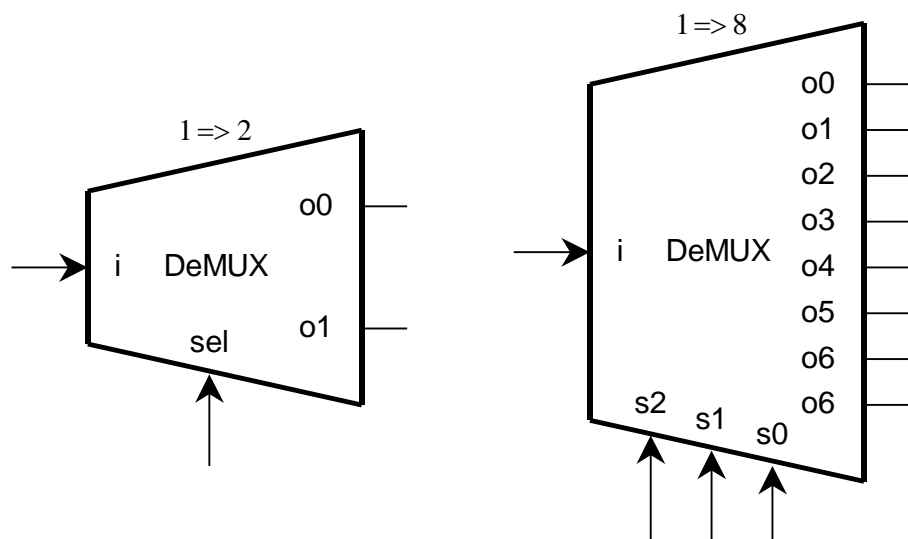
$$o0 = \overline{s1} \cdot \overline{s0} \cdot i$$

$$o1 = \overline{s1} \cdot s0 \cdot i$$

$$o2 = s1 \cdot \overline{s0} \cdot i$$

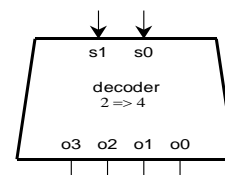
$$o3 = s1 \cdot s0 \cdot i$$

מפלגים יכולים להיות בעלי ממדים גדולים או קטנים יותר. להלן דוגמאות:



### 1.3.5 רכיבי ניתוב וקידוד מידע - המפענח

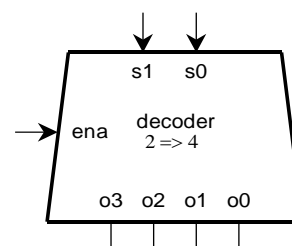
המפענח (Decoder) מבצע פעולת הפיכה של קוד בינארי, לקוד ישיר (קוד שבו סיבית אחת בלבד נבחרת). באיור הבא מתואר מפענח בעל ממדים  $2 \Rightarrow 4$ .



להלן טבלת אמת שמתארת את התנהגות הרכיב:

s1	s0	o0	o1	o2	o3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

בדרך כלל יש למפענחים גם כניסת אפשר (enable).



להלן טבלת אמת שמתארת את התנהגות הרכיב הנ"ל עם כניסת האפשר:

ena	s1	s0	o0	o1	o2	o3
0	$\Phi$	$\Phi$	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

המשוואות הבאות מתארות את המפענח:

$$o0 = ena \cdot \overline{s1} \cdot \overline{s0}$$

$$o1 = ena \cdot \overline{s1} \cdot s0$$

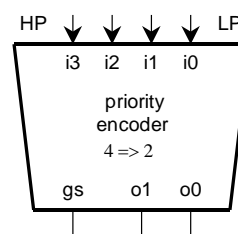
$$o2 = ena \cdot s1 \cdot \overline{s0}$$

$$o3 = ena \cdot s1 \cdot s0$$

המשוואות הנ"ל זהות למשוואות של המפלג. במלים אחרות, מפענח עם כניסת אפשר, זהה למפלג. לעתים קוראים לרכיב הזה בשם הכפול: Decoder/De-Multiplexer. כמובן שמפענחים יכולים להיות קטנים וגדולים יותר (דוגמאות:  $1 \Rightarrow 2$ ,  $3 \Rightarrow 8$ ,  $4 \Rightarrow 16$ ..).

### 1.3.6 רכיבי ניתוב וקידוד מידע - המקדד

להלן דוגמה של מקדד (Encoder) בעל ממדים  $4 \Rightarrow 2$ .

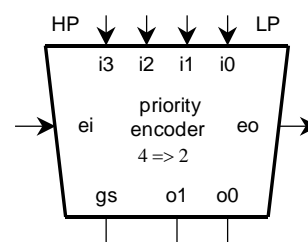


המקדד מבצע פעולה הפוכה מזו של המפענח, כלומר הוא ממיר קוד ישיר (סיבית אחת נבחרת) לקוד בינארי. הבעיה בהמרה כזו היא, שקיימים צירופי כניסה אפשריים שבהם יותר מכניסה אחת נבחרת, למשל גם  $i3$  וגם  $i1$  נבחרו בו זמנית. במקרה הנ"ל מפיקים ביציאה את הקוד הבינארי של הכניסה הגבוהה יותר ("11"). לרכיב כזה קוראים מקדד עדיפויות (Priority Encoder). להלן טבלת אמת של הרכיב הנ"ל:

i3	i2	i1	i0	o1	o0	gs
1	$\Phi$	$\Phi$	$\Phi$	1	1	1
0	1	$\Phi$	$\Phi$	1	0	1
0	0	1	$\Phi$	0	1	1
0	0	0	1	0	0	1
0	0	0	0	0	0	0

היציאה gs (Group Select) היא יציאה שפעילה כאשר לפחות אחת מהכניסות נבחרה. המטרה של יציאה זו היא ליצור הבדל בין המקרה שאף אחת מהכניסות לא נבחרה (יש "00" ביציאה הבינארית) והכניסה בעלת העדיפות הנמוכה בלבד  $i0$  נבחרה (גם במקרה זה יש "00" ביציאה הבינארית).

כאשר רוצים לאפשר שרשרת של Priority Encoders, בכדי ליצור רכיבים גדולים יותר, מוסיפים לרכיב כניסת אפשרור כניסה (Enable Input) ויציאת אפשרור (Enable Output).



להלן טבלת אמת של הרכיב:

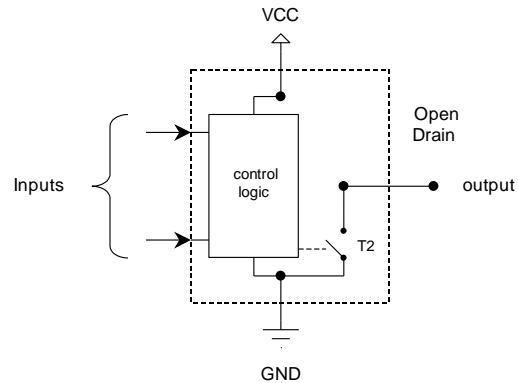
ei	i3	i2	i1	i0	o1	o0	gs	eo
0	$\Phi$	$\Phi$	$\Phi$	$\Phi$	0	0	0	0
1	1	$\Phi$	$\Phi$	$\Phi$	1	1	1	0
1	0	1	$\Phi$	$\Phi$	1	0	1	0
1	0	0	1	$\Phi$	0	1	1	0
1	0	0	0	1	0	0	1	0
1	0	0	0	0	0	0	0	1

### 1.3.7 רכיבים בעלי יציאות מתנתקות – Open Drain

קיימים שני סוגים של רכיבים בעלי יציאות מתנתקות.

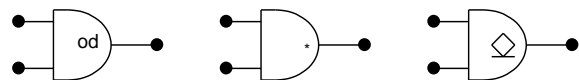
רכיבי Open Drain (או Open Collector במשפחות TTL)  
רכיבי Tri-State

להלן המבנה של שער שהיציאה שלו היא מסוג Open Drain.

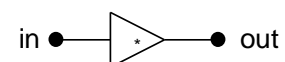


כאשר נגזר על היציאה של הרכיב הזה להיות ב- '0' לוגי המתג הטרנזיסטורי T2 מוליך. במקרה זה מתח היציאה יהיה נמוך וקרוב למתח GND. כאשר נגזר על היציאה להיות ב- '1' לוגי המתג הטרנזיסטורי T2 מנותק. היות ואין גם הולכה כל שהיא גם לכיוון VCC היציאה פשוט מנותקת במצב זה (בהמשך נקרא למצב זה בשם Z – שמשמל אימפדנס גבוה).

כל יציאה של כל רכיב יכולה להיות מסוג Open-Drain. יציאה מסוג Open Drain יכולה להיות מסומנת באחת מהדרכים הבאות:



נתאר את פעולתו של שער שאינו מבצע פעולה לוגית כל שהיא (למשל שער AND ששתי הכניסות שלו מחוטטות לכניסה משותפת) ושהיציאה שלו היא מסוג Open-Drain.

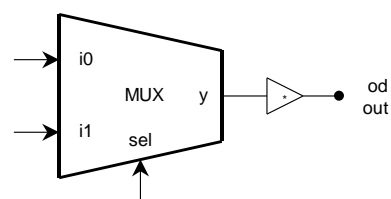


לרכיב כזה קוראים Open-Drain Buffer. להלן טבלת אמת שמתארת את הרכיב:

in	out
0	0
1	Z

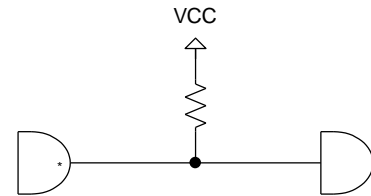
Low voltage  
High Impedance

באמצעות הרכיב הנ"ל ניתן להפוך כל יציאה רגילה של רכיב ליציאה מסוג Open Drain. להלן דוגמה:



כמובן שרכיב שהיציאה שלו היא מסוג Open Drain, לא יכול להתחבר באופן רגיל לכניסה של שער אחר, מכיוון שהכניסה של השער תהייה בעצם מוזנת מנתק וכפי שכבר ראינו באחד החלקים הקודמים, זהו חיווט אסור.

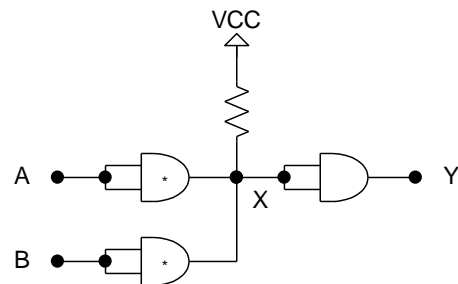
כאשר מחברים יציאה מסוג Open Drain לכניסה של רכיב, יש לעשות זאת באמצעות שימוש בנגד Pull-Up Resistor באופו הבא.



כאשר נגזר על היציאה להיות ב- '0' היציאה של הרכיב מוליכה ל- GND ובהנחה שהנגד אינו קטן מדי המתח הנמוך יגרום לכניסה להיות מוזנת ב- '0' לוגי תקין. כאשר נגזר על היציאה להימצא ב- '1' לוגי היציאה תהייה מנותקת. במקרה זה זרם הכניסה לשער הוא נמוך ואם הנגד לא יהיה גדול מדי הכניסה תזון במתח שהוא מאוד קרוב ל- VCC שמהווה '1' לוגי תקין.

ל- '1' לוגי שנוצר באמצעות נגד Pull-Up קוראים לפעמים בשם '1' לוגי חלש (להבדיל מ- '1' לוגי שנוצר באמצעות טרנזיסטור Pull-Up). בשיטת חיבור זו ניתן לחבר גם בין שתי משפחות לוגיות שפועלות במתחי ספק שונים.

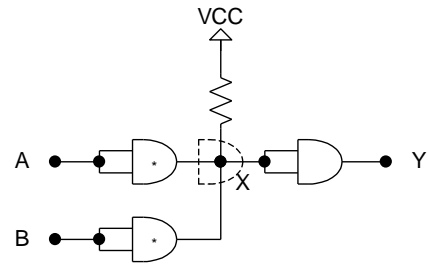
באיור הבא חיווטו שני Open-Drain Buffers לנקודה משותפת שמחווטת באמצעות נגד Pull-Up Resistor ל VCC. שער ה- AND מימין הוא Buffer (שאינו Open Collector).



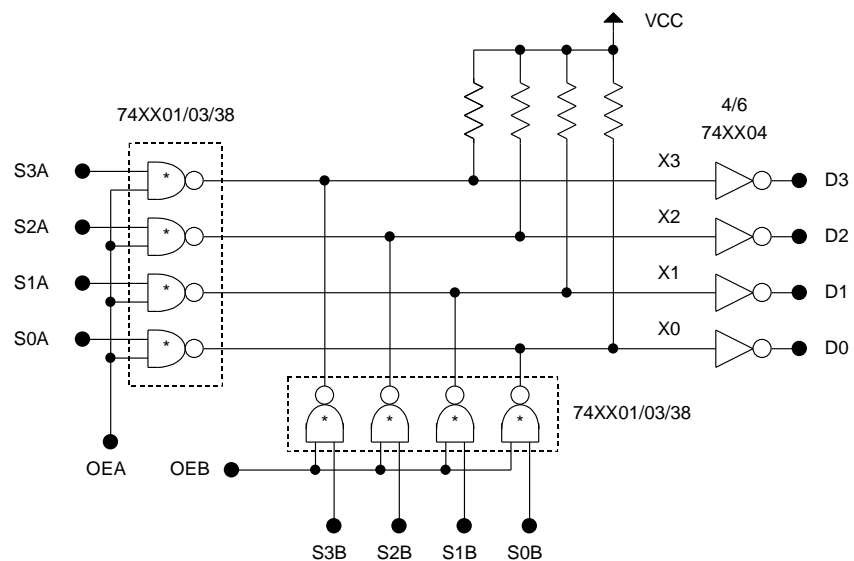
ננסה להבין את פעולת המערכת הנ"ל באמצעות הטבלה הבאה :

A	B	יציאה של השער העליון	יציאה של השער התחתון	$V_x$	Y
0	0	קצר ל- GND	קצר ל- GND	Low	0
0	1	קצר ל- GND	נתק	Low	0
1	0	נתק	קצר ל- GND	Low	0
1	1	נתק	נתק	High	1

הגודל  $V_x$  הוא המתח בנקודה X. מכיוון ששערי ה- AND לא מבצעים בעצמם שום פעולה לוגית, החיבור בין שתי יציאות Open Drain שווה ערך לפעולת AND. לצורת חיווט כזו שיוצרת לוגיקה קוראים Wired Logic. האיור הבא מציג את פעולת ה- AND שנוצרת בפועל.



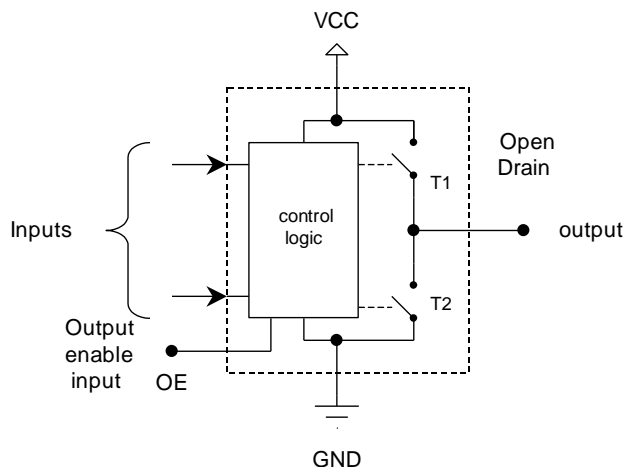
אחת הפעולות השימושיות שניתן לבצע באמצעות יציאות Open-Drain היא לממש Shared BUS. מדובר באוסף חוטים משותף שמסוגל לנתב מידע מכמה מקורות ליעד אחד או כמה יעדים. להלן דוגמה:



באיור הנ"ל כניסת האפשר OEA מנתבת את האותות S3A..S0A ליעד D3..D0. לעומת זאת כניסת האפשר OEB מנתבת את האותות S3B..S0B ליעד D3..D0. כמובן שאין לאפשר את שתי הכניסות הנ"ל בו זמנית.

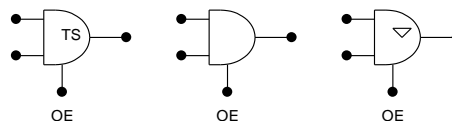
## 1.4 רכיבים בעלי יציאות מתנתקות – Tri-State

להלן המבנה של שער שהיציאה שלו היא מסוג Tri-State.

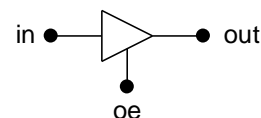


כפי שאפשר לראות, לרכיב הנ"ל יש גם כניסת אפשור יציאה שמסומנת באותיות OE (Output Enable). כאשר כניסת האפשור OE נמצאת ב- '1' לוגי היציאה של הרכיב פועלת כיציאה רגילה. כאשר כניסת האפשור OE אינה מאפשרת שני המתגים הטרנזיסטוריים T1 ו T2 מנותקים בו זמנית (כלומר היציאה של הרכיב מנותקת).

כל יציאה של כל רכיב יכולה להיות מסוג Tri-State. יציאה מסוג Tri-State יכולה להיות מסומנת באחת מהדרכים הבאות:



הרכיב שנקרא Tri-State Buffer הוא רכיב מסוג Tri-State שאינו מבצע פעולה לוגית כל שהיא.

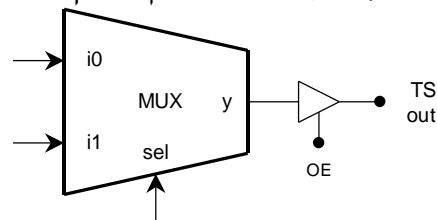


להלן טבלת אמת שמתארת את התנהגות הרכיב:

oe	in	out
0	$\emptyset$	Z
1	0	0
1	1	1

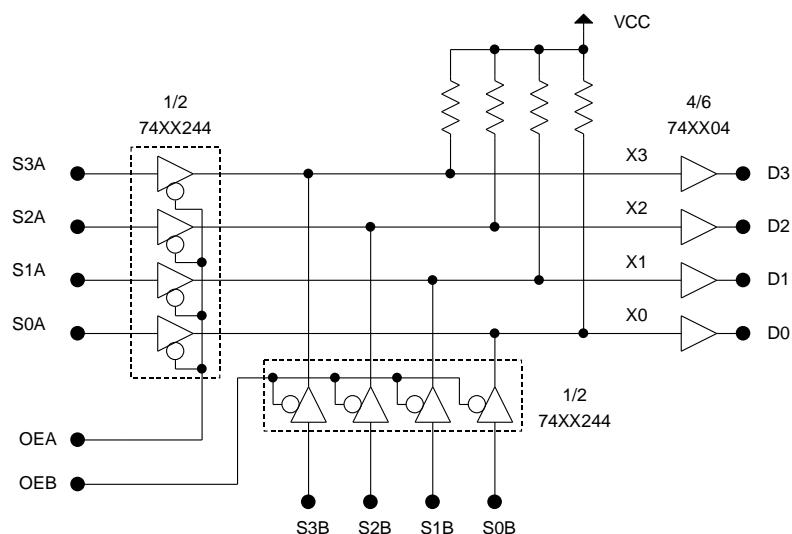
High Impedance  
Low voltage  
High voltage

באמצעות הרכיב הנ"ל ניתן להפוך כל יציאה רגילה של רכיב ליציאה מסוג Tri-State. להלן דוגמה:





**דוגמה:**



באזור הני"ל כניסת האפשר OEA מנתבת את האותות S3A..S0A ליעד D3..D0. לעומת זאת כניסת האפשר OEB מנתבת את האותות S3B..S0B ליעד D3..D0. כמובן שאין לאפשר את שתי הכניסות OE בו זמנית.

כאשר אף אחד מהמקורות אינו מאופשר, רכיבי ה – Tri-State מנותקים והמצב הלוגי נקבע באמצעות הנגדים ('1' לוגי חלש). תפקיד הנגדים הוא למנוע מצב שבו הכניסות שמוזנות מחוטי ה – BUS תהיינה מנותקות.

## 2 שיטות ייצוג של מספרים

ייצוג מספר (Number) באמצעות הספרות (Digits) שלו מבוסס על הנוסחה הבאה :

$$(a_n a_{n-1} \dots a_2 a_1 a_0 . a_{-1} a_{-2} \dots a_{-m})_b = a_n b^n + a_{n-1} b^{n-1} + \dots + a_2 b^2 + a_1 b + a_0 + a_{-1} b^{-1} + a_{-2} b^{-2} + \dots + a_{-m} b^{-m}$$

הגודל  $b$  הוא בסיס הספירה (Radix or Base). הספרות  $a_n a_{n-1} \dots a_2 a_1 a_0$  הן של החלק השלם (Integer) של המספר וספרות  $a_{-1} a_{-2} \dots a_{-m}$  הן הספרות של החלק השבור (Fraction) של המספר וההפרדה בין החלקים נעשית באמצעות נקודת השבר. בנוסחה הנ"ל דורשים גם שכל הספרות יקיימו :

$$0 \leq a_i < b$$

ניתן להראות שדרישה זו מבטיחה שלכל מספר יהיה ייצוג אחד ויחיד. הטבלה הבאה מציגה דרישה זו באופן מפורש בכמה מהבסיסים הנפוצים שבם משתמשים.

Name	Radix (base)	Existing Digits
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

ככל שספרה היא שמאלית יותר, המשקל שלה מוכפל בחזקה גדולה יותר של הבסיס (מדובר בקוד משקלי). לשיטת ייצוג כזו קוראים גם בשם Positional Number System. לספרה השמאלית ביותר  $a_n$  קוראים Most Significant Digit או בקצור MSD. לספרה הימנית ביותר  $a_{-m}$  קוראים Least Significant Digit או בקצור LSD.

כאשר מדובר בבסיס הבינארי (שהוא כמובן הבסיס החשוב יותר לחמרה ספרתית), לספרה קוראים בשם מיוחד – bit, שהוא קצור של המלים Binary digIT (או בעברית – סיבית שהוא קצור של המלים "ספרה בינארית"). לספרה השמאלית ביותר קוראים בבסיס זה – MSB (Most Significant Bit) ולספרה הימנית ביותר בבסיס זה קוראים LSB (Least Significant Bit).

הנוסחה הנ"ל יכולה לשמש אותנו כאחת מהשיטות להמרה מבסיס כל שהוא לבסיס העשרוני. להלן דוגמה :

$$(1101.1101)_2 = 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 + 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} + 0 \cdot \frac{1}{8} + 1 \cdot \frac{1}{16} = 13.8125$$

ניתן לבצע גם המרה הפוכה (מעשרוני לבינארי) בכמה שיטות.

נציג לדוגמה שיטה להמרת שלם עשרוני לשלם בינארי באמצעות חלוקות ב-2. נניח שמעוניינים לבצע את ההמרה הבאה :

$$(211)_{10} = (?)_2$$

בשיטה זו מחלקים את המספר נתון ב – 2. אם המספר הוא אי-זוגי נוצרת שארית 1 ואם המספר הוא זוגי החלוקה היא מדויקת ונוצרת שארית שהיא 0. השארית הראשונה בחלוקה היא ספרת ה – LSB. במקרה שלנו השארית הראשונה של 211 היא 1. פעולה זו מודגמת בשורה הראשונה בטבלה הבאה.

N	Remainder	
211	1	LSB
105	1	
52	0	
26	0	
13	1	
6	0	
3	1	MSB
1	1	
0	-	
		END

לאחר מכן חוזרים על התהליך כאשר מחלקים את המנה השלמה שהתקבלה מפעולת החלוקה הקודמת. במקרה שלנו ממשיכים את התהליך על המספר 105 (השורה השניה בטבלה הני"ל). באופן כזה ממשיכים את פעולות החילוק ובכל פעם מקבלים ספרה בינארית נוספת. את התהליך מסתיים ברגע שהמנה השלמה שנותרת היא 0, מכיוון שבמקרה כזה מקבלים אך ורק אפסים מובילים (Leading Zeros) שאינם משנים את ערכה של התוצאה.

נעבור לשברים. נציג לדוגמה שיטה להמרת שבר עשרוני לשבר בינארי באמצעות הכפלות ב – 2. נניח שמעוניינים לבצע את ההמרה הבאה :

$$(0.5625)_{10} = (?)_2$$

בשיטה זו מכפילים את השבר הנתון ב – 2. בעקבות פעולת הכפל יכולה להתקבל ספרה 1 או 0 של החלק השלם. החלק השלם שמתקבל מההכפלה הראשונה מהווה את ספרת ה – MSB. בדוגמה שלנו ספרת ה – MSB שמתקבלת היא 1. פעולה זו מודגמת בשורה הראשונה בטבלה הבאה.

Integer	Fraction	
	5625	MSB
1	1250	
0	2500	
0	5000	
1	0000	LSB

לאחר מכן חוזרים על התהליך כאשר מכפילים את החלק השבור שנותר מפעולת הכפל הקודמת. במקרה שלנו ממשיכים את התהליך על המספר 0.125. (השורה השניה בטבלה הני"ל). באופן כזה ממשיכים את פעולות הכפל ובכל פעם מקבלים ספרה בינארית נוספת. את התהליך אפשר להפסיק ברגע שהחלק השבור שנותר הוא 0, מכיוון שבמקרה כזה מקבלים אך ורק אפסים מזדנבים (Trailing Zeros) שאינם משנים את ערכה של התוצאה.

את ההמרה של מספרים עשרוניים לבסיסים אחרים (כמו לבסיס האוקטלי או הקסדצימלי) ניתן לבצע באופן דומה אך במקרה כזה יש לחלק (בהמרת שלמים) או לכפול (בהמרת שברים) בערכו של הבסיס (8 או 16 בהתאמה). למשל בהמרה של שלם מבסיס עשרוני להקסדצימלי ניתן להשתמש בשיטת החלוקות ב – 16. בהמרה של השבר ניתן להשתמש בשיטת ההכפלות ב – 16.

קל לעבור בין הבסיס הבינארי ובין בסיסים שהם חזקות של 2 כמו 8 ו 16. נניח שרוצים לבצע את המעבר הבא :

$$(11111.01011)_2 = (?)_8 = (?)_{16}$$

בכדי לעבור לבסיס האוקטלי יוצרים משמאל ומימין לנקודת השבר, קבוצות של שלוש סיביות. אם יש צורך בכך מוסיפים אפסים מובילים לספרות של השלם ואפסים מזדנבים לספרות של השבר, וזאת על מנת להשלים את הספרות לקבוצות בנות שלוש סיביות. כל שלוש סיביות מומרות לספרה אוטקלית אחת לפי הטבלה הבאה :

Binary Digits	Octal Digit
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

המספר שמתקבל בדוגמה שלנו הוא המספר הבא :

$$(011,111.010,110)_2 = (37.26)_8$$

בכדי לעבור לבסיס ההקסדצימלי יוצרים משמאל ומימין לנקודת השבר, קבוצות של ארבע סיביות. אם יש צורך בכך מוסיפים אפסים מובילים לספרות של השלם ואפסים מזדנבים לספרות של השבר, וזאת על מנת להשלים את הספרות לקבוצות בנות ארבע סיביות. כל ארבע סיביות מומרות לספרה הקסדצימלית אחת לפי הטבלה הבאה :

Binary Digits	Hexadecimal Digit	Decimal Value
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

המספר שמתקבל בדוגמה שלנו הוא המספר הבא :

$$(0001,1111.0101,1000)_2 = (1F.58)_{16}$$

כשם שקל לעבור בין הבסיס הבינארי והבסיס ההקסדצימלי, ניתן לעבור בקלות בין הבסיס העשרוני ובין הקוד שנקרא BCD (Binary Coded Decimal). בשיטת ייצוג זו מייצגים כל ספרה עשרונית באמצעות ארבע ספרות בינאריות. קיימים כמובן עשרה צירופים בינאריים אפשריים לכל ספרה בהתאם לטבלה הבאה:

Binary Digits	Decimal Value
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

שים לב למשל להמרה הבאה:

$$(0011,0111.1001,1000)_2 = (37.98)_{BCD}$$

לספרות הבינאריות של שיטת הייצוג BCD יש את המשקלים שמתוארים בנוסחה הבאה:

$$\begin{aligned}
 (...a_8a_7a_6a_5a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}a_{-4}...)_{BCD} = & \\
 ... + 100a_8 & \\
 + 80a_7 + 40a_6 + 20a_5 + 10a_4 & \\
 + 8a_3 + 4a_2 + 2a_1 + 1a_0 & \\
 + \frac{8}{10}a_{-1} + \frac{4}{10}a_{-2} + \frac{2}{10}a_{-3} + \frac{1}{10}a_{-4} + ... &
 \end{aligned}$$

המספר הבא אינו חוקי בשיטת הייצוג BCD מכיוון שערכה של אחת מקבוצות הספרות הבינאריות עולה על הערך 9.

$$(0011,0111.1001,1000)_{BCD} = ?$$

## 2.1 שיטות ייצוג של מספרים - בעלי סימן SIGNED

קיימות ארבע שיטות נפוצות לייצג מספרים בעלי סימן (Signed).

שיטת הסימן וגודל (Sign & Magnitude)  
שיטת המשלים ל-1 (One's Complement)  
שיטת המשלים ל-2 (Two's Complement)  
שיטת ההיסט הבינארי (Binary Offset)

הטבלה הבאה מציגה מספרים בעלי סימן בארבעת השיטות הנ"ל במערכת שמוגבלת לארבע ספרות.

Decimal	SM	1C	2C	BO
+7	0111	0111	0111	1111
+6	0110	0110	0110	1110
+5	0101	0101	0101	1101
+4	0100	0100	0100	1100
+3	0011	0011	0011	1011
+2	0010	0010	0010	1010
+1	0001	0001	0001	1001
0	0000 1000	0000 1111	0000	1000
-1	1001	1110	1111	0111
-2	1010	1101	1110	0110
-3	1011	1100	1101	0101
-4	1100	1011	1100	0100
-5	1101	1010	1011	0011
-6	1110	1001	1010	0010
-7	1111	1000	1001	0001
-8			1000	0000

בשיטת הסימן וגודל, הספרה השמאלית מייצגת את הסימן (Sign) ושאר הספרות הנמוכות יותר מייצגות את הגודל (Magnitude). ספרה שמאלית שהיא '0' מייצגת סימן חיובי וספרה שמאלית שהיא '1' מייצגת סימן שלילי. הגודל מיוצג כמספר בינארי חסר סימן. החסרונות של שיטה זו הם, שיש בה שני ייצוגים שונים עבור 0 ופחות נוח לבצע בה חישובים.

הנוסחה הבאה מציגה את המשקלים השונים של הספרות בשיטת ייצוג זו והיא מאפשרת מעבר ישיר בין שיטה זו לשיטה העשרונית:

$$(a_n a_{n-1} \dots a_2 a_1 a_0)_{SM} = (-1)^{a_n} [a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0]$$

בשיטת המשלים ל-1, המספרים החיוביים זהים בייצוג שלהם לייצוג של מספרים חיוביים בשיטת הסימן וגודל. המספרים השליליים מתקבלים באמצעות ביצוע פעולת משלים מוקטן (היפוך כל הסיביות) על המספרים החיוביים המתאימים. גם בשיטה קיימים החסרונות של: שני ייצוגים שונים עבור 0 וקושי מסוים בביצוע חישובים.

הנוסחה הבאה מציגה את המשקלים השונים של הספרות בשיטת ייצוג זו והיא מאפשרת מעבר ישיר בין שיטה זו לשיטה העשרונית:

$$(a_n a_{n-1} \dots a_2 a_1 a_0)_{1C} = -(2^n - 1) + a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0$$

בשיטת המשלים ל-2, המספרים שגדולים מ-0, זהים בייצוג שלהם לייצוג של מספרים בשיטת הסימן וגודל ושיטת המשלים ל-1. המספרים השליליים מתקבלים באמצעות ביצוע פעולת המשלים המלא (True Radix Complement), על המספרים החיוביים המתאימים. ניתן לבצע פעולה זו למשל באחת משתי השיטות הבאות.

### בצוע פעולת משלים בשיטה ראשונה:

הופכים את כל הסיביות של המספר ומוסיפים סיבית '1' בצד ימין.

### בצוע פעולת משלים בשיטה שנייה:

נעים מימין לשמאל ומעתיקים את כל הספרות, עד שנתקלים באחד הראשון (גם את ה-1 הראשון מעתיקים). לאחר מכן מעתיקים את כל הספרות במהופך (Logic Inversion).

היתרונות של שיטת המשלים ל-2 היא, שיש ייצוג בלעדי ל-0 ושנוח לבצע בשיטת ייצוג זו פעולות חשבון. זוהי הסיבה לכך שזוהי אולי השיטה הנפוצה ביותר לייצוג של מספרים בעלי סימן. שים לב שבשיטת ייצוג זו, טווח המספרים אינו סימטרי וקיים ייצוג של מספר שלילי נוסף: "100..." (כלומר '1' לוגי מוביל ושאר הספרות הן '0' לוגי). בנוסף לכך כדאי לשים לב שבכדי לעבור ממספר שלילי במשלים ל-1 למספר שלילי מתאים במשלים ל-2, יש להוסיף 1 בצד ימין של המספר.

הנוסחה הבאה מציגה את המשקלים השונים של הספרות בשיטת ייצוג זו והיא מאפשרת מעבר ישיר בין שיטה זו לשיטה העשרונית:

$$(a_n a_{n-1} \dots a_2 a_1 a_0)_{2C} = -a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0$$

שיטת ההיסט הבינארי (Binary Offset) דומה מאוד לשיטת המשלים ל-2, פרט לסיבית השמאלית ביותר שהיא הפוכה. המעבר בין מספר חיובי ושלילי נעשה בשיטת המשלים המלא (כמו בשיטת המשלים ל-2). לשיטת ההיסט הבינארי יש כמה יתרונות (פרט למעבר הקל בינה לבין שיטת המשלים ל-2). ראשית יש בשיטה זו ייצוג בלעדי ל-0 (כלומר "100..."). היתרון העיקרי של שיטה זו הוא שהמספרים מסודרת ממספרים שליליים לחיוביים בסדר בינארי עולה (כמו Unsigned). יתרון זה הופך שיטה זו לשימושית ביישומים שבהם משתמשים ברכיב שהוא חסר סימן להפעלה של מערכת עם סימן (למשל ברכיבי ADC ו DAC מסוימים). תכונה זו גם מקלה מאוד על ביצוע פעולות של השוואה (Compare) של מספרים עם סימן. תכונה זו גם הופכת את שיטת הייצוג הזו למתאימה ליישומים שבהם רוצים שהמספר השלילי ביותר יהיה סדרה של אפסים (כמו למשל במקרה של ייצוג חזקה של מספר בשיטת נקודה צפה).

הנוסחה הבאה מציגה את המשקלים שונים של הספרות בשיטת ייצוג זו והיא מאפשרת מעבר ישיר בין שיטה זו לשיטה העשרונית:

$$(a_n a_{n-1} \dots a_2 a_1 a_0)_{OB} = (a_n - 1)2^n + a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0$$

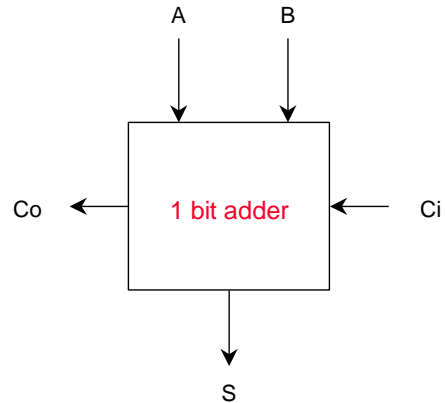
להלן הנוסחאות הנ"ל כאשר קיימת במספר גם נקודת שבר.

$$\begin{aligned} (a_n a_{n-1} \dots a_2 a_1 a_0 . a_{-1} \dots a_{-m})_{SM} &= (-1)^{a_n} [a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m}] \\ (a_n a_{n-1} \dots a_2 a_1 a_0 . a_{-1} \dots a_{-m})_{1C} &= -a_n (2^n - 2^{-m}) + a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m} \\ (a_n a_{n-1} \dots a_2 a_1 a_0 . a_{-1} \dots a_{-m})_{2C} &= -a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0 + a_{-1} 2^{-1} + \dots + a_{-m} 2^{-m} \\ (a_n a_{n-1} \dots a_2 a_1 a_0 . a_{-1} \dots a_{-m})_{OB} &= (a_n - 1)2^n + a_{n-1} 2^{n-1} + \dots + a_2 2^2 + a_1 2 + a_0 + a_{-1} + \dots + a_{-m} 2^{-m} \end{aligned}$$

### 3 רכיבים אריתמטיים

קיימות פעולות חשבוניות רבות אך הפעולה החשבונית הבסיסית ביותר שבה ניתן להיעזר בכדי לבצע פעולות חשבוניות אחרות היא פעולת החיבור. יחידת החיבור המודולרית הפשוטה ביותר מחברת בין שתי סיביות והיא נקראת Full-Adder או 1-Bit-Adder. להלן תיאור של יחידה זו:

A	B	Ci	S	Co
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



היציאה S מתארת את תוצאת החיבור (Sum) והיציאה Co מתארת את הנשא (Carry) שיחובר לדרגת החיבור הבאה. הכניסה Ci מתארת את הנשא שמגיע מדרגת החיבור הנמוכה יותר. שתי הכניסות A ו B הן הכניסות לסיביות המחוברות. ננסה לצמצם את שתי המשוואות שמתקבלות בטבלת האמת הנ"ל.

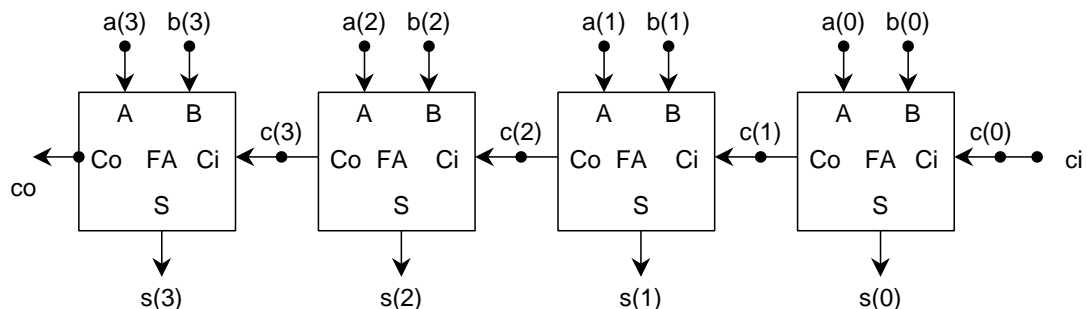
BCi	00	01	11	10
A				
0	0	1	0	1
1	1	0	1	0

$$S = A \oplus B \oplus Ci$$

BCi	00	01	11	10
A				
0	0	0	1	0
1	0	1	1	1

$$Co = A \cdot B + A \cdot Ci + B \cdot Ci$$

בכדי ליצור מחבר ברוחב כל שהוא (למשל מחבר ברוחב 4 סיביות) ניתן לשרשר יחידות דומות מסוג זה.



באיור הנ"ל האותות  $c(4), c(3), c(2), c(1), c(0)$  הם האותות הפנימיים שמעבירים את Carry - מדרגת חיבור אחת לשנייה. האותות  $s(i)$  ו  $b(i), a(i)$  הם האותות חיצוניים (כניסות ויציאות) של המחבר. בפועל מחבר ברוחב כל שהוא אינו מחובר בהיכרך באופן הנ"ל היות ובשיטה זו נוצרת הצטברות של השהיית מעבר (Propagation Delay Time) בשרשרת Carry - ה. בפועל קיימות שיטות שונות להאצת תהליך החיבור.

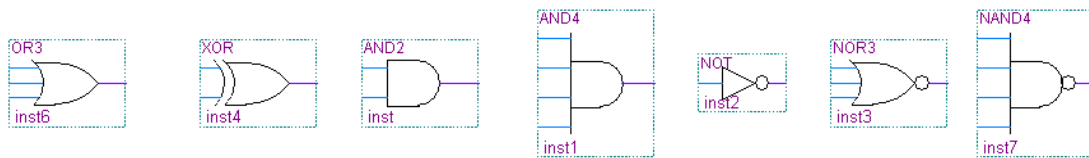


במהלך הניסוי אנו נשתמש ברכיבים אריתמטיים שונים אך בפועל, לא נבנה אותם מפעולות חשבוןיות בסיסיות כמו ה - Full-Adder. אנו נבנה את הפעולות החשבוןיות מרכיבים מוכנים גדולים יותר כמו מחברים-מחסרים, משווים, מונים וכו'.

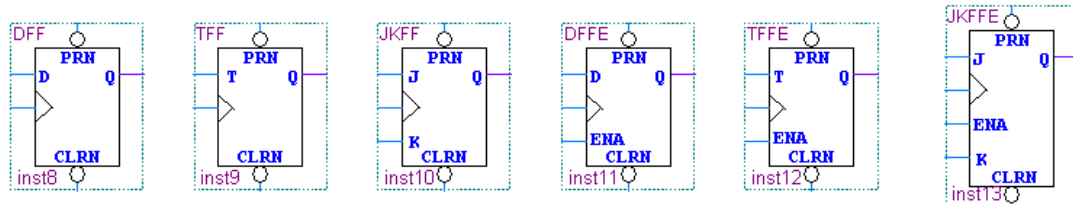
### 3.1 רכיבים אריתמטיים של Quartus

בניסוי זה נשתמש במערכת פיתוח (תכנה) של חברת Altera שנקראת בשם Quartus. כאשר מתארים את התכנ בתכנה זו ניתן לתאר אותו באופן טכסטואלי או באופן גרפי. בתי הפגישות של ניסוי זה נתאר את התכנ באופן גרפי ובניסויים שבהמשך נתאר את התכנ באופן טכסטואלי בשפת VHDL.

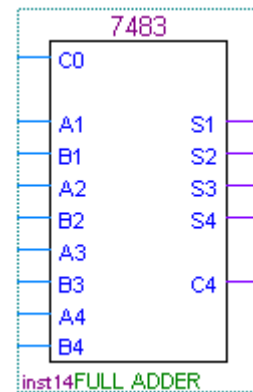
כאשר מתארים מערכת באופן גרפי עומדים לרשותנו אבני בניה מוכנים רבים של התכנה. אבני הבניה הפשוטים ביותר (שנקראים primitives) כוללים שערים. להלן כמה דוגמאות.



אבני בניה פשוטים נוספים הם למשל הפליפ-פלופים.



(על סוגי הפליפ-פלופים ואופן פעולתם נחזור בהמשך) התכנה כוללת גם אבני בניה גדולים יותר כמו מחברים, משווים מונים וכו'. להלן דוגמה למחבר ברוחב ארבע סיביות.



המחבר שמתואר באיור הנ"ל הוא מחבר קשוח שהרוחב שלו ואוסף הפעולות וההדקים שלו אינם ניתנים לשינוי. תכנה ה - Quartus תומכת גם ברכיבים שהם גמישים שבהם ניתן לשנות למש את רוחב המחבר ואוסף הפעולות וההדקים שלו.

מערכת הפיתוח Quartus תומכת ברכיבים אריתמטיים סטנדרטיים שנקראים רכיבי LPMs (השם LPM בא מהקיצור Library Parameterized Modules). קיימים כיום 29 רכיבי LPM וחמישה מתוכם יכולים להיחשב כרכיבים אריתמטיים. רכיבי LPM הם רכיבים כלליים וגמישים מאוד, שבהם המשתמש קובע את תכונות הרכיב באמצעות בחירת סוג ההדקים של הרכיב ובאמצעות בחירת פרמטרים.

## 3.2 רכיבים גנריים

כאשר משתמשים למשל ברכיב שנקרא LPM\_ADD\_SUB אפשר לקבוע תכונות כמו למשל:

- רוחב היחידה
- האם היחידה תחבר או תחסר או שתהיה מסוגלת לבצע את שתי הפעולות
- האם פעולת החיבור תיעשה במחזור שעות אחד או בכמה
- ועוד..

התקן של LPM נוצר על ידי ארגון שנקרא EDIF. השם EDIF הוא קיצור של המלים: Electronic Design Interchange Format. ארגון EDIF עסק ביצירת תקנים, שמאפשרים יצירת קשר בין כלי תכנה שמיועדים לתכנון מערכות אלקטרוניות. החל משנת 1993 התקן LPM של EDIF עבר לארגון התקינה - EIA (Electronic Industries Alliance). תקן EIA שעוסק ברכיבי LPM נקרא:

### EIA IS 103 EDIF Library of Parameterized Modules (LPM) Version 2.0

רכיבי LPM נתמכים על ידי כלי אמפלמנטציה של יצרני רכיבים רבים כמו: Cypress, Altera, Actel ו QuackLogic ועל ידי יצרני כלים צד שלישי רבים כמו: Synopsys, Mentor Graphics, Cadence, Synplicity, ViewLogic.

כל החברות הנ"ל עשויות להרחיב את אוסף הרכיבים הגמישים מעבר לרכיבים הסטנדרטיים שקיימים בתקן. לדוגמה חברת Altera קוראת לאוסף הרכיבים הגמישים שלה בשם Mega-Functions והם כוללים הן את רוב רכיבי ה - LPM הסטנדרטיים וכן רכיבים גמישים רבים נוספים. הטבלה הבאה מתארת לדוגמה את אוסף של רכיבי LPM אריתמטיים סטנדרטיים שנתמכים על ידי חברת Altera.

Name	Description
LPM_COUNTER	Counter *
LPM_ADD_SUB	Adder/ Subtractor
LPM_COMPARE	Comparator
LPM_MULT	Multiplier
LPM_ABS	Absolute value
LPM_DIVIDE	Divider

\* על רכיבי Counter נדבר גם בהמשך

חברת Altera תומכת גם ברכיבים אריתמטיים נוספים כמו ברכיב ALTSQRT, שמחשב פעולת שורש או ברכיבים ALTFP\_ADD\_SUB, ALTFP\_DIV ו ALTFP\_MUL, שהם כולם רכיבים אריתמטיים שמבצעים חישובים חשבוניים בנקודה צפה (Floating Point Arithmetic).

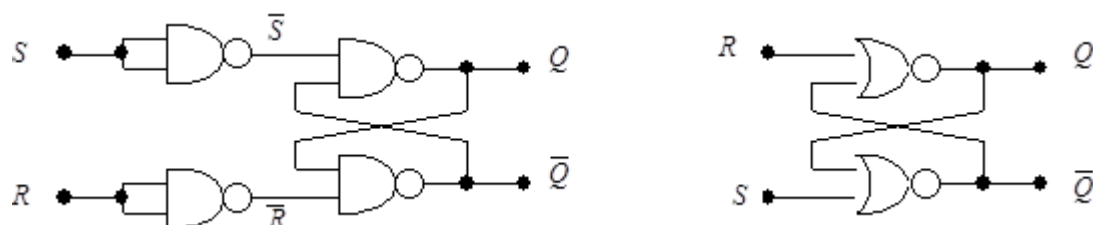
רכיבי LPM הסטנדרטיים הם גם נוחים מאוד לשימוש, בהשוואה למשל לרכיב 74 הסטנדרטיים והקשוחים. כאשר משתמשים ברכיבים גמישים אין צורך להכיר למשל סוגים רבים של מונים (74160, 74161, 74162, 74168, 74169, ...) , כפי שהדבר קורה כאשר משתמשים באבני בניה ממשפחת רכיבי ה - 74. בנוסף לך, אם רוצים למשל ליצור מונה גדול, אין צורך לחוות מספר רב של מונים קטנים ביחד (כפי שהדבר נדרש בחיבור רכיבי 74 הקשוחים). בכדי לתאר מונה גדול, פשוט בוחרים פרמטר שנקרא LPM\_WIDTH ברוחב הרצוי.

אחד היתרונות החשובים ביותר של שימוש ברכיבי ה - LPM הוא, שהשימוש בהם מאפשר לנצל משאבי חמרה פנימיים שגורמים למימוש להיות מאוד יעיל (מבחינת השטח והמהירות). מדובר בעיקר ברכיבים האריתמטיים וברכיבי הזיכרון.

כאשר משתמשים ברכיבי LPM אריתמטיים יעשה למשל שימוש במשאבים פנימיים מיוחדים בתוך הרכיב המתוכנת (על נושא זה נדון בהמשך). מדובר ברכיבים כמו Carry-Chain או Cascade-Chain או בחמרה מיוחדת שמיועדת לעזור לפעולות הכפל. במקרה של שימוש ברכיב LPM הניצול האופטימלי של משאבי הסיליקון הפנימיים ברכיב המתוכנת יהיה ודאי ולא יהיה תלוי בסגנון הכתיבה הטכסטואלי (VHDL) הספציפי שבו השתמשנו או באינטליגנציה של כלי הסינתזה שלנו, כפי שהדבר קורה בכתיבה בסגנון התנהגותי. גם על נושא זה נדון יותר בהרחבה בהמשך.

### 3.3 התקני זיכרון בסיסיים - Latch

Latch – (או נועל בעברית) הוא התקן זיכרון בסיסי שלעיתים מכונה בספרות שעוסקת באלקטרוניקה גם בשם Bi-Stable Multi-Vibrator (או בשם העברי דו-יציב). הנועל יכול להיות גם אבן יסוד של התקני זיכרון אחרים כמו ה – Gated Latch ו Flip-Flop. רכיב Latch יכולים להיות ממומשים באופנים שונים. להלן שתי דוגמאות למימוש שנעשו עם רכיבי NOR ו NAND.



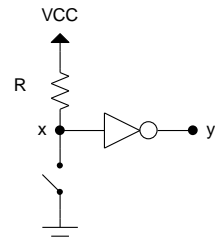
טבלת המצבים הבאה מתארת את התנהגות ה – Latch.

S	R	Behavior of Q
0	0	N.C.
0	1	0
1	0	1
1	1	--

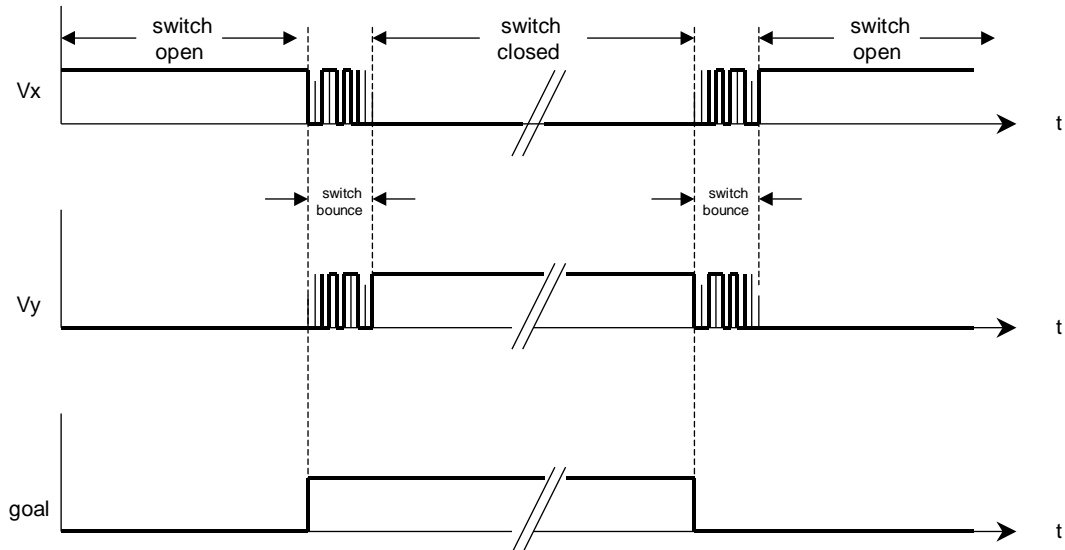
אין שינוי - זיכרון  
טעינת 0  
טעינת 1  
פעולה אסורה ולא שימושית

הפעולה האחרונה ( $S=1$  ו  $R=1$ ) גורמת ליציאות  $Q$  ו  $\bar{Q}$  שלא להיות הפוכות אחת יחסית לשניה והמצב הזה בשתי היציאות שונה במימוש עם שערי NOR ובמימוש עם שערי NAND. הפעולה האחרונה בטבלה נקראת פעולה אסורה מכיוון שאסור להשתמש בה, כל עוד רוצים להמשיך לתת ליציאות שמות הפוכים  $Q$  ו  $\bar{Q}$ . הפעולה גם אינה שימושית, מכיוון שאם נעבור מפעולה זו חזרה לפעולת שמירת מצב ( $S=0$  ו  $R=0$ ), המצב שיזכור ה – Latch אינו ברור והוא יהיה תלוי במהירויות של השערים שאינם ידועים לנו ואינם בשליטתנו. כאשר מבצעים מעבר כזה בסימולציה של מודל של שערים שיש להם זמן השהייה זהה (מה שבדרך כלל לעולם לא יקרה בעולם המעשי) היציאות של המערכת יכנסו למצב של נדנודים מתמשכים בתדר גבוה.

אחד השימושים ברכיבי Latch, הוא למנוע הפרעה חשמלית מיוחדת שנובעת מריטוט מכני של מתגים (Switch Bounce) בזמן סגירה ובזמן פתיחה של מתג מכני המחובר בשיטה הרגילה של Pull-Up Resistor וגם בצורות האחרות.

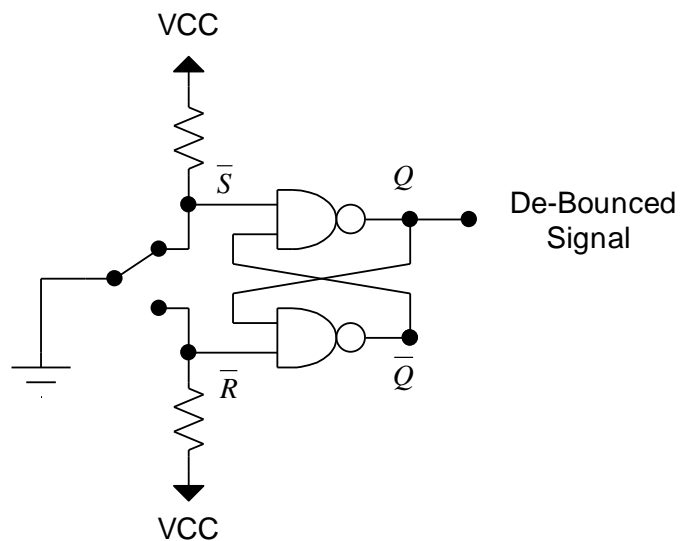


האיור הבא מדגים את המתח שנוצר במערכת הנ"ל בנקודה x ובנקודה y בזמן הסגירה והפתיחה של המתג.

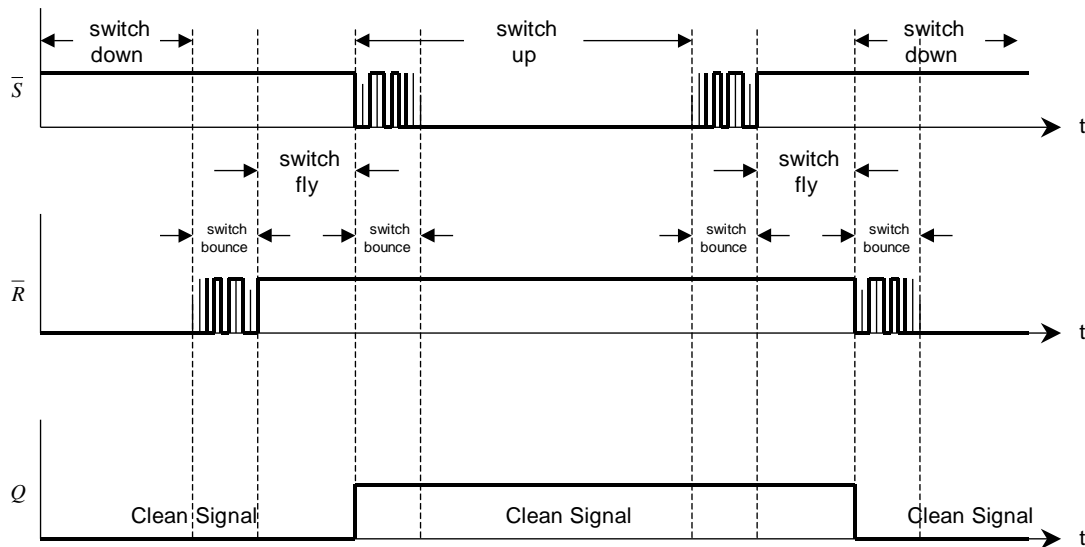


זמן הריטוט האופייני למתג בדרך כלל קטן מ- 10 ms. בשימושים מסוימים של מתגים הרעש החשמלי שנובע מתופעת הריטוט אינו מפריע, אולם קיימים שימושים שבהם רעש זה הוא בלתי נסבל ויש לבטלו. במלים אחרות, המטרה שלנו היא "לנקות את הרעשים הללו" ולקבל את המתח שמתואר בחלק התחתון של דיאגרמת הזמנים (באות בדיאגרמה שנקרא goal). "ניקוי הרעשים" שנובעים מריטוט מגעים נקרא בשם De-Bounce.

קיימות שיטות אחדות שבהן ניתן להתגבר על התופעה. בלוח DE2 קיימת מערכת למניעת ריטוטים שונה שאינה מבוססת על Latch ובה משתמשים בקבל, נגד ו-Schmitt Trigger. האיור הבא מציג את השיטה שעושה שימוש ב-Latch (בעל כניסות הפוכות).



כפי שאפשר לראות באיור הנ"ל, יש צורך להשתמש בשיטה זו במתג מסוג SPDT, שהוא מתג בעל מגע אחד משותף (Common) ושני מגעים נוספים. המתג הזה יכול להיות מבחינה מכנית באחד משלושה מצבים אפשריים. אפשרות אחת היא נגיעה של המגע השותף במגע העליון (כמו באיור). במקרה זה  $\bar{S}=0$  ו  $\bar{R}=1$  ומבצעת טעינת '1' ב-Latch, כלומר  $Q=1$ . אפשרות אחרת היא נגיעה של המגע המשותף במגע התחתון. במקרה זה  $\bar{S}=1$  ו  $\bar{R}=0$  ומבצעת טעינת '0' ב-Latch, כלומר  $Q=0$ . אפשרות נוספת היא שהמגע המשותף מרחף באוויר בין שני המגעים. במקרה זה  $\bar{S}=1$  ו  $\bar{R}=1$  ומתבצעת שמירת מצב ב-Latch. דיאגרמת הזמנים הבאה מדגימה את תנועת המתג כולל הריטוטים.



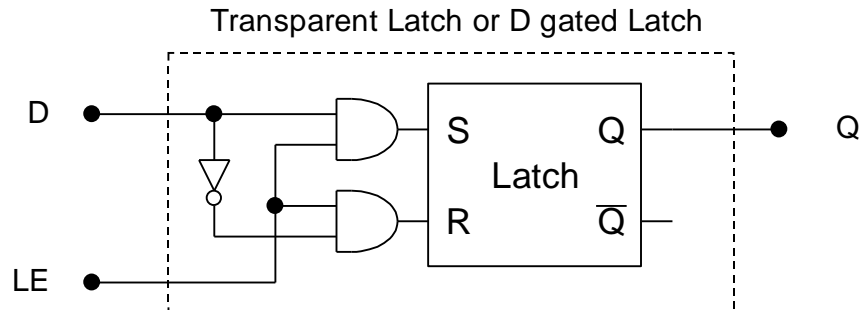
נעקוב עקוב אחרי דיאגרמת הזמנים הנ"ל. בחלקו השמאלי של האיור, המתג נמצא במצב התחתון שלו (הפוך ממה שמוצג באיור של המתג). בהרמת המתג כלפי מעלה המגע השותף מפסיק לגעת במגע התחתון. בשלב זה עלול להיווצר ריטוט מגעים כתוצאה מחיכוך המתכות שבמתג. פעולה זו מתוארת באיור הנ"ל באמצעות הגרף האמצעי של  $\bar{R}$ . הריטוט אינו גורם לשינוי כל שהוא ביציאה  $Q$ , מכיוון שהפעולות שנעשות לסירוגים ב-Latch הן: שמירת מצב וטעינת '0' (שממילא כבר טעון ברכיב). לאחר מכן מתבצע שלב של ריחוף, שבו המגע המשותף אינו נוגע באף אחד מהמגעים האחרים. בשלב זה הפעולה שמתבצעת ב-Latch היא שמירת מצב. גם בשלב זה היציאה  $Q$  אינה משתנה. בגמר פעולת הריחוף, המגע המשותף מתנגש עם המגע העליון. פעולה זו מתוארת באיור הנ"ל בגרף העליון של  $\bar{S}$ . בשלב זה נעשית ב-Latch פעולה של טעינת '1' והיציאה  $Q$  משתנית ל-'1' לוגי. כל הריטוטים שמתרחשים מיד לאחר המגע הראשון, אינם משפיעים על היציאה  $Q$ , היות והפעולות שנעשות לסירוגים ב-Latch הן: שמירת מצב וטעינת '1' (שממילא כבר טעון ברכיב). בצדו הימני של האיור, מתואר רצף הפעולות שמתרחש בהורדה של המתג כלפי מטה. גם בהורדה מתקבל אות נקי ביציאה  $Q$ .

בסופו של דבר בטלנו את כל הרעשים שנוצרו מריטוט המתג או במלים אחרות בצענו Switch-De-Bounce. ההתגברות על הריטוט בשיטה זו, אינה תלויה בזמן הריטוט של המתג, כלומר השיטה הנ"ל תצליח גם אם נשתמש במתגים לא כל כך מוצלחים, שזמן הריטוט שלהם מאוד ארוך.

הדרישות היחידות שצריכות להתקיים הם שהמתג יהיה מתג SPDT, שבו המגע המשותף לעולם לא יכול לגעת בשני המגעים האחרים בו זמנית (Break Before Make) ושהמתג אינו מסוגל לרטט ממגע אחד למגע השני. אלו הן דרישות שרוב המתגים עומדים בהם בקלות.

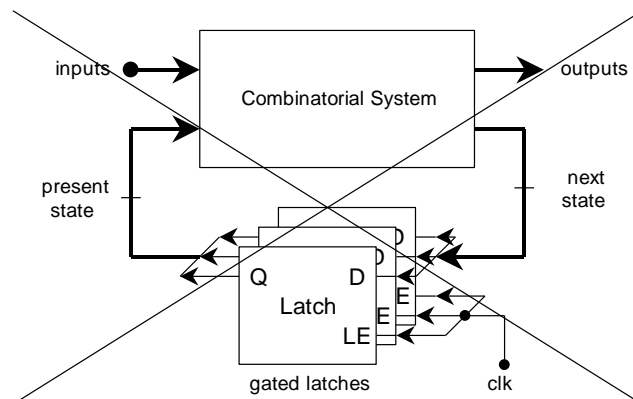
### 3.4 התקני זיכרון בסיסיים - Gated Latch

התקן זיכרון שימושי נוסף הוא ה-Gated Latch. זהו רכיב שבמרכזו נמצא Latch והטעינה שלו מאפשרת באמצעות כניסה שנקראת Gate או Latch Enable. נסמן כניסה זו באותיות LE (קיצור של השמות Latch Enable). האיור הבא מציג מימוש אפשרי של D-Gated Latch.



כאשר כניסת האפשרור מופעלת ('1' לוגי), המידע מהכניסה D עובר ל-Latch וליציאה Q. המידע שמגיע ליציאה Q זהה למידע שמוכנס לכניסה D, כאילו שה-Latch "שקוף" למידע שעובר דרכו. מסיבה זו לעתים קרובות קוראים לרכיב הנ"ל גם בשם Transparent Latch. כאשר כניסת האפשרור אינה מופעלת, המידע מהכניסה D נחסם על ידי שערי ה-AND וה-Latch שומר על המצב האחרון שבו הוא נמצא לפני שפעולת כניסת האפשרור הופסקה.

להבדיל מפליפ-פלופים, רכיבי Latch אינם מסוגלים להיות רכיבי זיכרון של מערכת סינכרונית (מכונת מצבים סינכרונית).



אם במערכת שתורכב באופן כזה, ננסה להפעיל את כניסות האפשרור (LE), ייווצר משוב (FeedBack) צירופי (מכונה מצבים אסינכרוניים). זהו כמובן מקור לצרות כמו נדודים מרוצים ומערכות ספרתיות שהאמינות שלהן תלויה בתזמוני הרכיבים. יש להימנע משימוש במערכות באופן כזה ויש כמובן להשתמש ברכיבי Gated Latch ליישומים אחרים.

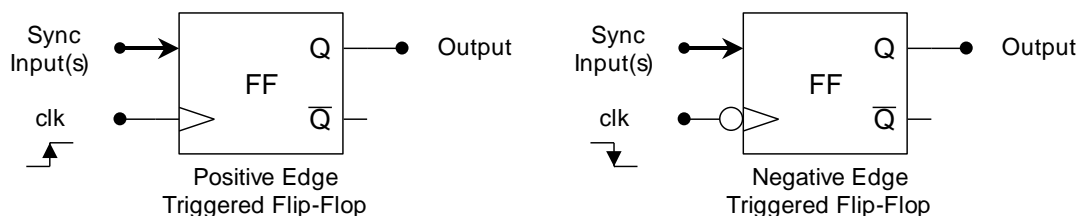
במערכת הפיתוח Quartus ניתן לקרוא לרכיבים מסוג Gated-Latch בודדים והם שייכם למשפחת הרכיבים הפשוטים שנקראים Primitives. קיימים שני רכיבים מסוג זה:

רכיב D-Gated Latch שנקרא DLATCH  
רכיב D-Gated Latch שיש לו גם איפוס א-סינכרוני שנקרא DLATCH

לשני הרכיבים הנ"ל יש יציאה Q ואין להם יציאה הפוכה.

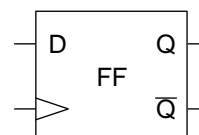
### 3.5 התקני זיכרון בסיסיים - Flip-Flop

בחלק הקודם ראינו שרכיבי Gated Latch אינם יכולים לשמש כרכיבי זיכרון של מכוונת מצבים סינכרונית. בדיוק לתפקיד הזה נועדו הפליפ-פלופים. בניגוד לפליפ-פלופים שפעלו בעבר בשיטת ה - Master-Slave הפליפ-פלופ המודרני הוא מדורבן קצה (Edge Triggered) והוא רגיש לעליה באות השעון שלו או לירידה באות השעון שלו. כניסת השעון מסומנת באופן הבא :



בצד שמאל של האיור הנ"ל מוצג פליפ-פלופ שרגיש לעליה ובצד ימין מוצג פליפ-פלופ שרגיש לירידה. מכאן ואילך נתייחס לפליפ-פלופים שרגישים לעליה בלבד. בזמן עליית אות השעון הפליפ-פלופ מבצע את מה "שאומרות" לן הכניסות הסינכרוניות שלו.

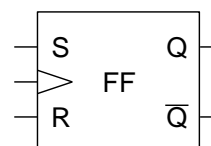
קיימים פליפ-פלופים בעלי כניסות סינכרוניות מסוגים שונים. הפליפ-פלופ הנפוץ ביותר ברכיבים מיתכנתים מודרניים הוא פליפ-פלופ עם כניסה סינכרונית מסוג D.



טבלת המצבים הבאה מתארת את ההתנהגות הפשוטה של פליפ-פלופ זה.

CLK	D	Behavior	
	0	$Q \leq 0$	טעינת 0
	1	$Q \leq 1$	טעינת 1
	else	$Q \leq Q$	שמירת מצב - No Change

פליפ-פלופ מסוג אחר הוא פליפ-פלופ עם כניסות מסוג SR.

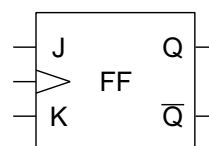


הטבלה הבאה מתארת התנהגות של פליפ-פלופ בעל כניסות סינכרוניות SR.

CLK	S	R	Behavior	
	0	0	$Q \leq Q$	שמירת מצב - No Change
	0	1	$Q \leq 0$	טעינת 0
	1	0	$Q \leq 1$	טעינת 1
	1	1	--	פעולה אסורה
	else		$Q \leq Q$	שמירת מצב - No Change

הפעולה  $S=1$  ו  $R=1$  היא פעולה אסורה ובלתי שימושית מסיבות דומות לאלה שראינו ברכיב ה - Latch.

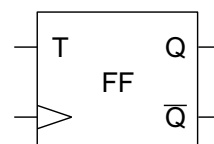
פליפ-פלופ מסוג אחר הוא פליפ-פלופ עם כניסות מסוג JK.



פליפ-פלופ זה דומה לפליפ-פלופ SR אך אין בו פעולה אסורה ובמקומה נעשית פעולת הפיכת מצב של הפליפ-פלופ, כפי שאפשר לראות בטבלה הבאה שמתארת התנהגות של פליפ-פלופ בעל כניסות סינכרוניות JK.

CL K	J	K	Behavior	
	0	0	$Q \leq Q$	שמירת מצב - No Change
	0	1	$Q \leq 0$	טעינת 0
	1	0	$Q \leq 1$	טעינת 1
	1	1	$Q \leq \text{not } Q$	הפיכת מצב - Change
	else		$Q \leq Q$	שמירת מצב - No Change

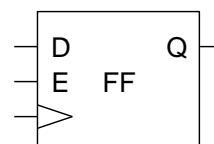
סוג נוסף של פליפ-פלופ שיש לו כניסה סינכרונית אחת הוא פליפ-פלופ עם כניסה סינכרונית T.



טבלת המצבים הבאה מתארת את ההתנהגות של פליפ-פלופ זה.

CLK	T	Behavior	
	0	$Q \leq Q$	שמירת מצב - No Change
	1	$Q \leq \text{not } Q$	הפיכת מצב - Change
	else	$Q \leq Q$	שמירת מצב - No Change

לכל אחד מסוגי הפליפ-פלופים הנ"ל יכולה להיות גם כניסת אפשר (Enable) סינכרונית. נציג לדוגמה פליפ-פלופ מסוג D שיש לו כניסת אפשר.

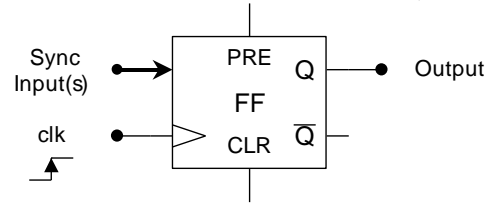


ההתנהגות שלו מתוארת בטבלת המצבים הבאה.

CLK	E	D	Behavior	
	0		$Q \leq Q$	לא מאפשר - Disabled
	1	0	$Q \leq 0$	טעינת 0
	1	1	$Q \leq 1$	טעינת 1
	else		$Q \leq Q$	שמירת מצב - No Change



לפליפ-פלופים יש גם כניסות א-סינכרוניות. כניסות אלו מתוארות באיור הבא בחלק העליון ובחלק התחתון של הפליפ-פלופ.



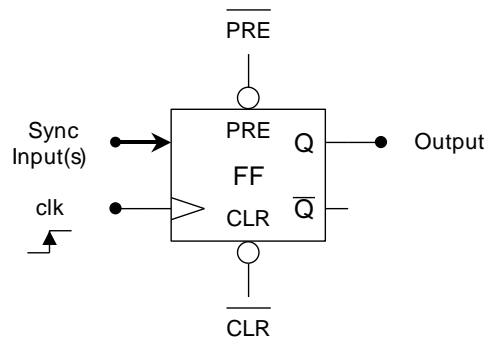
השמות PRE ו CLR הם קיצורים לשמות PRESET ו CLEAR בהתאמה. הטבלה הבאה מתארת את ההתנהגות של פליפ-פלופ בהשפעה של כניסות אלו.

PRE	CLR	Behavior
0	0	Sync Behavior
0	1	$Q \leq 0$
1	0	$Q \leq 1$
1	1	--

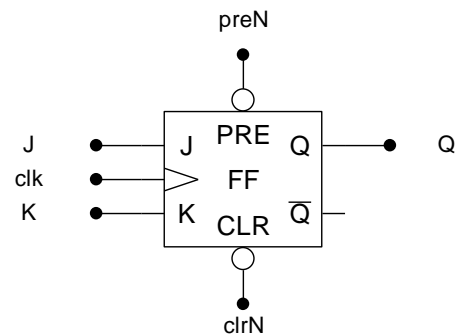
פעולה סינכרונית רגילה  
טעינה א-סינכרונית של 0  
טעינה א-סינכרונית של 1  
פעולה אסורה

הכניסות PRESET ו CLEAR הן כניסות ישירות שהן חזקות יותר מהכניסות הסינכרוניות. רק כאשר כניסות אלו אינן פעילות הפליפ-פלופ מתנהג באופן סינכרוני (בהתאם לתיאורים שהופיעו קודם). כפי שאפשר לראות הכניסה PRESET טוענת '1' לוגי באופן א-סינכרוני והכניסה CLEAR טוענת '0' לוגי באופן א-סינכרוני. ההתנהגות בהשפעת כניסות אלו דומה להתנהגות של Latch. הפעולה האחרונה  $PRE=1$  ו  $CLR=1$  אסורה בדיוק מאותן הסיבות שפעולה זו הייתה אסורה ב-Latch.





במקרים רבים (וכפי שהדבר נראה באיור הבא) כניסות אלו פעילות בנמוך (Active Low).



לעתים קרובות בכתיבה טכסטואלית (למשל בשפת תיאור חמרה) מסמנים אותות שהם Active-Low באמצעות אות מסיימת שהיא N כלומר במקרה הנ"ל אותות אלו יסומנו ב  $\text{preN}$  ו  $\text{clrN}$ . באיור הבא מתואר באופן כזה פליפ-פלופ מסוג JK שיש לו שתי כניסות אסינכרוניות שהן פעילות בנמוך.

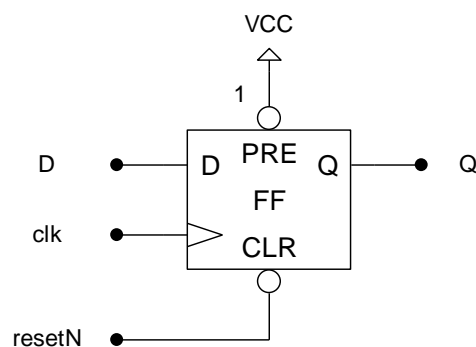


הטבלה הבאה מתארת את הפליפ-פלופ JK הנ"ל (שיש לו כניסות אסינכרוניות שפעילות בנמוך).

pre N	clr N	CL K	J	K	Behavior	
0	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	--	פעולה אסורה
1	0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$Q \leq 0$ (Async)	טעינה א-סינכרונית של 0
0	1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	$Q \leq 1$ (Async)	טעינה א-סינכרונית של 1
1	1		0	0	$Q \leq Q$	שמירת מצב - No Change
1	1		0	1	$Q \leq 0$	טעינת 0
1	1		1	0	$Q \leq 1$	טעינת 1
1	1		1	1	$Q \leq \text{not } Q$	הפיכת מצב - Change
1	1		else		$Q \leq Q$	שמירת מצב - No Change

רצוי שלא להשתמש בכניסות א-סינכרוניות במהלך העבודה הרגיל של המערכת. בדרך כלל משתמשים בכניסות אלו רק בכדי לאתחל את פעולת הפליפ-פלופ מיד לאחר הזנת המתח למערכת.

בנוסף לכך דרך כלל משתמשים רק באחת מהכניסות (או ב - Preset או ב - Clear) אך ולא בשתייהן. הכניסה שאינה בשימוש תחובר למצב לוגי קבוע שלא ישפיע על הכניסות. האיור הבא מציג פליפ-פלופ מסוג D שבו משתמשים בכניסה CLEAR אך לא משתמשים ב - PRESET שפעיל בנמוך.



בחלק זה לא עסקנו בנושא החשוב של נתוני תזמון של פליפ-פלופים. בנושא זה נעסוק באחד מהניסויים שבהמשך.

במערכת הפיתוח Quartus ניתן לקרוא לרכיבי פליפ-פלופים בודדים שהם חלק ממשפחת הרכיבים הפשוטים שנקראים Primitives. להלן רשימה של רכיבים אפשריים.

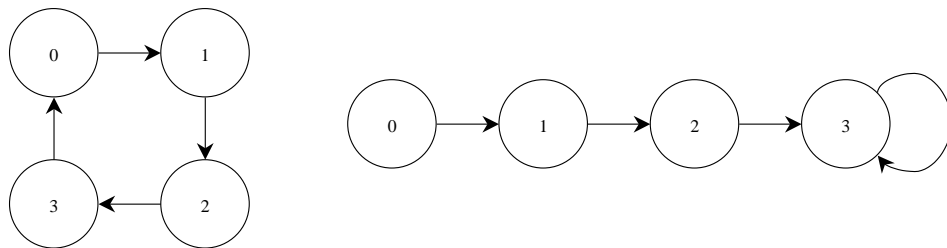
- פליפ-פלופ D נקרא - DFF
- פליפ-פלופ T נקרא - TFF
- פליפ-פלופ SR נקרא - SRFF
- פליפ-פלופ JK נקרא - JKFF
- פליפ-פלופ D עם כניסת אפשרור סינכרונית נקרא - DFF
- פליפ-פלופ T עם כניסת אפשרור סינכרונית נקרא - TFF
- פליפ-פלופ SR עם כניסת אפשרור סינכרונית נקרא - SRFF
- פליפ-פלופ JK עם כניסת אפשרור סינכרונית נקרא - JKFF

כלל הרכיבים הנ"ל יש כניסות אסינכרוניות שפעילות בנמוך ושנקראות clrN ו prnN. לפליפ-פלופים יש יציאה Q ואין להם יציאה הפוכה.

## 4 מונים

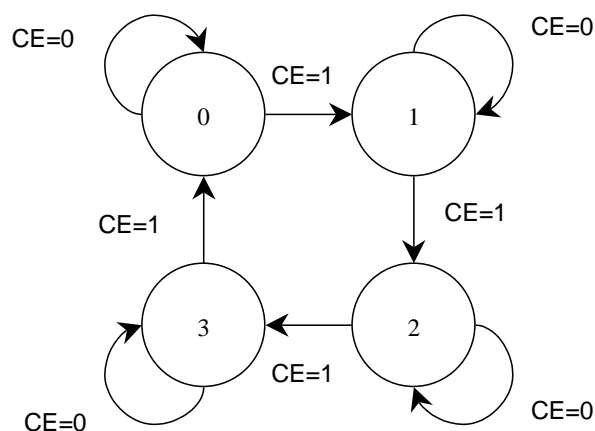
מונים סינכרוניים ורגיסטרים הם המקרים הפרטיים הנפוצים ביותר למכונת מצבים.

המונה הסינכרוני הבסיסי ביותר היא מכונת מצבים סינכרונית שעוברת ממצב למצב כתגובה לעלויות (או ירידות) באות השעון, וזאת לפי סדר מצבים קבוע. כל מצב של המונה מייצג מספר מסוים של פולסי שעון שהמונה ספר. המעברים ממצב למצב במונה הבסיסי הפשוט ביותר אינן תלויות בכניסות (מונה אוטונומי). באיור הבא מוצגות שתי דוגמאות של מונים שמסוגלים לספור עד ארבעה פולסי שעון.



המונה הימני הוא מונה בעל ארבעה מצבים ולאחר ארבעה פולסי שעון הוא נתקע במצב D (Dead End Counter). גם למונה השמאלי יש ארבעה מצבים אך הוא מחזורי (Cyclic) וכעבור ארבעה פולסי שעון, הוא חוזר על מחזור הספירה שלו. המונה המחזורי הוא נפוץ יותר.

המונים שהוצגו עד לשלב זה הם מונים בסיסיים שהם אוטונומיים, כלומר חסרי כניסות (פרט לאות השעון). למונים עשויות להיות גם כניסות שמסוגלות להשפיע על אופן הספירה של המונה. לדוגמה להלן מונה שיש לו כניסת אפשרור ספירה (Count Enable) שמסומנת באותיות CE.



להלן כמה סוגים נפוצים של כניסות של מונים :

כניסת איפוס א-סינכרוני - CLRN (האות N מציינת שהכניסה פעילה בנמוך)












כניסת איפוס סינכרוני - SCLR

כניסת טעינה סינכרונית במקביל (Parallel Load) - PL או PE

כניסת אפשרור ספירה (Count Enable) סינכרונית - CE

כניסת קביעת כיוון הספירה סינכרונית - UP.

בטבלה הבאה מתוארת דוגמה להתנהגות של מונה שיש לו ארבע מתוך חמשת הכניסות הנ"ל.

CLRN	CLK	PL	CE	UP	Behavior
0					count <= 0 (async)
1		1			count <= din
1		0	1	0	count <= count - 1
1		0	1	1	count <= count + 1
1		0	0		count <= count
				else	count <= count

שים לב שבדוגמה זו כניסת הטעינה במקביל היא הכניסה הסינכרונית החזקה ביותר וכניסת האיפוס הא-סינכרוני היא כניסה שחזקה יותר מכל הכניסות הסינכרוניות.

מונים מסוגלים לספור בקודים שונים כמו קוד בינארי, קוד Gray וקודים אחרים. מחזור הספירה המכסימלי של מונה שיש לו FF פליפ-פלופים (שנקרא גם יחס החלוקה המכסימלי של המונה) הוא  $N = 2^{FF}$ . לדוגמה יחס החלוקה המכסימלי של מונה בעל ארבעה פליפ-פלופים הוא 16. יחס החלוקה של מונה יכול להיות גם נמוך יותר מיחס החלוקה הנ"ל. לדוגמה יחס החלוקה של מונה בעל ארבעה פליפ-פלופים יכול להיות למשל 10. אם משרשרים כמה מונים מסוג זה (בעלי יחס חלוקה 10) ביחד, מקבלים בעצם מונה שסופר בקוד BCD. זוהי הסיבה לכך שלמונה בינארי בעל יחס חלוקה 10 קוראים בדרך כלל בשם מונה BCD. לעומתו למונה בינארי בעל ארבעה פליפ-פלופים שיחס החלוקה שלו 16 קוראים בדרך כלל בשם 4-Binary Counter.

פרט ליציאות של הפליפ-פלופים של המונה שמציגות את ספירת המונה, למונים יש לעתים קרובות גם יציאה נוספת שנקראת TC. זוהי יציאת ה-Terminal Count של המונה שמפיקה '1' לוגי כאשר המונה מאופשר ונמצא בספירה האחרונה שלו. יציאה זו משמשת בדרך כלל לצורכי שרשור לכניסת אפשר-ספירה של מונה גבוה יותר בחיבור בין מונים. בהמשך נציג את המערכת הצירופית שמייצרת יציאה זו.

לעתים קרובות למונה עשוית להיות שתי כניסות אפשר :

כניסת אפשר מקבילי (Count Enable Parallel) – CEP

כניסת אפשר מטפף (Count Enable Tricke) – CET

בכדי שהמונה יהיה מאופשר, שתי הכניסות צריכות להיות מאופשרות בו זמנית :

$$enable = CEP \bullet CET$$

ההבדל בין שתי כניסות האפשר הנ"ל, הוא שהכניסה CET משפיעה גם על היציאה TC. להלן דוגמאות להשפעה כזו במונה 4 סיביות בינארי ובמונה BCD.

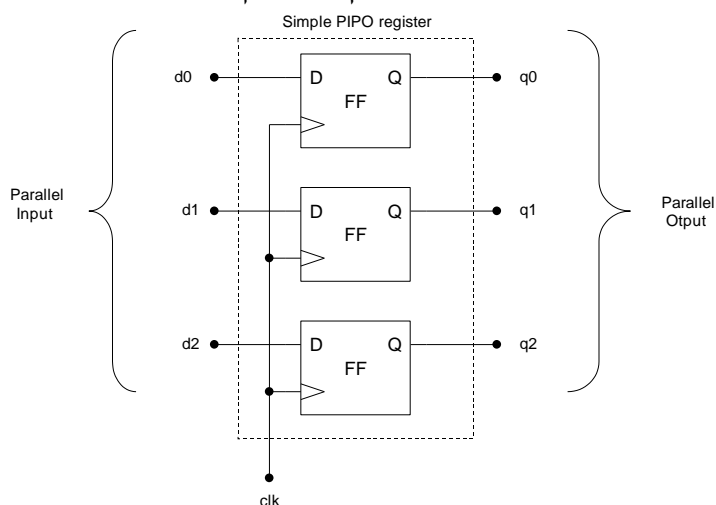
$$TC_{4BIT\_BINARY} = Q_3 \bullet Q_2 \bullet Q_1 \bullet Q_0 \bullet CET$$

$$TC_{4BIT\_BCD} = Q_3 \bullet \overline{Q_2} \bullet \overline{Q_1} \bullet Q_0 \bullet CET$$

בהמשך נראה שכאשר רוצים להשתמש ברכיבי מונים ב-Quartus ניתן אמנם להשתמש ברכיבים סטנדרטיים קשיחים כמו: 74160, 74161, 74162, 74168 ו-74169, אך בדרך כלל נוח יותר להשתמש ברכיב LPM גמיש בשם LPM\_COUNTER.

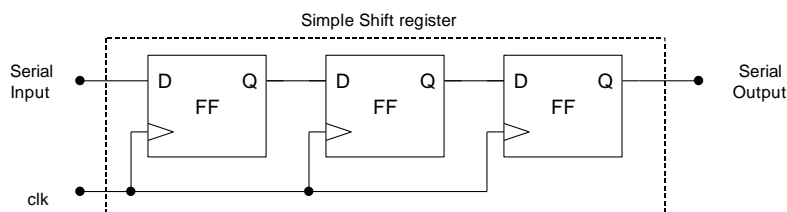
## 4.1 רגיסטרים (אוגרים)

גם רגיסטרים הם מקרים פרטיים נפוצים למכונת מצבים. הרגיסטר הפשוט ביותר האפשרי הוא רגיסטר בעל כניסות ויציאות במקביל. להלן דוגמה למבנה של רגיסטר כזה ברוחב שלוש סיביות.

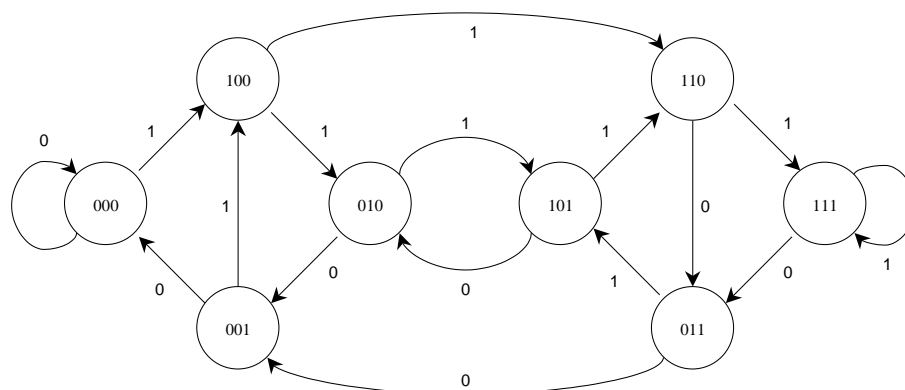


כניסות נפוצות נוספות של רגיסטרים מסוג זה הן כניסת אפשרור סינכרונית (Sync Enable) וכניסת איפוס א-סינכרונית.

סוג אחר של רגיסטר הוא רגיסטר ההזזה. להלן דוגמה לרגיסטר הזה ברוחב שלוש סיביות שמזיז את המידע שמוזן לכניסה הטורית (SI או Serial Input) ימינה לכיוון היציאה הטורית (SO או Serial Output).



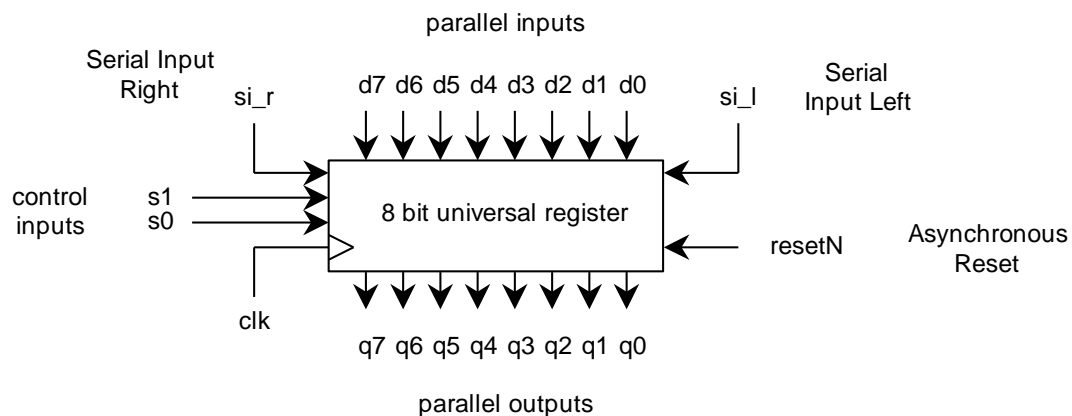
האיור הבא מציג דיאגרמת מצבים (בועות) של הרגיסטר הנ"ל (המעברים תלויים במצב הכניסה הטורית):



גם לרגיסטר מסוג זה (ההזזה) יכולה להיות כניסת אפשרור סינכרונית (Sync Enable) וכניסת איפוס א-סינכרונית.

רגיסטר גמיש יותר הוא רגיסטר אוניברסלי ברוחב שמונה סיביות שמסוגל לבצע פעולות שונות. להלן דוגמה של רגיסטר כזה.

resetN	clk	s1	s0	behavior
0	X	X	X	Asynchronous reset
1	⌊	0	0	No change (memory)
1	⌊	0	1	Shift Right
1	⌊	1	0	Shift Left
1	⌊	1	1	Parallel Load



הטבלה הנ"ל מתארת את ההתנהגות של הרגיסטר הנ"ל. שים לב שהכניסה resetN היא כניסה אסינכרונית שפעילה בנמוך. הפעולה הסינכרונית של הרגיסטר נקבעת על ידי כניסות הבקרה s0 ו s1. הרגיסטר מסוגל לטעון במקביל, להזיז ימינה ולהזיז שמאלה.

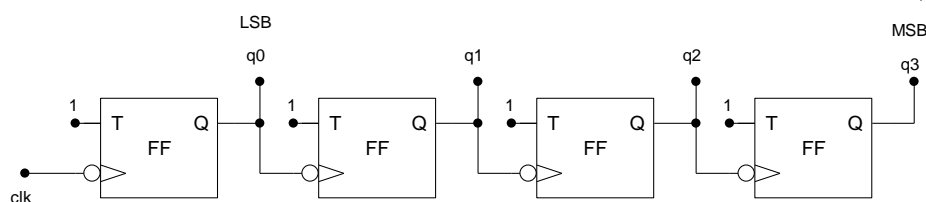
ניתן לבנות כל רגיסטר שהוא כולל רגיסטרים אוניברסליים, באמצעות שני סוגי רכיבים פנימיים:

פליפ-פלופים מסוג D  
ובוררים

כאשר רוצים להשתמש ברגיסטר אוניברסלי ב - Quartus ניתן אמנם להשתמש ברכיב הסטנדרטי הקשור 74194, אך בדרך כלל נוח יותר להשתמש ברכיב LPM גמיש בשם LPM\_SHIFTREG. ברגיסטרים פשוטים יותר ניתן גם להשתמש ברכיבים הגמישים LPM\_DFF או LPM\_FF.

## 4.2 מונים אסינכרוניים

מונים א-סינכרוניים (Ripple Counters) הפשוטים ביותר (בעלי יחס חלוקה שהוא חזקה שלמה של 2) הם מונים בעלי המבנה הבא :



מונים אסינכרוניים שימושיים במיוחד ביישומים של חלוקת תדר (Frequency Scaler). ביישומים אלו מכניסים בכניסת השעון תדר גבוה ובאחת מהיציאות של הפליפ-פלופים מפיקים אות יציאה שהוא בתדר נמוך יותר (פי 2 או פי 4 או פי 8 .. נמוך יותר). ביישומים אלו יש למונים א-סינכרוניים יתרונות (בהשוואה למונים סינכרוניים), היות והתדר המכסימלי של המונה ביישומים מסוג זה הוא תדר השעון המכסימלי של הפליפ-פלופ הראשון ומדובר בדרך כלל בתדר גבוה.

$$f_{MAX\_COUNTER} = f_{MAX\_FIRST\_FLIP-FLOP}$$

היישום הנפוץ יותר של מונים, הוא להיות סופרי מאורעות (Event Counter). ביישומים מסוג זה רוצים לספור את מספר הירידות או העליות שמתרחשים באות השעון. בניגוד ליישומי חלוקת תדר, ביישומים של סופרי מאורעות, אנו נדרשים "לצלם" ברגע מסוים את צירוף היציאה של המונה. תדר השעון המכסימלי במקרה זה הוא :

$$f_{MAX\_COUNTER} = \frac{1}{FFs \cdot t_{CO}}$$

הגודל  $t_{CO}$  הוא זמן השהייה של כל פליפ-פלופ. הגודל  $FFs$  הוא מספר הפליפ-פלופים.

היות ומספר הפליפ-פלופים מופיע במכנה של הביטוי הנ"ל, תדר השעון המכסימלי של המונה ה-אסינכרוני ביישומים הנפוצים מסוג זה (ספירת מאורעות) הוא אינו גבוה (בהשוואה למונה סינכרוני) וזאת כאשר מספר הפליפ-פלופים הוא גדול (גדול יותר למשל מ-3 או 4).

חסרון נוסף של מונים אסינכרוניים הוא הקושי לממש במונים אסינכרוניים מגוון כניסות בקרה סינכרוניות כמו : כניסות אפשר, כניסת שינוי כיוון, כניסת טעינה במקביל, כניסת איפוס סינכרוני וכו'.

סוג המונה שהוצג קודם הוא מונה בעל יחס חלוקה של 2 בלבד. בכדי ליצור מונים בעלי יחסי חלוקה שהם אינם חזקות שלמות של 2, יש להשתמש בטכניקות כמו : מונה גילוי מצב וקפיצה ל-0 או מונה חסימה. כאשר ממשים מונים בטכניקות אלו, הם נהפכים לאיטיים עוד יותר. היות והתכן של מונים אלו הוא אינו סינכרוני, יש להיעזר בטכניקות שונות על מנת למנוע מרוצים. לא נרחיב כאן את הדיבור על בעיות אלו.

בעבר, כאשר מימשו מונים באמצעות רכיבים שאינם מיתכנתים, נהגו להשתמש במונים אסינכרוניים מהסוג הנ"ל, וזאת כאשר רצו ליצור מונה ממספר קטן יחסית של רכיבים ובמיוחד ביישומים איטיים או ביישומים של מחלקי תדר.

כיום יצרני רכיבים מיתכנתים מודרניים ממליצים שלא להשתמש במונים אסינכרוניים, מכיוון שהשימוש בהם אינו חוסך כלל במשאבים של הרכיב. ברכיבים מיתכנתים מודרניים ניתן לממש מונים סינכרוניים מהירים, כאשר כל ביט של המונה גוזל בסך הכל Macrocell אחד או Logic Element אחד בלבד !

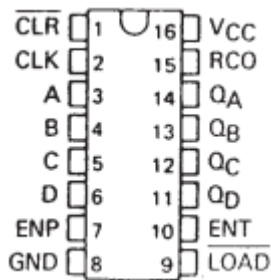
אם תממש מונה אסינכרוני באמצעות Quartus ותפעיל את כלי הדיאגנוזה הפנימי שנקרא Design Assistant, תקבל דיווח שדומה לדיווח הבא:

Warning: (Medium) Design should not contain ripple clock structures.

Warning: (Medium) Clock signal source should drive only input clock ports.

### 4.3 מונה סינכרוני: BCD - 74160 , BINARY - 74161

להלן הסבר קצר על יחידת המונה מסוג 74160. יש לו 10 מצבים והוא סופר ציקלית מ-0 עד 9. רכיב 74161 הוא מונה דומה אבל יש לו 16 מצבים 0 – 15.



הערה: הקו מעל חלק מהשמות של אותות הכניסה והיציאה אומר שאותות אלו פעילים בנמוך.

למונה יש כניסת איפוס א-סינכרונית שנקראת CLR שפעילה בנמוך. כלומר, כאשר כניסה זו ב-0 גם ללא עלית אות השעון CLK, ארבעת יציאות המונה (QA, QB, QC, QD), מתאפסות.

הכניסה הסינכרונית LOAD היא כניסת טעינה במקביל (Parallel Load) שפעילה בנמוך. כאשר כניסה זו ב-0 ויש עליה באות השעון CLK, המספר המוזן לכניסות (A, B, C, D), נטען למונה ומופיע ביציאות שלו. D היא סיבית ה-MSB ו-A היא סיבית ה-LSB.

למונה יש שתי כניסות אפשר ספירה (Count Enable) שפעילות בגבוה שנקראות ENP ו-ENT (Enable Parallel ו-Enable Trickle). המונה מאפשר לספירה, רק כאשר שתי כניסות האפשר מוזנות בו זמנית ב-'1' לוגי.

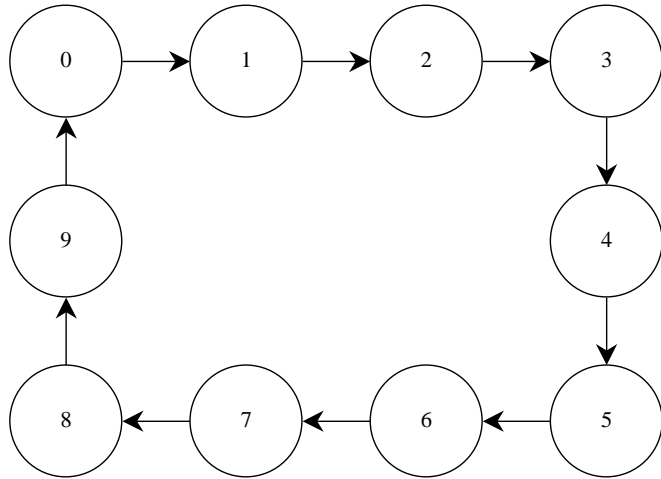
פעולת הטעינה הסינכרונית LOAD, חזקה יותר מפעולת הספירה (Count).

הטבלה הבאה מתארת את התנהגות המונה.

CLR N	CLK	LDN	ENP	ENT	Behavior
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	count <= 0000 (async)
1		0	<input type="checkbox"/>	<input type="checkbox"/>	count <= (D,C,B,A)
1		1	1	1	count <= (count + 1) mod 10
				else	count <= count (No Change)

סיביות היציאה של המונה הן (QA, QB, QC, QD), כאשר QD היא סיבית ה-MSB ו-QA היא הסיבית ה-LSB. המונה סופר עד למצב 9 ומשם חוזר למצב 0 (מונה מודולו 10).





היציאה RCO היא יציאת ה - Terminal Count של המונה, שפעילה כאשר המונה נמצא במצב האחרון "1001" והיא מושפעת גם מכניסת האפשרות ENT באופן הבא :

$$RCO = Q_D \cdot \overline{Q_C} \cdot \overline{Q_B} \cdot Q_A \cdot ENT$$

ההבדל היחיד בין ENT ו ENP הוא ש - ENP לא משפיע על RCO.

להסבר מפורט יותר, ראו בדפי הנתונים של הרכיב במודל. בעמוד 10 מתוארת דיאגרמת זמנים מפורטת של התנהגות הרכיב.

#### **כיצד אפשר לקבל מונה שעובר רק על חלק מהמצבים?**

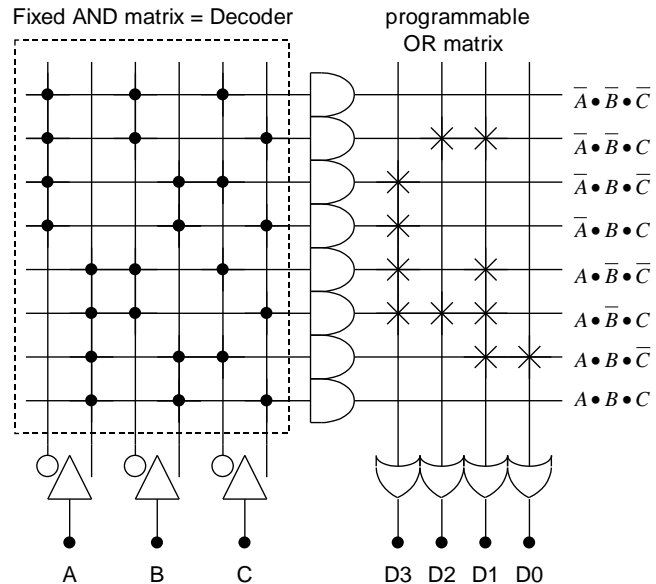
יש לזהות את המצב האחרון שעד אליו רוצים שהמונה יספור, לאפשר איתו את כניסת LOAD, ולחבר לכניסות D – A את המצב שאליו המונה יחזור וממנו ימשיך לספור בכל מחזור חדש.

## 5 ארכיטקטורות של רכיבים מיתכנתים –

### Simple Programmable Logic Devices - 5.1

ברכיב ה- ROM (Read Only Memory) קיימות שתי מטריצות חיבורים.

ROM 8 X 3



מטריצת החיבורים השמאלית (AND-Matrix), היא מטריצה בעלת חיבורים קבועים. חיבורים קבועים מסומנים באיור הנ"ל באמצעות הסימן הבא :



מטריצת החיבורים הקבועה שוות ערך ל- Decoder שמייצר את כל המכפלות המלאות האפשריות. המימוש שבו משתמשים ברכיב ROM הוא מימוש SOP קנוני. המטריצה הימנית (OR-Matrix) היא מטריצה מתוכנתת. חיבורים מיתכנתים מסומנים באיור הנ"ל באמצעות הסימן הבא :



המטריצה הנ"ל מתוכנתת בהתאם לטכנולוגית התכנות של רכיב ה- ROM (EPROM, PROM), EEPROM (וכ...), ה- ROM המקורי היה Mask-Programmable כלומר התכנות שלו נקבע במפעל שייצר אותו. רכיבי PROM (Programmable ROM), ניתנים לתכנות חד פעמי. הרכיבים הראשונים שהיו ניתנים לתכנות, כללו נתיך (Fuses) בכל הצטלבות במטריצה המתוכנתת והוא היה ניתן להתכה (אך לא לשחזור). רכיבים מודרניים שניתנים לתכנות חד פעמי נקראים בשם OTP- ROM (One Time Programmable-ROM). בכל הרכיבים הללו התכנות של הרכיב היה כרוך בדרך כלל בהכנסת הרכיב למכשיר תכנות מיוחד (Programmer). רכיבי EPROM הם רכיבים שניתנים לתכנות חשמלי והמחיקה שלהם נעשית באמצעות הקרנה של הרכיב באור אולטרא סגול (Ultra-Violet). לרכיבים אלו יש חלון קוורץ שקוף שמאפשר הקרנה של המוליך למחצה הפנימי ברכיב. החלון מכוסה במדבקה לאחר התכנות של הרכיב. בטכנולוגית התכנות הבאה השתמשו ברכיבים שניתנים לתכנות ומחיקה חשמליים - EEPROM (Electrically Erasable Programmable ROM). רכיבים אלו בדרך לא הוצאו מהמעגל שבו הם פעלו, כלומר הם מתוכנתים ונמחקים במעגל עצמו (In System Programming), כלומר אין צורך להכניס אותם למכשיר התכנות. הרכיבים המודרניים ביותר שניתנים לתכנות ומחיקה חשמליים הם רכיבי ה- Flash. אלו הם רכיבים זולים יותר מרכיבי EEPROM וניתן להגיע בהם לממדים גדולים.

השם הפופולרי לפעולת התכנות של הרכיב היא "צריבה". ההתפתחות הטכנולוגית של שיטת התכנות אינה בלעדית לרכיבי ROM, אלא מאפיינת את כל משפחות הרכיבים המיתכנות כמו PAL, PLA, CPLD. שיטת התכנות המודרנית מבוססת כיום בדרך כלל על רכיבי Flash שניתנים לתכנות במעגל עצמו.

שים לב שכל השערים בסכימה הנ"ל מצוירים באופן מיוחד, כאשר רואים רק קו אחד שמזין אותם. במציאות שערים אלו הם שערים בעלי כמה כניסות או שהלוגיקה של השערים נוצרת באמצעות Wired Logic. כאשר אחת היציאות אינה מחוברת לאף אחת מהמכפלות היא מפיקה '0' לוגי. אפשר לדמיין שבסוף כל קו אנכי במטריצה הימנית, מצוי נגד Pull-Down, שגורם להזנת כניסת השער ב - '0' לוגי חלש. ב - ROM חדש (או מחוק) שבו כל המכפלות מחוברות ליציאה, מתקבל '1' לוגי, היות והסכום של כל המכפלות המלאות האפשריות הוא תמיד '1'.

שים לב שרכיב ה - ROM שבאיור מימש בעצם את טבלת האמת הבאה :

Address			Data			
A	B	C	D3	D2	D1	D0
0	0	0	0	0	0	0
0	0	1	0	1	1	0
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	1	0	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	1	0	0	0	0

שים לב לקשר בין מיקום "האחדים" ומיקום החיבורים בצד הימני של הטבלה והאיור.

האורך של ה - ROM הוא מספר  $Length = 2^{inputs}$ . הרוחב של ה - ROM שווה למספר היציאות של ה - ROM. לעתים קרובות מסמנים את שני הממדים של ה - ROM באופן הבא :

$$size = length \times width = 2^{inputs} \times outputs$$

בדוגמה שבאיור הנ"ל, הממדים של ה - ROM הם מאוד קטנים :

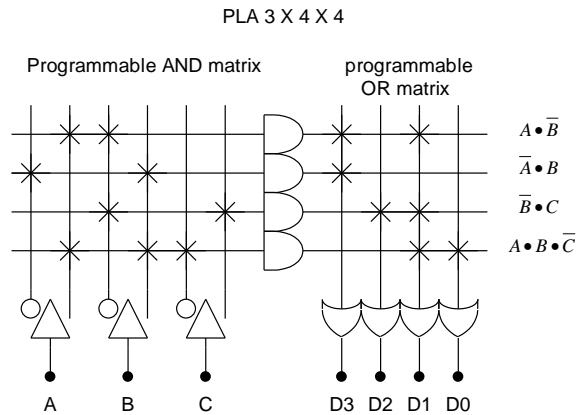
$$8 \times 4$$

ממדים של רכיב ROM מעשיים הם גדולים בהרבה. למשל ברכיבים שנקראים 27512/28512/29512 (שהם רכיבי EPROM, EEPROM ו FLASH בהתאמה בעלי אותם הממדים) מדובר בממדים הבאים :

$$512 = 64K \times 8$$

רכיבי ROM מצטיינים באוניברסליות מוחלטת, אך הם מאוד בזבזניים במשאבים (מכפלות) כאשר מספר הכניסות (Fan-In) הוא גדול.

ברכיבי PLA מוותרים על מימוש של כל המכפלות המלאות האפשריות (מימוש קנוני) ומסתפקים במספר הרבה יותר קטן. המכפלות שממומשות, נקבעות על ידי המשתמש, כלומר המטריצה השמאלית היא מתוכנתת (בנוסף למטריצה הימנית).



הממדים רכיבי PLA מסומנים בדרך כלל באמצעות שלושה מספרים.

$$inputs \times ands \times outputs$$

כאשר הגודל *ands* מסמן את מספר רכיבי ה- AND והוא בדרך כלל מקיים:

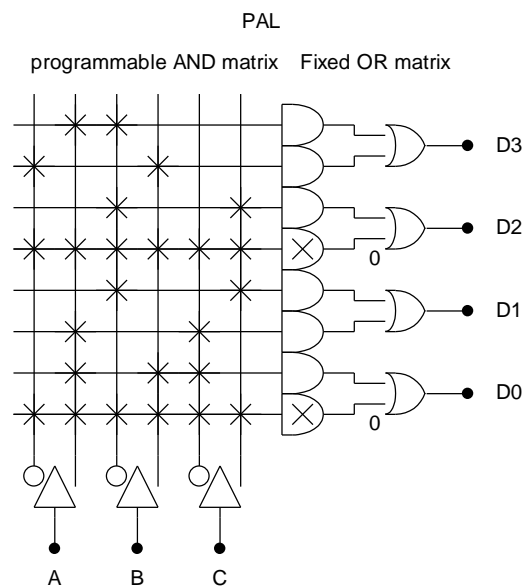
$$ands \ll 2^{inputs}$$

בדוגמה הקטנה שלנו הממדים של ה- PLA הם:

$$3 \times 4 \times 4$$

שים לב שהפונקציות שמומשו ברכיב ה- PLA וברכיב ה- ROM שבאיור הן פונקציות זהות. בכדי לממש את הפונקציה באופן יעיל על PLA קטן שבו יש מאגר משותף של מכפלות, יש לבצע לפונקציות הנ"ל צמצום SOP קבוצתי (Group Reduction).

רכיבי PAL הם רכיבים שבהם מטריצת החיבורים השמאלית (AND-Matrix) היא מיתכנתת, ומטריצת החיבורים הימנית (OR-Matrix) היא קבועה.



שים לב שבאיור הנ"ל המטריצה הימנית לא צוירה כלל כמטריצה. הרעיון המרכזי שעומד מאחורי רכיבי PAL הוא, שהאות אינו צריך לעבור דרך שתי מטריצות מיתכנותות (כפי שהוא עובר ברכיבי

PLA) ולכן זמני ההשהיה נמוכים והרכיב זול. בנוסף לכך רוב הפונקציות שבהם משתמשים בתכן ספרתי הן פונקציות בעלות מספר קטן של מכפלות. באיור הנ"ל כל יציאה חוברה אך ורק לשני שערי AND וזהו מספר קצת נמוך מהמקובל ברוב רכיבי ה - PAL.

ברכיבי PAL אין בדרך כלל שיתוף בין מכפלות, כפי שהוא קיים ברכיבי PLA ולכן הצמצום שנעשה לפונקציות, הוא צמצום SOP נפרד לכל פונקציה. שים לב שקיימות ברכיב הנ"ל שתי פונקציות שיש להן רק מכפלה אחת (D0 ו D2). בכדי שפעולתו של שער ה - AND המיותר לא תפריע, מאפסים את היציאה שלו, באמצעות השארת כל החיבורים שלו (מספיק היה להשאיר שני חיבורים של כניסה אחת שמאפסים את התוצאה, מכיוון שהם תמיד הפוכים אחד לשני).

רכיבי PAL מוגבלים במספר המכפלות שלהם. המגבלה עשויה לנוע למשל סביב 8 מכפלות ברכיבי PAL קטנים. ברכיבים גדולים (CPLDs) שמתבססים על ארכיטקטורות של רכיבי PAL (שנקראות ארכיטקטורות Product Terms או בקצור ארכיטקטורות PT), כמו למשל ברכיבי MAX של Altera, מספר המכפלות המכסימלי מוגבל לחמש מכפלות. רכיבי PAL וארכיטקטורות PT מתאימות במיוחד למימוש פונקציות שיש להן הרבה כניסות (Big Fan-In), אך מספר קטן של מכפלות (Low PT-Count), כמו למשל הפונקציה הבאה:

$$Y = \overline{A}\overline{B}\overline{C}\overline{D}\overline{E}\overline{F}\overline{G}\overline{H} + \overline{A}\overline{B}\overline{I}\overline{J}\overline{K}\overline{L}\overline{M} + \overline{C}\overline{D}\overline{N}\overline{O}\overline{P}\overline{Q}\overline{R} + \overline{M}\overline{S}\overline{T}\overline{U}\overline{V}\overline{W}\overline{X} + \overline{C}\overline{Z}$$

ארכיטקטורות אלו פחות מתאימות למימוש פונקציות עם הרבה מכפלות, כמו למשל הפונקציה הבאה שיש לה יותר מחמש מכפלות.

$$Y = A \cdot B \cdot C \cdot D \cdot \overline{E} + \overline{A} \cdot \overline{B} \cdot C \cdot D \cdot E + \overline{A} \cdot B \cdot \overline{C} \cdot D \cdot E + \overline{A} \cdot B \cdot C \cdot \overline{D} \cdot E + A \cdot B \cdot \overline{C} \cdot \overline{D} \cdot E + A \cdot B \cdot C \cdot D \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{C} \cdot D \cdot \overline{E} + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D \cdot E + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \overline{E}$$

היות ובחלק מהפונקציות שאותן ברצוננו לממש, מגבלה זו עשויה להיות קשה מדי, יצרני רכיבי PAL יצרו מנגנונים שונים שמאפשרים בכל זאת לשבור את המגבלה הקיימת. נציג שתי טכניקות שהן מאוד נפוצות.

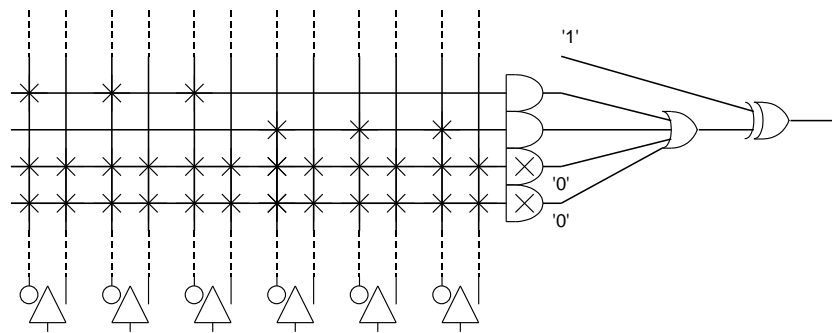
בחירת אופטימלית של קוטביות  
שימוש ב - Fold-Back

נציג את הטכניקה הראשונה. נניח שאנו רוצים לממש את הפונקציה הבאה שיש לה 9 מכפלות.

$$Y = A \cdot D + A \cdot E + A \cdot F + B \cdot D + B \cdot E + B \cdot F + C \cdot D + C \cdot E + C \cdot F$$

עם רכיב PAL יש לו רק ארבע מכפלות לכל יציאה. כמובן שהדבר אינו אפשרי.

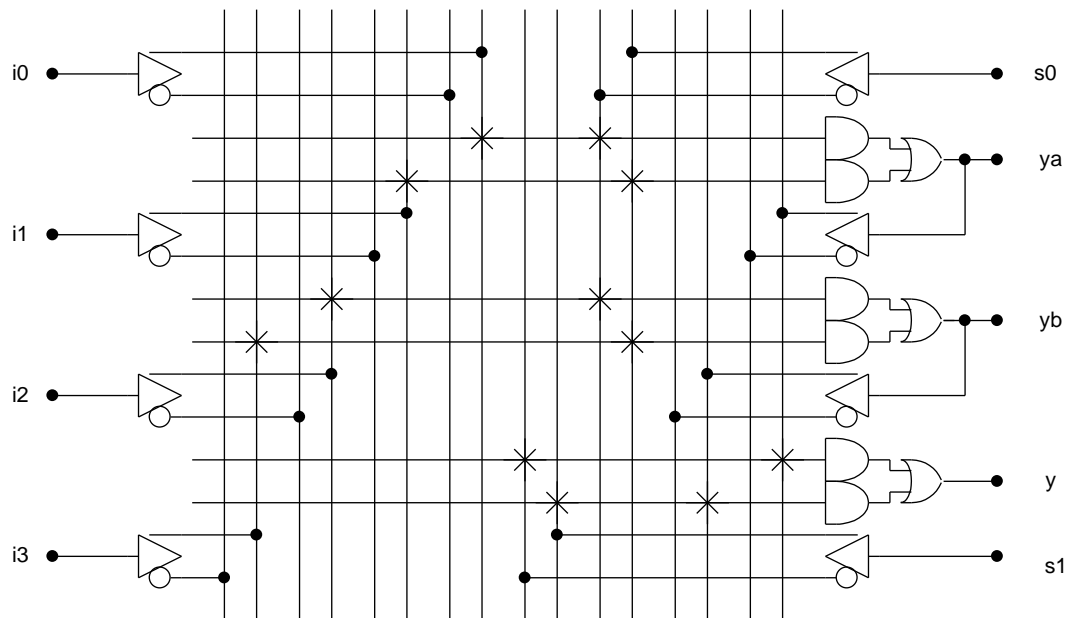
הארכיטקטורה הבאה מאפשרת בכל זאת לבצע זאת.



שים לב לשער XOR ביציאה. כאשר שער זה מוזן בחלק העליון שלו ב - '0' לוגי הוא אינו הופך את הפונקציה והוא מתאים למימוש הפונקציה בצורת  $SOP$ . כאשר שער זה מוזן בחלק העליון שלו ב - '1' לוגי הוא הופך את הפונקציה והוא מעביר את הפונקציה ליציאה בצורת  $\overline{SOP}$ . להלן מימוש הפונקציה בקוטביות ההפוכה:

$$Y = \overline{A} \bullet \overline{B} \bullet \overline{C} + \overline{D} \bullet \overline{E} \bullet \overline{F}$$

טכניקה נפוצה נוספת שבה משתמשים ברכיבי PAL, היא יצירת פונקציות ביניים שחוזרות למטריצה המתוכנתת (Fold-Back) ועוברת פעם נוספת או אפילו כמה פעמים דרך הרכיב (2 Pass, 3 Pass, ...). בכל מעבר דרך הרכיב הפונקציה אוספת מכפלות. הפעלת הטכניקה הזו מבוססת על קיפול היציאות (Fold-Back) חזרה למטריצה ה - AND-Matrix. להלן דוגמה של רכיב PAL מאוד קטן שמציג ארכיטקטורה שמאפשרת פעולה זו.



בדוגמה זו מצליחים לממש בורר  $4 \Rightarrow 1$ , באמצעות רכיב PAL שמוגבל מאוד במספר המכפלות שלו (שתי מכפלות לכל יציאה). משוואת הבורר המקורית דורשת ארבע מכפלות:

$$y = \overline{s1} \bullet \overline{s0} \bullet i0 + \overline{s1} \bullet s0 \bullet i1 + s1 \bullet \overline{s0} \bullet i2 + s1 \bullet s0 \bullet i3$$

בדוגמה זו יצרו הרחבה במבנה של עץ של שלושה בוררים בגודל  $2 \times 1$ . להלן המשוואות של שלושת הבוררים:

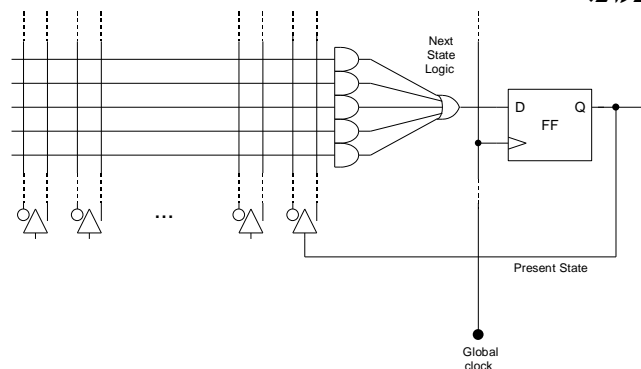
$$ya = \overline{s0} \cdot i0 + s0 \cdot i1$$

$$yb = \overline{s0} \cdot i2 + s0 \cdot i3$$

$$y = \overline{s1} \cdot ya + s1 \cdot yb$$

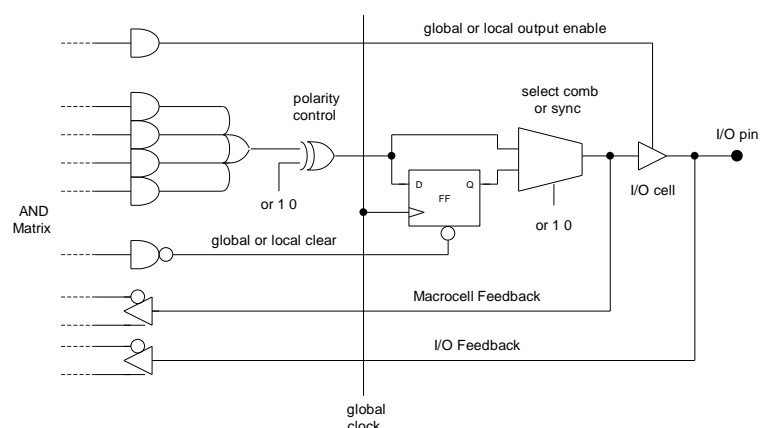
קיימות טכניקות נוספות להגדלת מספר המכפלות של ארכיטקטורות PAL או PT.

עד לשלב זה הצגנו רכיב PLA ו PAL כרכיבים צירופיים בלבד. האמת היא, שרכיבים אלו יכולים להיות גם רכיבים עם פליפ-פלופים. להלן הארכיטקטורה שמתארת ארכיטקטורת PT עם פליפ-פלופ.



היציאה של הפליפ-פלופ יכולה לשמש כ - Present State של מכונת מצבים שחוזר כמשוב למערכת הצירופית שיוצרת את העירור למצב הבא של המכונה.

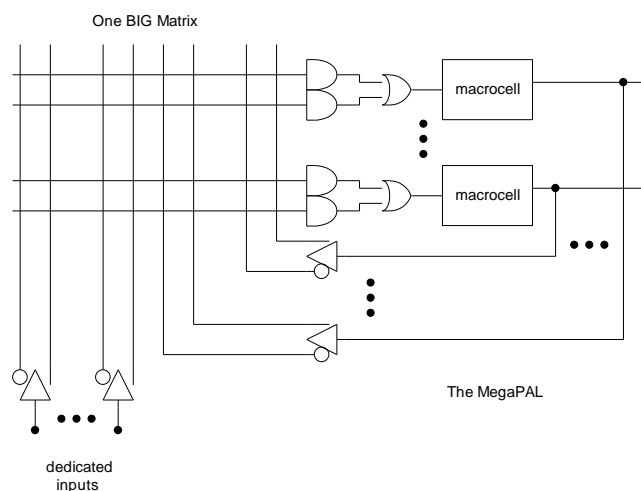
בדורות הראשונים של רכיבי PLA ו PAL (בשנות השבעים) אכן יצרו יציאות קשורות שהן צירופיות או יציאות מסונכרנות כמו היציאה בדוגמה שבאיור האחרון. ברכיבים מודרניים יותר (החל משנות השמונים) יצרו רכיבים גמישים, שאזור היציאה שנקרא ברכיבים אלו MacroCell והוא משמש כתא גמיש שיכול להתנהג באופן צירופי או באופן פליפ-פלופי, בקביעה של המשתמש. בנוסף לכך הדקי היציאה יכולים לשמש גם בהדקי כניסה ואפשר התחברות ל - BUS (הם נקראים הדקי I/O). להלן דוגמה אופיינית של Macrocell.



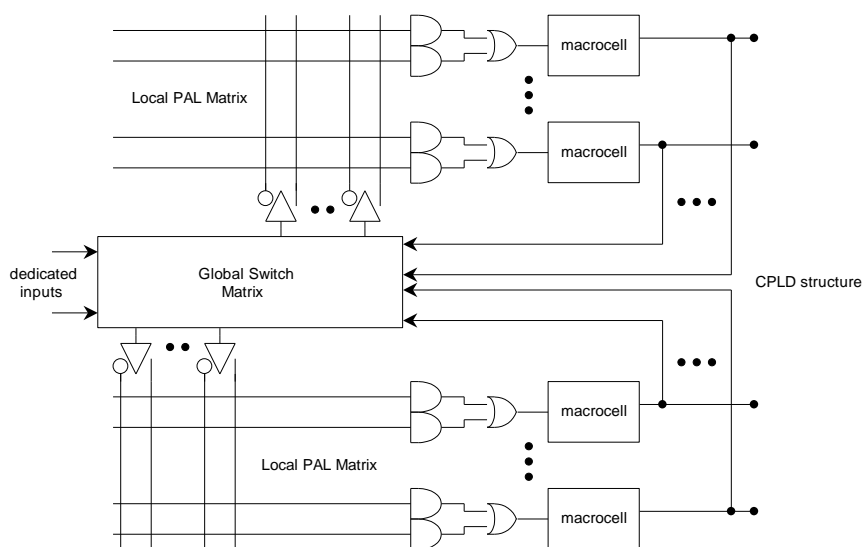
הבורר בוחר האם אות היציאה יהיה צירופי או פליפ-פלופי. שער ה - XOR בוחר קוטביות. קיימים שני קיפולים (או משובים) פנימה: אחד מהדק היציאה ואחד מתוך ה - Macrocell. היתרון של קיפול כפול הוא, שניתן להשתמש בהדק של הרכיב ככניסה מבלי שמבזבזים את המשאבים של ה - Macrocell. רכיב ה - Tri-State-Buffer מאפשר לנתק את ההדק מאות היציאה של ה - Macrocell או שמאפשר התחברות ל - BUS.

## Complex PLDs 5.2

רכיבי ה- SPLDs גדלו בנפח שלהם עד לשלב שבו המטריצה המתוכנתת (AND-Matrix) הגיע לשטח מכסימלי שהחל להאט את פעולת הרכיב.



בשלב זה (שקרה בתחילת שנות התשעים), יצרני רכיבים רבים זנחו את הרעיון ליצור Mega-PAL בעל מטריצה מתוכנתת אחת מאוד גדולה ועברו למבנה רכיב שנקרא CPLD (Complex PLD).

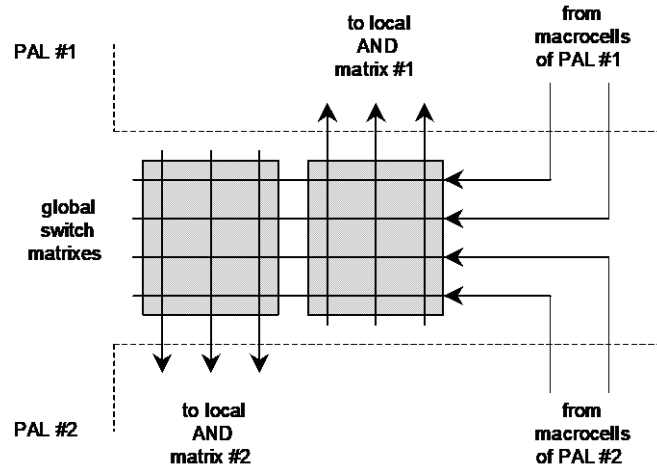


במבנה זה נעשה פיצול של המטריצה המתוכנתת הגדולה לכמה מטריצות קטנות יותר. הקישור בין החלקים השונים נעשה באמצעות מטריצה גלובלית מאוד מהירה, שזמן ההשהיה שלה זניח בהשוואה לזמני ההשהיה של המטריצות הלוקליות.

באיור הנ"ל מוצג פיצול של הרכיב לשני חלקים. מבנה כזה אפשר ליצרנים לייצר רכיב עם הרבה משאבים (Macrocells) ולא להגיע לגודל גדול מדי של המטריצה הלוקלית, שהיא מהווה את צוואר הבקבוק בקביעת מהירות הרכיב. ככל שהרכיב שאותו רצה היצרן ליצור היה גדול יותר הוא פיצל את הרכיב ליותר חלקים.

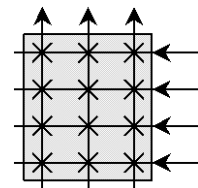


נתבונן בדוגמה מיניאטורית של המטריצה הגלובלית הבאה.

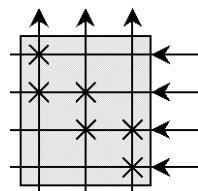


באיור הנ"ל המטריצה הגלובלית חולקה לשתי מטריצות גלובליות, שכל אחת מנתבת אותות מה - Macrocells חזרה למטריצות ה - AND של כל PAL לוקלי. כפי שאפשר לראות, לכל מטריצה חלקית כזו נכנסים יותר אותות (4) מאשר יוצאים ממנה (3). זהו המצב שקיים בכל רכיבי ה - CPLD. מבנה כזה אינו מאפשר לחבר את כל האותות שמופקים ב - Macrocells למכפלה אחת של מטריצות ה - AND של הרכיב (מה שכמובן ניתן היה לבצע ברכיבי PAL). הגמישות הנמוכה יותר בניתוב האותות, הוא כמובן מחיר שאותו מוכנים לשלם על מנת לאפשר יצירה של רכיבי CPLD גדולים.

ננסה להבין כיצד בנויה המטריצה הגלובלית. לשם כך נתבונן בחלק הימני של המטריצה הנ"ל.

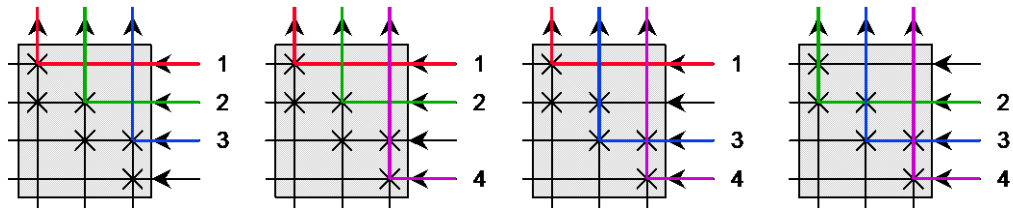


האם בכל הצטלבות חוטים במטריצה ימצא חיבור (כפי שהדבר אכן מודגם באיור הנ"ל) ? התשובה לכך ברכיבי CPLD היא, בדרך כלל שלילית. המטריצה הגלובלית בדרך כלל מאוכלסת רק באופן חלקי בחיבורים. האיור הבא מציג דוגמה :



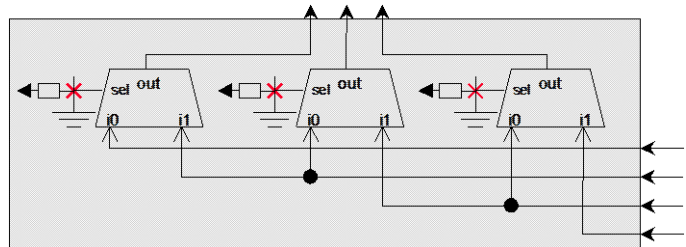
בדוגמה הנ"ל מאכלסים רק שישה חיבורים מתוך 12 האפשריים.

האיור הבא מראה שלמרות שהמטריצה אוכלסה באופן חלקי, כל ניתובי החוטים של ארבע מקורות לשלושה יעדים הם עדיין אפשריים:



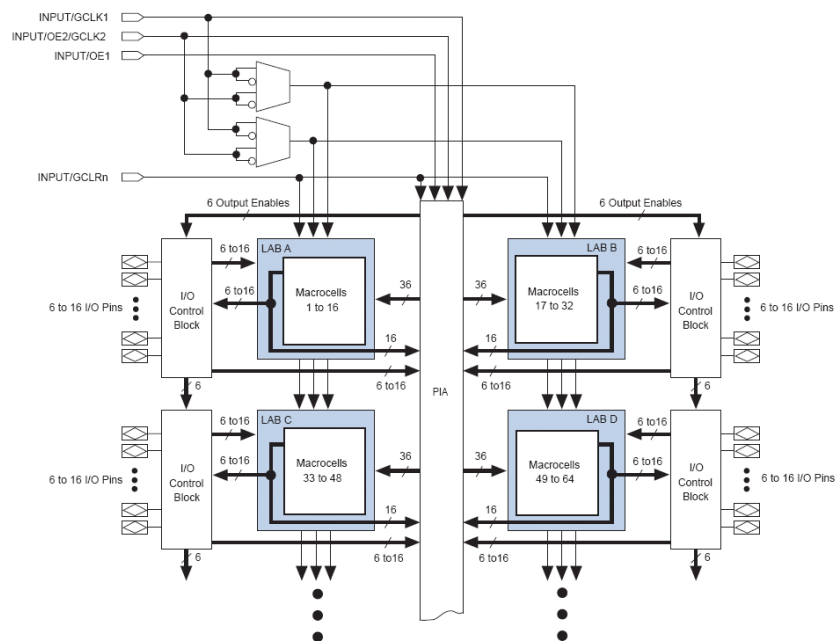
ברכיבי CPLD אמיתיים רבים, מסתפקים בחיבורייות שהיא אף נמוכה יותר ממה שהוצגה בדוגמה האחרונה.

מעוט החיבורים במטריצה הגלובלית אינו מהווה את הסיבה היחידה למהירות הגדולה של המטריצה הגלובלית. הסיבה העיקרית למהירות הגדולה של המטריצה הגלובלית היא שהטכנולוגיה של המטריצה היא כזו שהאותות אינם עוברים באופן ישיר דרך חיבורים מתוכנתים כל שהם (שהם החלק האיטי ברכיב המיתכנת). המטריצה שבדוגמה המיניאטורית שלנו בנויה בעצם באופן הבא:



שים לה שהאות עובר דרך הבוררים ללא מעבר כל שהוא דרך חיבור מיתכנת כל שהוא ורק כניסות הבקרה של הבוררים נשלטות על ידי חיבורים מיתכנתים.

האיור הבא מציג מבנה של רכיב CPLD ממשפחת MAX7000S של חברת Altera. מדובר בתת משפחה מאחת ממשפחות הרכיבים הפופולריות ביותר שנוצרו אי פעם (שיוצרה נמשך מעל לעשור - שנחשב זמן ארוך מאוד בעולם הרכיבים המיתכנתים).



באיור הנ"ל מתוארת המטריצה גלובלית שנקראת על ידי Altera בשם PIA

(Programmable Interconnect Array). סביבה רואים ארבע יחידות שיכולות להיחשב כיחידות PAL. חברת Altera אינה יכולה לקרוא ליחידות אלו בשם PAL, מכיוון שזהו שם מסחרי של חברת MMI, שהיא ממציאת רכיבי ה - PAL המקוריים. בהמשך שם זה נהיה שייך לחברת AMD שקנתה את MMI. הרכיבים המיתכנתים של חברת AMD שיכים כיום לחברה אחרת - Lattice. חברת Altera משתמשת במקום בשם PAL בשם LAB (Local Area Block).

כל רכיבי ה - PAL שמצויים מסביב למטריצה הגלובלית הם מסוג 36V16. צורת סימון זו של רכיבי PAL הונהגה על ידי חברת MMI. המספר הראשון - 36 מראה שבכל PAL קיימים 36 אותות שמנותבים למטריצת ה - AND של ה - PAL. ברכיבי CPLD האותות הללו מנותבים מהמטריצה הגלובלית. האות V מסמנת שמדובר ברכיב גמיש (Versatile) שמשתמש ב - Macrocell. המספר הסופי מראה שלכל PAL יש 16 Macrocells.

שים לב שכל רכיבי ה - PAL הם בעלי גודל זהה ובכדי ליצור CPLD גדול יותר, חברת Altera פשוט מוסיפה רכיבי PAL נוספים מסביב למטריצה הגלובלית.

שים לב שברכיב זה קיימים שני קיפולים (Double Fold-Back) של אותות לתוך המטריצה הגלובלית. בכל PAL מגיעים 16 קיפולים מכיוון ה - Macrocell. בנוסף לכך בכל PAL מגיעים בין 6 ל 16 קיפולים מכיוון הדקי הרכיב. מספר הקיפולים שמגיע מההדקים תלוי בסוג האריזה. באריזה שמשופעת בפינים יגיעו הרבה קיפולים מסוג זה. באריזה בעלת מספר קטן של פינים יגיעו מעט קיפולים מסוג זה.

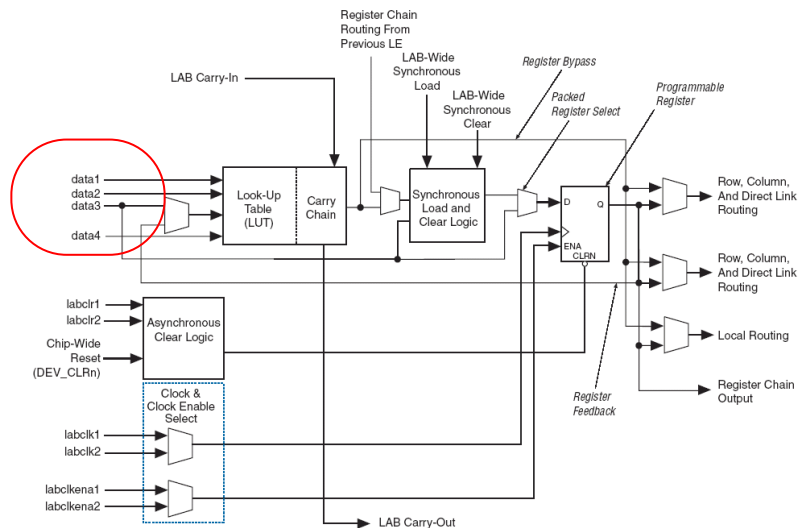
היחידות שמסומנות בשם I/O Control Block הן יחידות שמכילות את ה - Tri-State Buffer שמסוגל לנתק בין היציאה של ה - Macrocell וההדק החיצוני.

### 5.3 דוגמאות למשאבי חמרה בארכיטקטורות LUT של Altera

נתייחס תחילה לענייני טרמינולוגיה. רכיב מיתכנת גדול נקרא לעתים בשמות : FPGA (Field Programmable Gate Array) או CPLD (Complex Programmable Logic Device). השם הספציפי שנבחר לתיאור רכיב גדול אינו חד משמעי (תלוי את מי שואלים) וטרמינולוגיה זו אינה כל כך חשובה לנו.

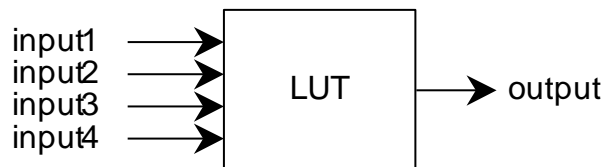
בחלק זה נציג אלמנטים מרכזיים בארכיטקטורות של רכיבים מיתכנתים גדולים של חברת Altera. הרכיבים הגדולים של חברת Altera (כמו גם של חברת Xilinx), אינם מבוססים על ארכיטקטורות Product-Terms (כמו רכיב MAX ורכיבי PAL), אלא על אלמנטים מסוג Look-Up-Table (LUT). מדובר במשפחות רכיבים מרכזיות של Altera שנקראות : APEX, ACEX, FLEX, Cyclone, Mercury ו Stratix (הרכיב שבו נשתמש בניסוי שיד למשפחת Cyclone - ה).

ה LUT - הוא טבלת האמת שממומשת על RAM (זיכרון כתיבה וקריאה) קטן, שמתפקד לאחר הקונפיגורציה של הרכיב כרכיב ROM. ה LUT - הוא החלק הצירופי של יחידה שנקראת Logic Element. להלן המבנה של Logic Element של רכיב בארכיטקטורת Cyclone-II שבו נשתמש בניסוי.



כפי שאפשר לראות באיור הנ"ל ה - LE (Logic Element), מכיל גם פליפ-פלופ אחד וכמה רכיבי עזר נוספים, שמטפלים בטעינת הפליפ-פלופ ובניתוב האותות מסביב ל - LE.

ה - LUT הוא רכיב בעל ארבע כניסות.



לרכיב LUT יש מספר כניסות (Fan-In) קטן, אך הוא מסוגל לממש כל פונקציה בעלת ארבע כניסות. אפשר לומר במלים אחרות שהמימוש של ה - LUT הנ"ל שווה ערך ל - 16 מכפלות קנוניות (מלאות) בארכיטקטורת PT.

רכיבי LUT מתאימים במיוחד למימוש פונקציות שיש להן מעט כניסות אך הרבה מכפלות, כמו למשל הפונקציה הבאה:

$$Y = A \cdot B \cdot C \cdot D + \bar{A} \cdot \bar{B} \cdot C \cdot D + \bar{A} \cdot B \cdot \bar{C} \cdot D + \bar{A} \cdot B \cdot C \cdot \bar{D} + A \cdot B \cdot \bar{C} \cdot \bar{D} + A \cdot B \cdot C \cdot D + A \cdot \bar{B} \cdot \bar{C} \cdot D$$

ארכיטקטורות שמבוססות על LUTs, פחות מתאימות למימוש פונקציות שמספר הכניסות שלהן (Fan-In) גבוה. להלן דוגמה לפונקציה בעלת Fan-In גבוה, אך בעלת מספר קטן של מכפלות.

$$Y = \bar{A}\bar{B}\bar{C}\bar{D}\bar{E}\bar{F}\bar{G}\bar{H} + \bar{A}\bar{B}\bar{I}\bar{J}\bar{K}\bar{L}\bar{M} + \bar{C}\bar{D}\bar{N}\bar{O}\bar{P}\bar{Q}\bar{R} + \bar{M}\bar{S}\bar{T}\bar{U}\bar{V}\bar{W}\bar{X} + \bar{C}\bar{Z}$$

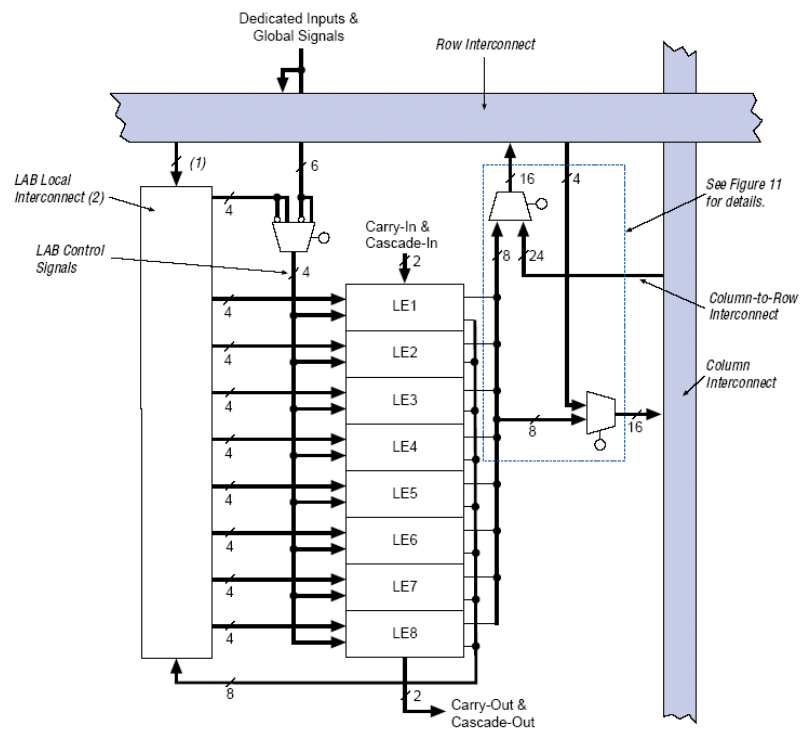
בעיית ה-Fan-In הנמוך של רכיבי LUT, מכתיבה שימוש במנגנונים שונים בארכיטקטורה, שמאפשרים להגדיל את ה-Fan-In. נציג כאן שלוש פתרונות אפשריים:

שימוש ב-LEs שכנים שמקובצים ביחידה שנקראת LAB

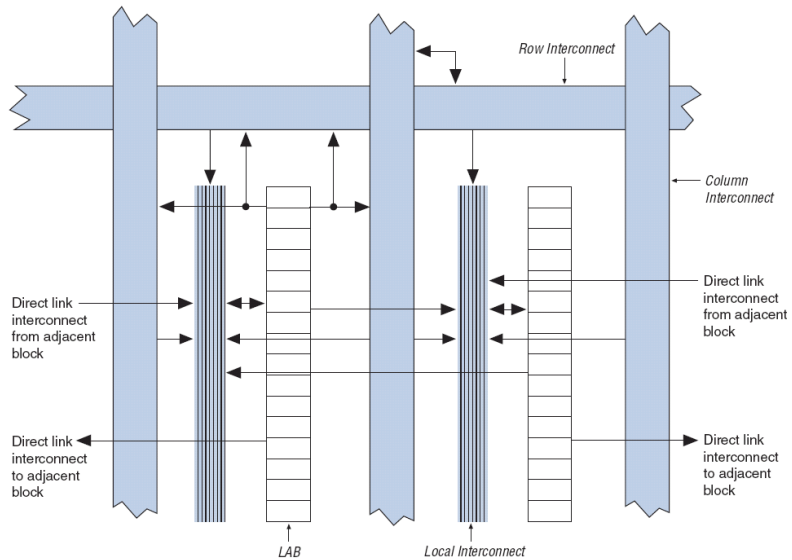
שימוש ב-Carry Chain

שימוש ברכיבי זיכרון גדולים יותר

בכל ארכיטקטורות ה-LUT של Altera, מקבצים מספר LEs ביחד ביחידה שנקראת LAB (Local Area Block). ברכיבי Flex מדובר ב-8 יחידות LE בכל LAB. האיור הבא מציג LAB אחד של ארכיטקטורת ה-Flex.

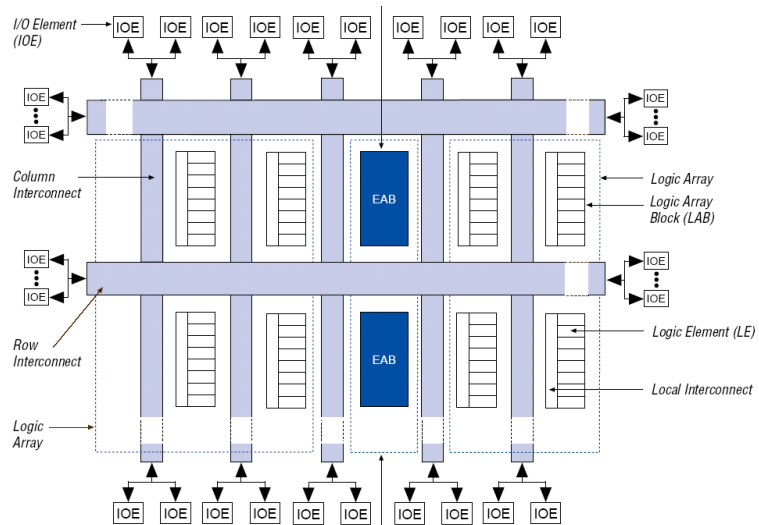


ברכיבי Cyclone מדובר ב - 16 יחידות LE בכל LAB. האיור הבא מציג שני LABs של ארכיטקטורת ה - Cyclone.



שים לב שבניגוד לארכיטקטורות של רכיב CPLD (כמו רכיבי ה - MAX), שלהם יש רק מטריצת חיבורים גלובלית אחת (שנקראת PIA במקרה של רכיבי MAX), הפעם מדובר ברכיבים שהם כולם דו-ממדיים. לרכיבים אלו יש מטריצות גלובליות (שנקראות Fast Track), שמסודרות באופן אנכי ואופקי ברכיב. לחוטים המתכתיים האופקיים של המטריצות קוראים Row interconnect ולחוטים המתכתיים האנכיים של המטריצות קוראים Column interconnect.

בקצוות של ה - Rows ו Columns, יש יחידות התחברות להדקים החיצוניים, שדומות במבנה שלהם ל - Macrocells של CPLD או PAL. יחידות אלו נקראות I/O Elements. אלו הן יחידות שאין בהן לוגיקה צירופית כל שהיא והן מיועדות לנתב ולסנכרן אותות שנכנסים או יוצאים מהרכיב. האיור הבא מציג I/Os בקצוות של רכיב Flex.

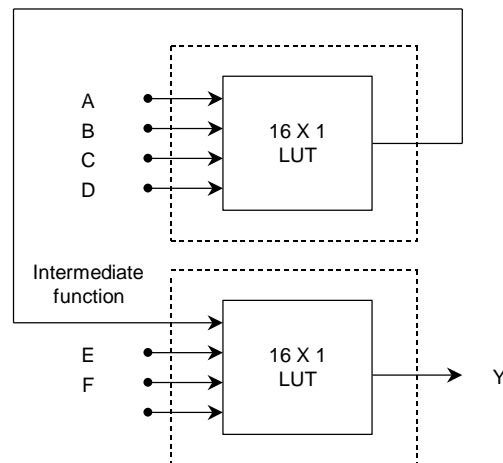


שים לב באיור הנ"ל גם ליחידות שנקראות EAB, שאליהן נתייחס בהמשך.

נחזור לעניין ה-Fan-In המוגבל. מה קורה כאשר רוצים לממש פונקציה שיש לה יותר מארבע כניסות, על גבי רכיב שהארכיטקטורה שלו מבוססת על LUTs בני ארבע כניסות? לדוגמה, נניח שורצים למשל לממש פונקציה שיש לה שש כניסות:

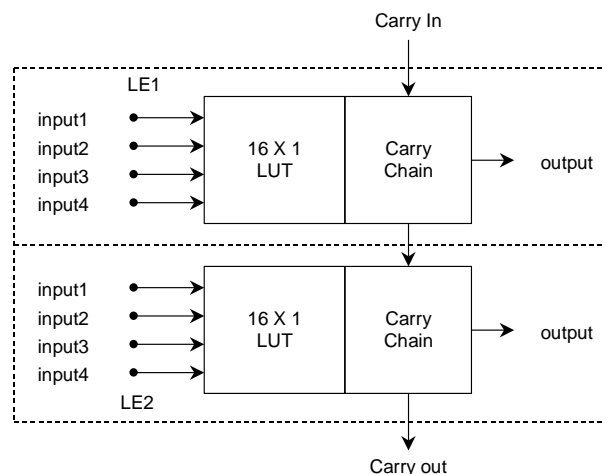
$$Y = f(A, B, C, D, E, F)$$

במקרה כזה המימוש צריך להתפצל בין שני Logic Elements. שני ה-Logic Elements יכולים להימצא באותו ה-LAB או ב-LABs שונים. בכדי ליצור חיבור בין שני ה-Logic Elements חייבים להשתמש במשאבי ניתוב חוטים (Routing) שמקשרים בין שני האלמנטים. האיור הבא מדגים את החיבור שנעשה בין שני האלמנטים.



חיבורים בין שני LUTs עלולים להאט את המערכת (כלומר להגדיל את ה-tpd של המערכת הצירופית). במיוחד הדבר נכון כאשר נעשים חיבורים בין שני LUTs שנמצאים ב-LABs שונים ושנעשה שימוש בניתוב באמצעות ה-Rows ו-Columns של הרכיב. כאשר נעשים חיבורים בין שני LUTs באותו ה-LAB עצמו, החיבורים נעשים באמצעות מערכת חיבורים מקומית (Local Interconnect), וההגדלה בזמן ההשהיה נמוכה יחסית בהשוואה להשהיה שנוצרת במקרה שבו מחברים בין LUTs שנמצאות ב-LABs נפרדים. ברכיבים מודרניים כמו Cyclone, קיימים גם חיבורים מהירים יחסית בין LUTs של LABs שהם שכנים.

ברכיבים מיתכנתים שמבוססים על LUTs, קיימים משאבי חמרה מיוחדים שנקראים Carry Chain, שיוצרים חיבורים ישירים ומהירים בין ה-LUTs השכנים באותו ה-LAB ואין צורך להשתמש במשאבי Routing חיצוניים.



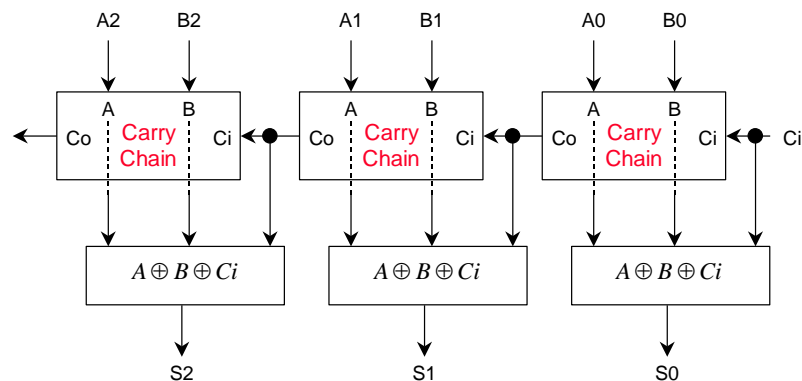
משאבי החמרה מסוג Carry Chain יוצרים תשתית של פונקציות Carry של Full Adder, בכל Logic Element. פונקציות ה-Carry היא הפונקציה Co שתוארה בטבלת האמת של Full-Adder שהוצגה באחד החלקים הקודמים.

$$C_o = AB + AC_i + BC_o$$

ה-Carry Chain מנוצל למימוש פעולות אריתמטיות שונות, וזאת מבלי שיש צורך להשתמש במשאבי Routing חיצוניים ואיטיים בכדי לאפשר הגדלה ב-Fan-In של רכיב אריתמטי. נניח שרוצים למשל לממש מחבר בן שלוש סיביות שמבצע את הפעולה הבאה:

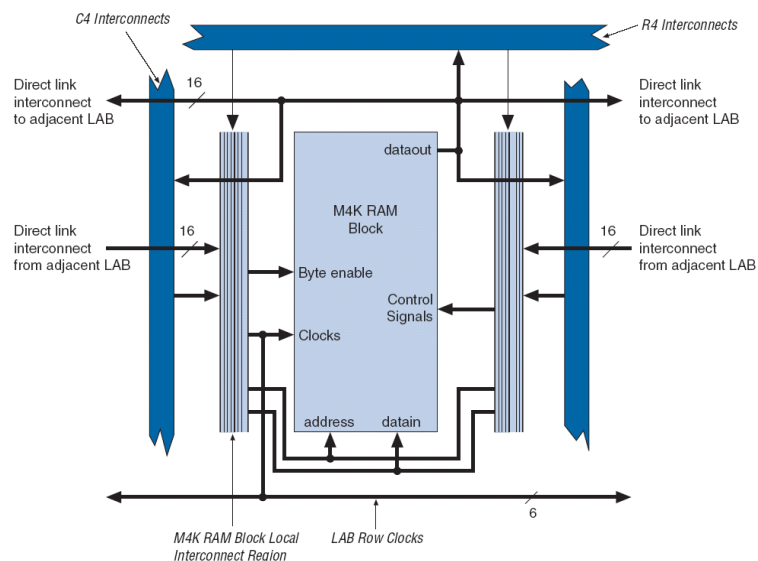
$$(Co, S2, S1, S0) = (0, A2, A1, A0) + (0, B2, B1, B0) + (0, 0, 0, Ci)$$

הסכימה הבאה מציגה מימוש של המערכת.



פעולות ה-Carry מומשו בדוגמה הנ"ל באמצעות ה-Carry Chain, ופעולת ה-XOR ממומשת באמצעות LUTs. גם פעולות אריתמטיות אחרות כמו: חיסור, הגדלה והקטנה וספירה, השוואה ועוד, יכולים להסתייע במשאב פנימי חשוב זה.

סוג אחר של משאב פנימי שקיים ברכיבים מיתכנתים גדולים של Altera הם רכיבי זיכרון גדולים. כבר ראינו קודם, באיור של רכיב ה-Flex, רכיב שנקרא בשם EAB (Embedded Array Block). שמות אפשריים אחרים לרכיבי זיכרון אלו הם ESB (Embedded System Blocks) ו-M4K Blocks. האיור הבא מתאר יחידה מסוג M4K Block ברכיב Cyclone.



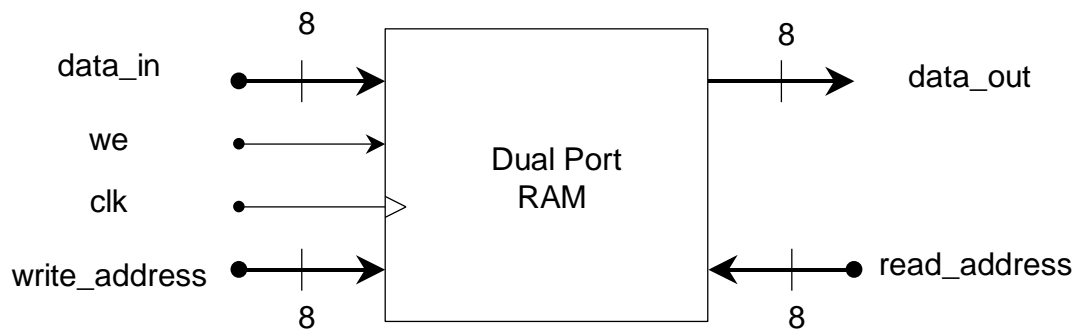


מדובר ברכיבי זיכרון מאוד גמישים שבהם המשתמש ברכיב, יכול להגדיר את רוחבו ואת אורכו של הזיכרון הרצוי לו. כלי הסינתזה Quartus דואג במידת הצורך הן להמרה בין אורך ורוחב באותה יחידת זיכרון והן לשרשור בין כמה יחידות זיכרון, וזאת במקרה שיחידה אחת אינה מספיקה. בניגוד לרכיבי זיכרון כמו ה-LUT, לרכיבים אלו יש כמובן Fan-In גדול יותר.

רכיבי זיכרון אלו יכולים לשמש כטבלאות ROM גדולות ואוניברסליות, אך הם יכולים לשמש גם כרכיבי זיכרון בעלי יכולת כתיבה וקריאה (RAM). הכתיבה לרכיב מאופשרת באמצעות כניסת אפשרור כתיבה (Write Enable).

משטר הכתיבה לרכיב הזיכרון יכול להיות סינכרוני - כלומר הכתיבה מסונכרנת לאות השעון. משטר הכתיבה לזיכרון יכול להיות גם א-סינכרוני ובמקרה זה הכתיבה נעשית כמו כתיבה ל-Gated Latch. הקריאה מהרכיב יכולה להיעשות במשטר עבוד צירופי (כמו ב-ROM) או במשטר עבודה סינכרוני.

ברכיבים שהם מודרניים יותר מרכיבי ה-Flex (כמו למשל ברכיב Cyclone שעליו מתבצע הניסוי), קיימת לרכיבי הזיכרון מערכת גישה כפולה ורכיבי הזיכרון יכולים להתנהג כרכיבי Dual Port RAM. אלו הם רכיבי שמאוד נוח להשתמש בהם בתכן ספרתי. נציג דוגמה. באיור הבא מוצגת דוגמה ל-DPRAM ברוחב שמונה סיביות שמאפשר כתיבה סינכרונית בכתובת אחת ובו זמנית קריאה מכתובת אחרת. מרחב הכתובות בדוגמה זו נע בין 0 ל-255.



קל מאוד לממש רכיב כזה על רכיבי Cyclone, מכיוון שמשאבי הזיכרון הפנימיים של רכיבי ה-M4K תומכים בסוג כזה של זיכרון.

משאבים חשובים נוספים שקיימים ברכיבים גדולים של Altera הם: יחידות אריתמטיות לחישוב פעולות כפל ופעולות DSP שונות, רכיבי PLL (Phase Locked Loop) להכפלת תדרי שעון ושינוי פזה של אותות שעון.

ציינו קודם ש-PLDs ו-CPLDs מודרניים מתוכנתים בדרך כלל באמצעות טכנולוגיה של כתיבה ומחיקה חשמליים (טכנולוגיית Flash). התכנות של הרכיב אינו נמחק לאחר כיבוי והדלקה של המתח.

כיצד מתכנתים רכיבים מיתכנתים גדולים שמבוססים על רכיבי LUT ?

שיטת התכנות היא שונה ! רכיבים אלו מתוכנתים באמצעות כתיבה לזיכרון מסוג RAM פנימי. מדובר ברכיבים נדיפים שנמחקים לאחר כיבוי המתח ויש לתכנת אותם מחדש לאחר כל הפעלה של המתח. אפשר להשתמש ברכיב זיכרון חיצוני שטוען את הרכיב מחדש בכל הדלקה של המתח. בניסוי לא נשתמש בזיכרון חיצוני אלא נתכנת את הרכיב באמצעות מחבר ה-JTAG באמצעות כבל USB שמחובר למחשב (על נושאים אלו נדון בקיצור בהמשך). התכנות של הרכיב נעשה כמובן במעגל עצמו ללא הוצאה של הרכיב.

כיצד תוכל לתקן את התכן הנ"ל בצורה הפשוטה ביותר כך שהוא אכן יפעל ?

## 6 מבנה קובץ TCL

הגדרת רכיבים

```
#=====
# DEFINE PROJECT NAME !!!!!!!!!!!!!!!
#set proj_name experiment_data_conversion
#=====

#project_open $proj_name
set_global_assignment -name FAMILY "Cyclone V"
set_global_assignment -name DEVICE 5CSXFC6D6F31C6
```

הגדרת פינים

```
#=====
# KEY
#=====
set_location_assignment PIN_AJ4 -to resetN ; # KEY[0]
set_location_assignment PIN_AK4 -to KEY[1]
set_location_assignment PIN_AA14 -to KEY[2]
set_location_assignment PIN_AA15 -to KEY[3]
```

מקרו לאילוח 3.3V ביציאות

```
# Assigning 3.3 V to all pins

#source constraints/setup_pinlist.tcl

set name_ids [get_names -filter * -node_type pin]

foreach_in_collection name_id $name_ids {
    set pin_name [get_name_info -info full_path $name_id]
    post_message "Assigning 3.3-V LVTTTL to $pin_name"
    set_instance_assignment -name IO_STANDARD "3.3-V LVTTTL" -
to $pin_name
}

export_assignments
```

## 7 חומר רקע הדרוש לניסוי במעבדה