

מבנה המחשב | 67200

הרצאות | אוהד פאליד ורונ גבר

כתביה | נמרוד רק

תשפ"ג סמסטר ב'

תוכן העניינים

3	I מבוא ומוליכים-למחאה
3	הרצאה
3	רקע כימי לרנזיסיטורים
4	רקע פיזיקלי לטרנזיסיטורים
5	מוליכים למחאה
5	צומת ח-ק
6	MOS-FET
8	בנייה של שערים מטרנזיסיטורים
8	תרגול
10	II שערים
10	הרצאה
11	סכום מכפלות ומכפלת סכומים
13	יחידות סטנדרטיות בمعالגים לוגיים
14	معالגים סדרתיים
16	معالגים סדרתיים מורכבים על בסיס SR-Latch
17	DFF
17	תרגול
19	מפות קרנו
23	פונקציות שלמות
23	III
23	הרצאה
26	FSM
29	זמן מעבד והגדירות זמני פעולה
33	תרגול
36	MIPS IV
36	הרצאה
37	RISC vs CISC
37	ריגיסטרים ב-MIPS
38	פקודות אРИתמטיות
38	פעולות לוגיות
39	פעולות זכרון
40	פילוסופיות ארגון זכרון
40	פקודות קפיצה והחניות
41	שימוש פונקציות באסבלי
42	פקודות קראיה לפונקציה
43	תרגול
43	אנליזה של מעגלים סינכרוניים
46	синטזה של מעגלים סינכרוניים

שבוע II | מבוא ומוליכים-למחצה

הרצאה

המחשבים הראשונים היו עצומים, כבדים ויקרות, ויחידת הבסיס של המעבד שלהם הייתה שפופורת קוודיות (אבי הטרנזיסטור) ואוז שפופורת ריק, וכיום משתמשים בטכנולוגיית CMOS שמאפשרת גדילה בעשרות סדרי גודל בזמן המוחזר, מהירות השעון, מספר הטרנזיסטורים ועוד. הקורס עוסק במבנה המעבד בעיקר.

בתוך כל מחשב יש אביזרי קלט ופלט (חישוני סונאר, מסך, עכבר), אמצעי אחסון (נדף כMO RAM ולא נדי' כמו דיסק קשיח), מעבד, מערכת הפעלה, דרייברים ותוכנה. בקורס עוסק במעבדים ותוכנה ונזכיר מערכות הפעלה ואמצעי אחסון.

קצב ההתקדמות עד לשנים האחרונות התנהג לפי חוק מור (1965) - כל שנה مضליים להכפיל את מספר הטרנזיסטורים כל שנתיים. העליה במספר הטרנזיסטורים מספקת גם עלייה אקספ' בביטויים, ביעילות אנרגיה וגם בגודל האחסון שאפשר לייצר.

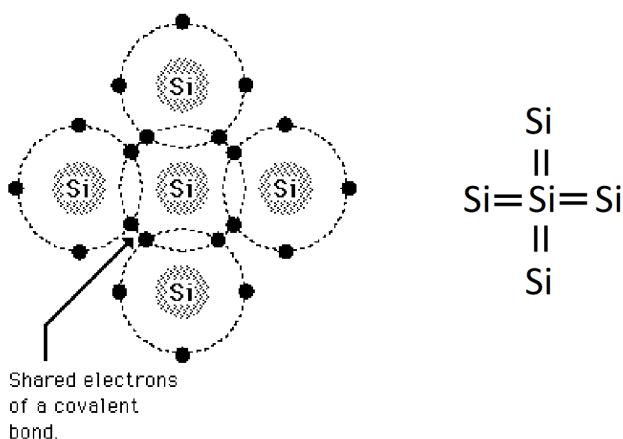
רקע כימי לטרנזיסטורים

המודל של בוחר ניסה להסביר את תופעת התכוונות המשותפות ליסודות באוטה עמודה של הטלחה המחזוריית אותה בנה מנדليب כמה עשרות שנים קודם לכן. המודל קובע שככל חומר מורכב מאטומים, שלהם יש גרעין עם פרוטונים (חלקיים עם מטען חיובי חיובי) וניוטרונים (חלקיים ללא מטען) ואלקטרונים (עם מטען שלילי) שסובבים את הגרעין.

דוגמה סיליקון (צורן) הוא מספר 14 בטבלה המחזورية, כלומר יש לו 14 פרוטונים (ובמקרה הזה גם 14 ניוטרונים) וכן אלקטرونים בשכבות שונות סביבה הגרעין עם מספר שונה של אלקטرونים בכל שכבה.

בזה גילה בוסף שהמסלול (המעגל, השכבה של אלקטرونים) האחרון של כל אטום במצב יציב הוא מלא, אך אטום ירצה לחת או לחת אלקטرونים מאטומים אחרים כדי לקבל מסלול מלא.

דוגמה הרבה אטומים של סיליקון יוצרים יחד במצב יציב שmorכב משציג אטומי סיליקון עם מסלול אחד מלא לכולם כך שהם חולקים אלקטرونים (ראו איור במקורה של חמיישה אטומים)



קשר בו אטומים חולקים (sharing) אלקטרונים נקרא קשר קוולנטי (covalent bond).

יוו הוא אטום או מולוקולה עם מספר לא שווה של פרוטוניים (+) ואלקטרונים (-) והמטען של החלקיק הוא הפרש הערכיים האלו. קשר יוני הוא קשר בין אטומים שנוצר כאשר מושלול האחרון של אטום אחד לאחר (כדי להשלים מסלול) אבל מספר הפרוטוניים באטומים שווה למספר האלקטרונים בכל אטום למעבר האלקטרון. כך לשנייהם כעת יש מטען מנוגד לשני וממש ניגודיות המטעןם מקרבת (באמצעות כוח משיכה אלקטرون-סטטי) בין האטומים לכדי קשר.

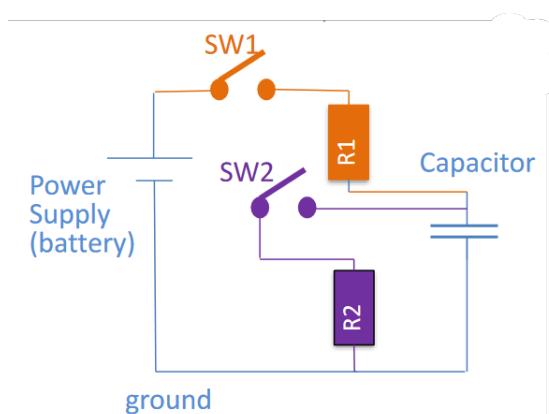
רקע פיזיקלי לטרנזיסטורים

זרם חשמלי הוא תנועה של אלקטרונים (או באנלוגיה תנועה של חורים, כאשר האלקטרונים השיליליים עוברים בין חורים שהם אחורי חיוביים). בكونבנצייה הזרם נע מה- + ל-.

מתוך (פוטנציאל) נמדד בולט והוא היכולת להזרים, כאשר סוללה עצמה פשוט מחזיקה מתח וכמו מגדל מים באנלוגיה מים, אפשר לעשות בו חור וכך לגרום לזרימה אבל כל עוד לא מחברים/מחוררים את הכליל לא יקרה שום דבר. מטען הוא מספר האלקטרונים שמאחזרים.

קיבול זו היכולת להחזיק מטען, כאשר בעת אחסון מטען נוצר מתח בקביל (מקביל למיכל מים ולחץ).

דוגמא נביט בمعالג הבא. R_1 הוא נגד (סקול לצינור צר יותר, שיוצר התנגדות). אם נסגור את המתג הראשון, נסגור מעגל בין הסוללה לקביל כך שאלקטרונים יזרמו מהסוללה לקבל מלמעלה ומהקביל לסוללה מלמטה והקביל יקבל את אותו המתח של הסוללה. אם ננטק את המתג הקביל ימשיך להחזיק את המתח, ואם נדליק את המתג השני יהיה לנו זרם קצר מהקביל אליו עם כיוון השעון (בחילוקו העליון של הקובל היוניים החיוביים ו מתחתיו השיליליים, והאלקטרונים הם אלו שענים).



בקשר של מחשבים נקבע מתח נמוך (עד 1.5V) להיות 0 ומח גבוה (לפחות 3.5V) להיות 1 - "כי כה". בין 1.5V ו-3.5V לא אמורים להיות במצב יציב, רק במעבר בין המצבים. האנלוגיה לכך היא האם מיכל מים והוא מלא או ריק.

מוליצים למתח

מוליך למתח הוא חומר שלא מוליך חשמל מאוד טוב (מוליכותו היא בין נחותת למוליך).

דוגמה סיליקון טהור הוא מוליך למתח כי בצורת הגביש עלייה דיברנו יש מעט מאוד אלקטرونים חופשיים (הרבה מהם תפוסים בין כמה אטומים בקשר קוולנטי) ולכן קשה להזורים דרכו זרם.

אילוח (doping) הוא תהליך שבו "מלככים" את הסיליקון.

- P-type : נוסיף לסיליקון קצת אלומיניום (לו 3 אלקטرونים מתוך 4 במסלול האחרון) כך שייקשר לאטומי הסיליקון כך שלחומר עכשו יהיה חסר אלקטרון והוא ישמש לקבל אותו, ואז אלקטرونים יעברו בклות בין האי-שלМОיות האלה (פעם ישלים את המחסור במוקד כלול אחד, ואז יקפו אחר, וכו'). החורים שנוצרים כשהאלקטרון הנדרש ליציבות נעים (לשם התיאוריה), בכיוון ההפוך מהאלקטرونים וכן נוצר זרם חיובי בכיוון ההפוך מתנועת האלקטרונים.
- N-type : העיקרון הנ"ל רק עם זרחן (לו 5 אלקטرونים לעומת 1 במסלול האחרון) כך שייעזר עודף אלקטرونים והאלקטرونים העודפים חופשיים לנوع לאן שירצטו וייצור זרם.

لمוליך למתח מסוג P ו-N אין מטען חיובי או שלילי אלא רק סיבות שונות לתנועת האלקטרונים כתוצאה מהרצון להשלים מסלולים אחרים בתאומים.

הצמדה של חומרים מסוג P ו-N יוצרת יוניים - האלקטרונים מ-N שמחים לקפוץ ל-P שם "צrik" אותם. שני כוחות פועלים בעת הצמדת חומרים שכזו :

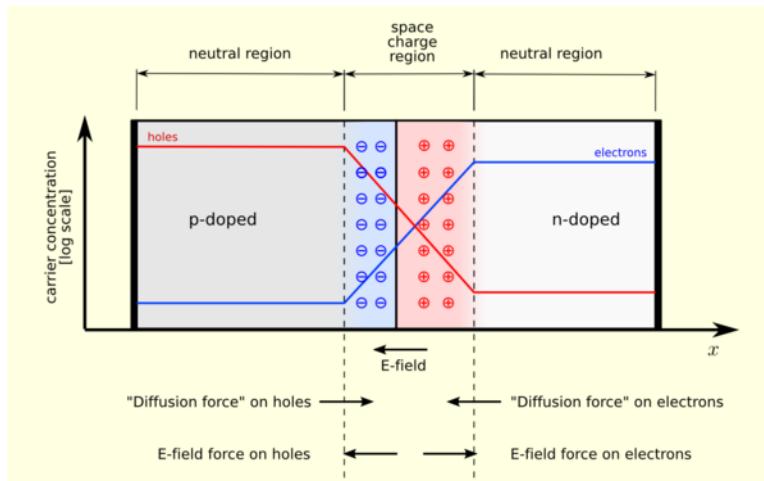
- הרצון של המסלולים להתملא - מה שגורם לאלקטרונים לקפוץ מ-N ל-P.
- הכוח החשמלי שיוצרים האלקטרונים - כוח דחיה בין מטענים שליליים שמונע קפיצה אלקטرونים מ-P ל-N (N אומר "יש לי מספיק שליליים כבר").

צומתpn

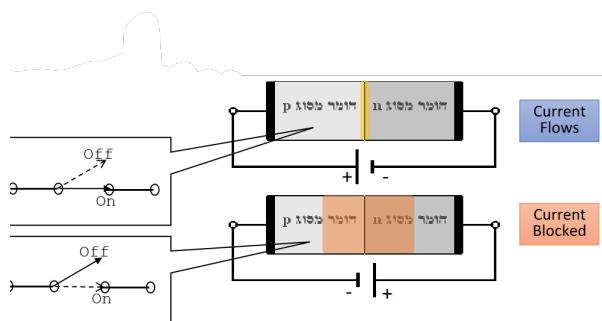
הצמדה זו נקרא צומת PN (PN junction) והוא מוליכה מכיוון אחד וחוסמת זרם מכיוון אחר (כמו שסתום!). המנגנון שהצומת מספק נקרא דיודה (diode) ומסומן באירועים הבאים



בשל תזוזה מקנית של אלקטرونים ופוטוונים בסמוך לצומת, מצלחים לעבר אטומים יוניים שליליים מ-N ל-P וחובבים להפוך (בניגוד לכיוון הזורימה). כמשמעות התחלפויות כאלה קורות, ישנה מסה של אטומים יוניים חיוביים ב-N ומסה של שליליים ב-P שלא יעברו לצד השני בגל שהם חלק מהגביש כבר. לצורך זה נוצר מחסום מכוח כוח הדחיה החשמלי שגורם להפסקת הזורימה דרך הצומת והחזקת מתח בו (ראוי איוור)



עתה חיבור סוללה לדיודה נותן לנו תכונות מעניינות.



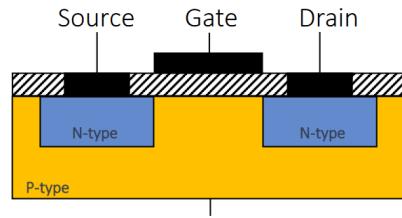
- חיבור סוללה עם + ל-P ו- ל-N יגרום להרבה מאד יוניים חיוביים לזרום מ-P ל-N ויוניים שליליים מ-N ל-P (בהתאם לכיוון הזרימה לפניה שנחסמה) וכך יצטמצם המרווח במרכז עד ל-0 ותהייה לו זרימה רגילה, כאילו סגרנו מתג.
- חיבור סוללה עם + ל-N ו- ל-P יגרום ליוניים חיוביים ושליליים להגיע לחומרים ולהזקק את היוניים במרווח שכרגע מונעים מעבר ורק יחזקו את החסימה, כך שלמעשה דימינו התנהגות של פתיחת מתג.

MOS-FET

טרנזיסטור MOS-FET עשוי שלושה חומרים: מוליך למחצה; תחמושת מבודדת; ומתקת. יש לו בנוסף שלוש רגליים: source ; drain ; gate (ורgel הארקה שמוחברת תמיד).

דרך הפעולה של וורייאנט ה-N-Channel

באיור ניתן לראות NMOS, וורייאנט מבין שניים של MOS-FET (השני משלים לו רק ש-N ו-P במיקומים הפוכים). החומר המוקוק הוא המבודד והשחור הוא המתקת.

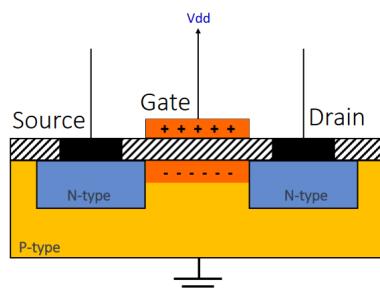


זכור כי זרימה ניתנת לקבל רק מ-P ל-N וכן אי אפשר אף פעם להזיר שום דבר מהמקור (source) לשפק (drain) ולהפך. יש לנו למעשה שתי דיזודות גב אל גב (שकצוטיתין המקור והשפך).

אם המתכת של השער מקבלת מטען, האзор בין השער למוליך-למחצה יהיה קובל כי הוא מחזיק הפרשי מטען!

הערה: את כל ארבעת הרגליים תמיד נחבר לאנזהו - או לאדמה או לטרנזיסטור אחר.

- אם נחבר את השער לאדמה (הארקה), יהיו לנו שני צמתים זה-ק שללא ניתן יהיה להזירם דרך (ובפרט בין S ל-D דבר).
- אם נחבר את השער למתח, הקובל שהזכרנו לעיל מתח ועכשו יש מטען חיובי מעליו ושלילי מתחתיו, ככלומר יישםALKTRONIIMI חופשיים ב-a-type-k, וכשהלו נוגעים בחומר הצדדים יוצרם ערז אלקטרוניים חופשיים דרכו כן יכול לעבור זרם, ומכאן השם **n-channel**.



מה שקיבלו בסופו של דבר הוא סויץ' שאנו יוכולים לשנות בו באמצעות זרם לשער (0 או 1). את הטרנזיסטור נסמן בסימול הבא –
עובד אותו הדבר רק הפוך – הזרמת "0" לשער מסגור את המtag ו-"1" תפתח אותו. ההבדל המהותי היחיד הוא מהחומרים הוא שיש מתח שמחובר מלמטה במקום הארץ. להלן טבלת סיכום של דרך הפעולה של הטרנזיסטורים.

Gate	"0"	"1"
N-MOS	"0" —	"1" —
P-MOS	"0" —	"1" —

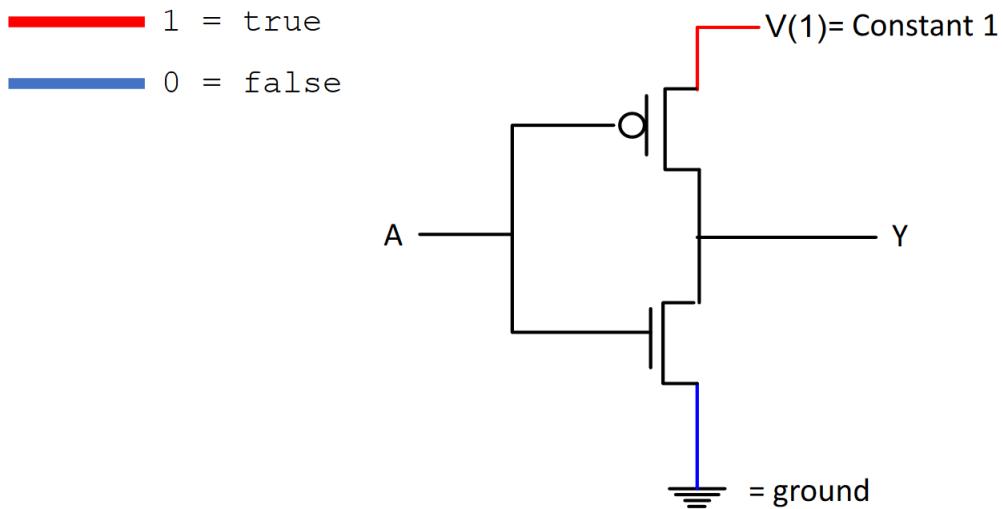
הערה: בתחום הכללי, מסמן שליליה, ב-PMOS למשל "שוללים" את המתח ומתנהגים הפוך ממנו. גם שערי NOT מסוימים עם עיגול.

בנייה של ריצוף מטרניזיסטוריים

טרניזיסטורים הם שימושיים לנו כי אנחנו יכולים לבנות מהם ריצופים לוגיים.

דוגמה בניית שער NOT (Inverter) שמסומן ב-

הבנייה דורשת שני MOF-SET-ים (P-Channel ו-N-Channel למטה, למטה), והיא כבאיור.



הוא הקלט ו- Y הוא הפלט. כשה- A הוא "1", יעבור זרם של 0 (שהוא זניח) דרך ה-N-Channel למטה (השער דלוק) ולא יזרום שום דבר דרך ה-P-Channel למיטה (השער דלוק לכך אין זרימה). סה"כ אין שום זרימה עם מתח לא זניח שכן Y יהיה "0".

בהתאם אם A הוא "0" אז ה-PMOS יזרום זרם קבוע של "1" ו- Y יהיה "1".

הערה צריκ את ה-NMOS התיכון כי אחרת ב-" $A = 1$ " יש לנו מעגל פתוח שיכול להיות לו כל זרם ולא בהכרח "0" כמו שאנו רוצים.

תרגול

בסיס ספירה הוא דרך לייצג מספרים ממשיים. בסיס עשרוני נבעץ את ההמרה הבאה

$$(a_4 a_3 a_2 a_1 a_0 \cdot a_{-1} a_{-2} a_{-3})_{10} = a_4 10^4 + \dots a_0 10^0 + a_{-1} 10^{-1} + a_{-2} 10^{-2} + a_{-3} 10^{-3}$$

בסיס בינארי משתמש בסיביות (ביטים) שערכן 0 או 1, בעשרוני 0 עד 9, ובקסדצימלי .0 – 9, $A - F$.

טווח המספרים בעל n ספרות בסיס r הוא $\{0, \dots, r^n - 1\}$

בקראה הכללי $\sum_{i=-n}^m a_i r^i$ מייצג את המספר בסיס r

פעולות חשבוניות אפשר לעשות באותו האופן כבסיס עשרוני (חיבור אורך שבו עברת ספרה הלאה, כאשר הספרות נגמרות לא בהכרח אחריו).

מעבר לבסיס 10 הוא די פשוט (פריסת הייצוג וחישוב מכפלות וחיבור בסיס עשרוני). מעבר מבסיס 10 לבסיס אחר הוא פשוט תהליך איטרטיבי של פעולות מודולו ('הבסיס) כאשר מה שהוא לא השארית יהיה הספרה הראשונה (משמאלו), השנייה, וכו' עד שנגמרות הספרות.

לא כלתי כאן אינסוף דוגמאות להמרות בין בסיסים כי לא מספיק משעמים לי.

במקרה הכללי נמיר משני בסיסים כלשהם דרך בסיס 10 כאשר את הרכיב משמאלו ומימין בספרה העשונית נטפל באופן נפרד וזהה עד כדי חלוקה איטרטיבית בשמאלו ומכפלה איטרטיבית בימין).

דוגמה נמיר את₁₀ (0.79272) לבסיס 4. נכפיל את המספר ב-4 ונקבל 3.17088. لكن הספרה הראשונה היא 3 והשארית היא 0.17088. נבצע זאת שוב ועתה נקבל 0 ושארית 0.68352, וחזור חלילה עד שהשארית תהיה 0, כאשר עתה הספרות ה-0 מלמעלה למיטה במקום למיטה למטה בספרות הרגילים.

שיטת לייצוג מספרים

- **שיטת גודל וסימן:** מספר יתחל בביט סימן (0 חיובי 1 שלילי) ושאר הביטים יהיו ייצוג בינארי של המספר.

$$\text{דוגמה } 9 = (-1)^0 (2^3 + 2^0)$$

טוחה הייצוג בשיטה זו והוא $(2^{n-1} - 1), \dots, 0, \dots, (2^{n-1} - 1)$

• **שיטת המשלים לאחד:** בית סימן, שלפי ערכו נדע האם שאר הספרות הן הערך הבינארי של המספר זהה או שזו הערך הבינארי של המשלים של המספר $-l = -(2^{n-1} - 1)$, ובנוסך הפיכת כל בית מספק לנו את השילילה של המספר. לדוגמה, 4 = 00100, ואם נהפוך כל בית נקבל 4 = 11011.

חישור מספרים הוא די פשוט: צריך לחבר את המספרים, ואם יש carry לאחר החישוב נמחק אותו ונוסיף אחד לתוצאה.

$$\text{דוגמה } 5 = 9 - 4 = (+9) + (-4) = 100100 + 11011 = 01001 + 11011 = 100100 \text{ זה הופך ל-00100.}$$

טוחה הייצוג הוא כמו בשיטת גודל וסימן ויש בו, כמו בשיטה הקודמת 0 חיובי ו-0 שלילי.

- **שיטת המשלים לשתיים:** בית סימן, שעבור מספרים שליליים ערכו הנגדי למספר שמתקיים מהיפוך הביטים והוספה אחד.

$$\text{דוגמה } 4 = - (00011 + 1) = -00100$$

לחולופון לחישוב המשלים אפשר ללבת מימין לשמאל עד ל-1 הראשון, לא לשנות אותו, ואז להפוך את כל מה שמשמאלו (ואז לא צריך להוסיף 1).

דוגמה כמה שווה 5 – בשיטת המשלים ל-2? 5, נוסיף אחד ונשנה את בית הסימן ונקבל 10110.

כדי לחסר מספרים, נסכום אותם ונמחק carry אם יש.

הגדירה overflow הוא מצב שבו אנחנו סוכמים שני מספרים באותו הסימן ומקבלים מספר בסימן הפוך, במקרה זה התוצאה כמובן שגויה.

דוגמה נשתמש בשיטת המשלים ל-1 עם 5 ביטים, $11000 + 01101 = 01011$ כלומר סכמנו חיוביים וקיבלו תוצאה שלילית!

הערה הפתרון overflow הוא הוספת ביטים כך שיגדל טווח הייצוג.

שבוע III | שערים

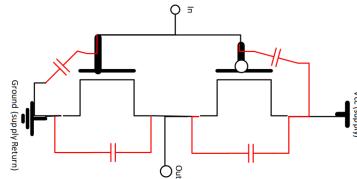
הרצאה

הערה כל שער שנבנה נבנה סימטרית מבחינת השימוש ב-PMOS ו-NMOS, שכן השערים עם טרנזיסטורים אלה נקראים .MOS

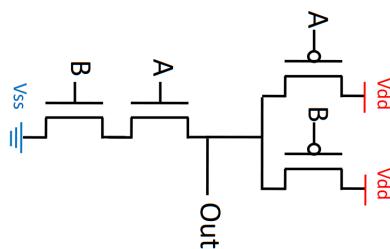
הערה לעולם לא נרצה להזרים זרם אל תוך טרנזיסטור אחר דלوك (מהכיון הלא נכון) כי זה גורם לסתור.

שער אחד בפני עצמו זה נחמד, אבל מעבדים בונים ע"י חיבור שערים. את השערים נחבר עם חומר מוליך בין הקלט של שער אחד לפלט של השער שמןנו אנחנו לוקחים את התוצאה.

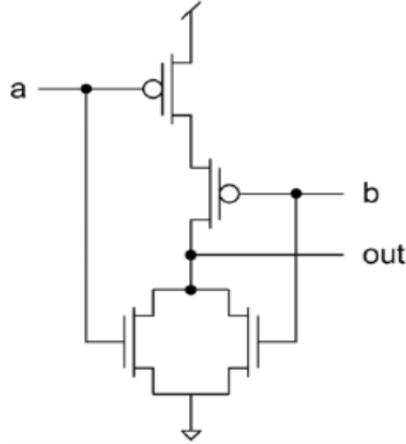
בין שער למקור, בין שער לשפק ובין שער לניצרים קבלים אפקטיביים (לא הוספנו שום דבר, פשוט יש רכיבים פיזיים الموجودة עצמן מחזיקים מתח בין הצדדים). את הקבלים האלה לוקחים זמן לטעון או לפרק מתח בעת שינוי מצב (שינוי הזרם מהשער, הזרם מהמקור). לכן במצבים רגילים יש זרימה שאנו נא בהכרח מצפים לה במהלך המעבר ($10\text{--}15\text{ ns}$ שנייה). ראו באירור באדום את הקבלים שנוצרו.



השער היסודי שמאפשר לבנות את כל שער השערים הוא NAND - שנותן 0 אם "ס שני הפלטים 1 (ואהחרת 1), ומסומן כך \overline{AB} . ניתן לבנות את NOR ב奧פן הבא (הריצו פלטים כדי לראות שזה עובד).



בדומה ניתן לבנות שער NOR ב奧פן הבא (קו אלכסוני הוא (1) V קלומר מתח קבוע דלוק וחץ הוא (0) GND (0) קלומר הארקה)



הגדירה פ' בוליאנית היא פ' שמקבלת קלטיים בוליאניים (משתנה שיכول לקבל אחד מבין שני ערכיים, ביטים לדוגמה) ופולטת פלט בוליאני אחד בדיקוק.

הערה מעתה נסמן x' להיות החפה ל- x (כלומר שהפעלו עליו שער NOT).

דוגמא $F = xy'$ היא פ' שבבינהו y, x , פולטת פלט שערכו x וגם לא y .

סכום מכפלות ומכפלת סכומיים

הגדירה בהינתן משתנים בוליאניים לפ' בוליאנית כלשהי, minterm הוא מכפלה (AND) של ליטרלים, כאשר ליטרל הוא משתנה או היפוכו וכל משתנה מופיע בדיקוק פעם אחת כליטרל או בתוך ליטרל מהופך.

דוגמא עבור שלושה משתנים וההשמה $x = 1, y = 1, z = 1$ המינטרם היחיד שערךו 1 יהיה $m_6 = xyz'$, ועבור $x = 1, y = 1, z = 0$ המינטרם $m_7 = xyz$ (אינדקס ה-minterm עם ערך 1 מתќבל ע"י הערך הדצימלי של הסטריאג הבינארי המתקבל מהצמדת ההשומות במשתנים).

נשים לב שש-minterm מקבל ערך 1 בבדיקה שורה אחת של טבלת אמת מלאה על המשתנים.

הגדירה בהינתן השמה במשתנים, maxterm הוא סכום (OR) של המשתנים או היפוכיהם כך שכל משתנה מופיע פעם אחת בדיקוק.

דוגמא עבור שלושה משתנים עם ההשמה $x = 1, y = 1, z = 0$ המקסטרם היחיד שמקבל ערך 0 הוא $M_6 = x' + y' + z$ וכו'.

הערה m_i משלים ל- M_i .

הגדירה Sum of Products הוא סכום מכפלות minterm-ים ו-maxterm-ים.

דוגמא הפ' F_1 עם טבלת האמת הבאה

x	y	z	F ₁	Minterm	Maxterm
0	0	0	0	m ₀ =x'y'z'	M ₀ =x+y+z
0	0	1	1	m ₁ =x'y'z	M ₁ =x+y+z'
0	1	0	0	m ₂ =x'yz'	M ₂ =x+y'+z
0	1	1	1	m ₃ =xyz	M ₃ =x+y'+z'
1	0	0	1	m ₄ =xy'z'	M ₄ =x'+y+z
1	0	1	0	m ₅ =xy'z	M ₅ =x'+y+z'
1	1	0	1	m ₆ =xyz'	M ₆ =x'+y'+z
1	1	1	0	m ₇ =xyz	M ₇ =x'+y'+z'

ניתנת לייצור ע"י

$$F_1(x, y, z) = m_1 + m_3 + m_4 + m_6 = x'y'z + x'yz + xy'z' + xyz'$$

הטכנית היהתה לסכום את כל ה-minterm-ים שמקבילים 1 בפ' (בכלוח).

לחולופין וכל לייצג את הפ' עם PoS ע"י הכפלת כל ה-maxterm-ים שמקבילים ערך 0 (באדום) כולם

$$F_1(x, y, z) = M_0 \cdot M_2 \cdot M_5 \cdot M_7$$

כולם הצנו בצורה קוננית פ' לכאהורה מרכיבת!

הערה למה משתמשים כל הייצוגים השונים (טבלת אמת, PoS, SOP ופתח קרנו שנלמד בתרגול)? נרצה בסופו של דבר את הייצוג המינימלי, כך שנדרש למספר הקטן ביותר של שערים כדי למשמש אותן.

דוגמה מולטייפלסקר (MUX) מקבל שלושה קלטים : S(*elect*), D0, D1 . בטבלה X משמעו "לא משנה" מה הערך"

S	D1	D0	Y
0	X	0	0
0	X	1	1
1	0	X	0
1	1	X	1

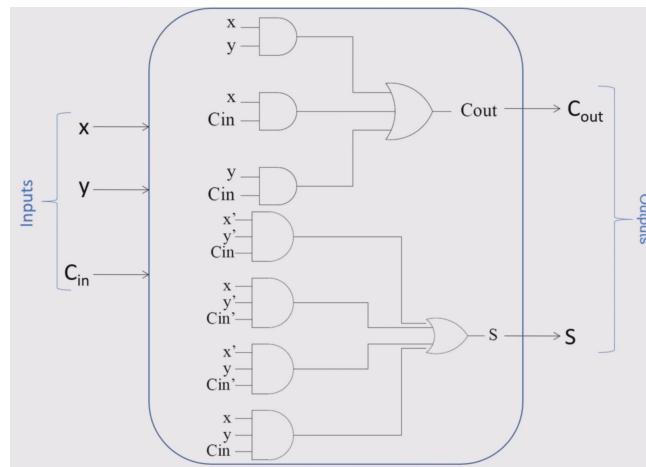
וניתן לרשום את הפ' כ- $MUX(S, D_0, D_1) = S \cdot D_1 + S' \cdot D_0$, ואת זה ניתן למשם באמצעות שערים שכבר בנינו. עם זאת, ניתן למשם את המעגל עם פחות טרנזיסטורים מאשר בIMPLEMENTATIONusing AND, OR ו-NOT.

דוגמה הוא מעגל שמקבל C_{out} (כאשר S, C_{in} נשא מסכימה קודמת) ופולט C_{out} כאשר S הוא הסכום והוא הנשא (overflow)

Truth Table					
x	y	C _{in}	S	C _{out}	
0	0	0	0	0	
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	
1	0	0	1	0	
1	0	1	0	1	
1	1	0	0	1	
1	1	1	1	1	

ניקח את $S = 0, C_{out} = 1$ $x + y + C_{in} = 0 + 1 + 0 = (10)_2$, או $x = 0, y = 1, C_{in} = 1$

למעשה, בגלל שיש למעגל שני פלטים, הרי שהוא מורכב משתי פ' בוליאניות. כרגע אפשר למשת את המעגל באמצעות SoP (סכימות minterm-ים), ובמימוש הבא צמצמו כמה minterm-ים באמצעות מנות קרנו שנלמד בהמשך (בנוסף מופיע במעגל OR עם שלישה וארבעה כניסה - הסתודנטית המשקיעה תראה כיצד ניתן לעשות זאת עם פי 2 טרנזיסטורים ממספר הכניסות).



הערה קל לשדרר כמה FA-ים כדי לקבל מעגל שסוכם מספרים עם מספר ביטים גדול יותר (לוקחים את C_{out} של ה-FA על זוג הביטים הראשונים, מכנים אותו ל-FA ווחזר חלילה).

יחידות סטנדרטיות במעגלים לוגיים

1. **מפענה** (Decoder): הקלט הוא n ביטים d_0, \dots, d_{k-1} כאשר $k = 2^n$ והפלט הוא x_0, \dots, x_{n-1} .

$$d_j = \begin{cases} 1 & (j)_{10} = (x_{n-1} \dots x_0)_2 \\ 0 & \text{אחרת} \end{cases}$$

כלומר פורש וקטור של n ביטים על 2^n ביטים שכל אחד מייצג מספר מותך $\{0, \dots, 2^n - 1\}$.

ברגע שיש לנו בлок של מפענה, אפשר להשתמש בו כדי למשה פ' בוליאנית באופן טריוויאלי, כי כל מה שצריך לעשות זה לעשות OR על כל d_i -ים שמייצגים x_0, \dots, x_{n-1} שמקבל ערך 1 בטבלת האמת של הפ' הבוליאנית.

2. מפלג (DeMultiplexer) : הקלט הוא f_0, \dots, f_{k-1} כל אחד בגודל בית והפלט הוא x, s_0, \dots, s_{n-1} כאשר $k = 2^n$ ו $x = (s_{n-1} \dots s_0)_2$.

$$f_j = \begin{cases} x & (j)_{10} = (s_{n-1} \dots s_0)_2 \\ 0 & \text{אחרת} \end{cases}$$

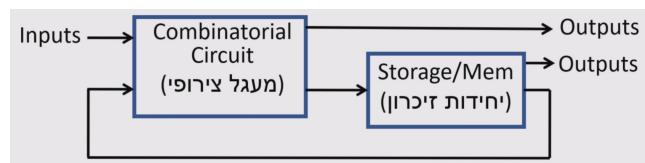
כלומר בהינתן בית, מחזיר מחרוזת שבה הכל אפסים חוץ מאולי הבית ה- $(s_{n-1} \dots s_0)_2$ שערכו x .

3. מקודד (Encoder) : הקלט הוא ביטים x_0, \dots, x_{k-1} כאשר $k = 2^n$ והפלט הוא ביטים e_0, \dots, e_n כאשר אם $x_i = 1$ עבור i אחר בדיקות (וכל השאר אפסים) אז $(e_{n-1} \dots e_0)_2 = (i)_{10}$, כלומר מחזיר את האינדקס (בבינה-ארית) של הבית היחיד הדלוק בקלט.

4. מרובב (Multiplexer) : הקלט הוא ביטים s_0, \dots, s_{n-1} וביטים x_0, \dots, x_{k-1} והפלט הוא f שהוא ערך הבית עם אינדקס $.x \cdot (s_{n-1} \dots s_0)_2$.

מעגלים סדרתיים

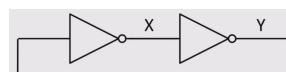
הגדרה מעגל סדרתי הוא מעגל קומבינטוררי שחלק מהקלטים שלו הם פלטמים של ייחידת זיכרון שמחזיק השער (ראו איור)



מעגלים סדרתיים אסינכرونיים יכולים לשנות מצב בכל זמן, ואילו מעגלים סינכרוניים משנים מצב בהתאם לשעון.

הגדרה שעון סיגナル בצורה גל מחזורי (או 1 ואו 0 ואו 1 וכו').

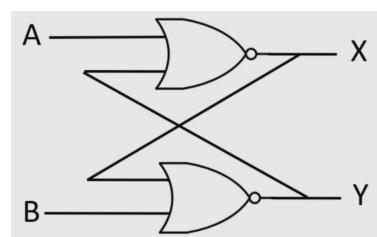
דוגמה כיצד השער הבא יתנהג?



אם הקלט בהתחלה הוא $X = 0, Y = 1$ או $X = 1, Y = 0$ ונקלט מצב יציב של $(X, Y) = (0, 1)$. בדומה עבור

כלומר, יש לנו מערכת דו-יציבה (עם שני מצבים יציבים), שיכולה לשמש אותנו לאחסון זכרו.

דוגמה נבית בשער הבא, שנקרא SR Latch (נזכיר שהשערים במעגל הם NOR-ים)



הפ' הזו מקיימת $X(t+1) = (A + Y(t))'$, $Y(t+1) = (B + X(t))'$ לאחר שינוי קלטים מסווגין עם שעון שביצע t טיקים), או בטבלת אמת עמוסה

A	B	X(t)	Y(t)	X(t+1)	Y(t+1)
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
0	1	1	1	0	0

A	B	X(t)	Y(t)	X(t+1)	Y(t+1)
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

ואם נשלוף רק את הערכים המעניינים, נוכל להביט בתופעה מעניינת (חצים מעגליים משמע לאחר טיק נשאר באותו מצבים וחציהם אומרים שנעבור לזוג ערכים אחר)

	A=0, B=0		A=0, B=1		A=1, B=0		A=1, B=1	
	X=0	X=1	X=0	X=1	X=0	X=1	X=0	X=1
Y=0								
Y=1								

Stable Clear (Y) Set (Y) Undefined

ונשים לב שם $X = Y$ נקבל מצב לא יציב (לא משנה מה ערכי A, B תמיד יש חץ שיוציא אותו למצב אחר) ולכן תמיד נניח שאנו נמשתמשים ב-SR Latch כאשר $Y = X'$ (זה דומה למצב בדוגמה הקודמת שבו $Y = X = 0$ שהוא שונה לא מוגדר בכלל כי יש לנו מהפך עם אותו ערך משני הצדדים).

אם כן, עבור $(X, Y) = (0, 0)$, נקבל ש- $(X, Y) = (A, B) = (0, 1)$, ואם $(X, Y) = (0, 1)$ אז מוגדר מצב לא מוגדר שנטעלם ממנו.

לכן, באמצעות $(S, R) = (A, B)$ נוכל לשולוט בערכו של Y כשייש לנו זכרון של המצב הקודם, עם טבלת אמת מצומצת נוחה ביותר, כאשר $Q(t) = Y(t) = X(t)'$ (בהתעלם מהמקרים הלא חוקיים)

S	R	Q(t+1)	Q'(t+1)
0	0	Q(t)	Q'(t)
0	1	0	1
1	0	1	0
1	1	0	0

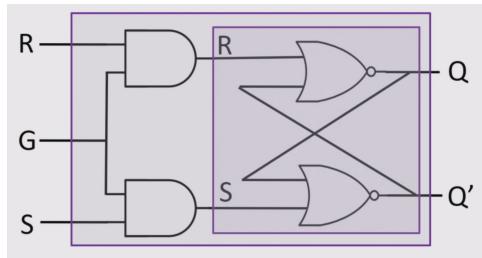
הערה מעגל סדרתיים ניתן לייצג באמצעות טבלת עירור (הטבלה הנ"ל) וטבלת מעברים, שיענה על השאלה "אילו קלטים נדרשים כדי לעבור בין מצב כלשהו לאחר" (Φ הוא ערך Don't care)

From Q(t)	To Q(t+1)	S	R
0	0	0	Φ^+
0	1	1	0
1	0	0	1
1	1	Φ	0

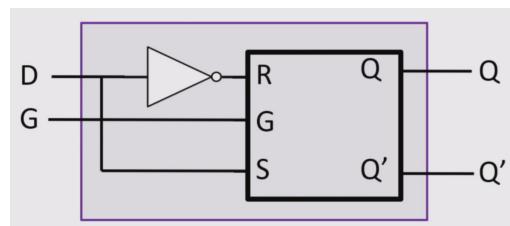
הערה יש היגיון בשמות הביטים S,R - אם רק (et) S דלוק, קובעים את הפלט להיות 1, אם רק (eset) R דלוק קובעים את הפלט להיות 0, ואם לא זה ולא זה דלוקים לא עושים כלום, כלומר שומרים על המצב כמוות שהיה. כמובן שתחת סימולדים אלה, גם S וגם R דלוקים זה לא מוגדר היטב.

מעגלים סדרתיים מורכבים על בסיס SR Latch

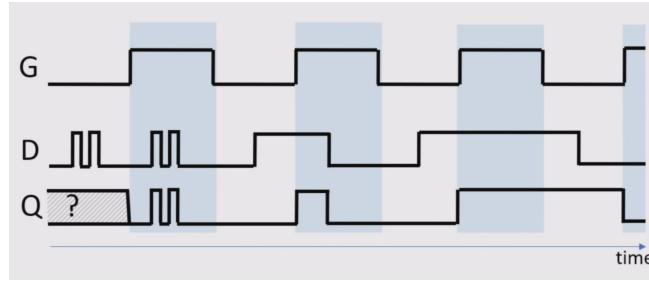
- **Gated SR Latch** הוא שער שמנוע פליטתית ערכיים לא חוקיים מ-S,R, והוא מקבל קלטיים S,R כאשר G הוא בית שער שאם הוא דלוק אז המעגל מתפרק כ-RLatch וgil, ואם כבוי אז הפלטים לא משתנים לא משנה מהו. המימוש הוא די פשוט, כולל AND של G עם S,R לפני הכניסה למעגל הפנימי (ראו איור)



- **Gated D-Latch** הוא גרטה אפילו יותר בטוחה ל-S,R - Gated SR Latch, נקלט D שייהי מחובר ל-S ודרך מהפץ ל-R, וכן לא נקלט מצב של (1,0) וכדי לקבל (0,1) אפשר פשוט לכבות את G (ראו איור)



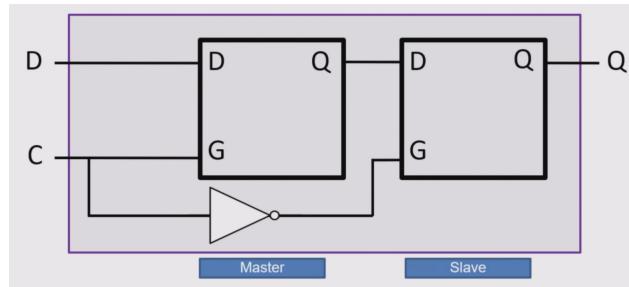
הערה לרוב נחבר את השעון ל-G, כלומר אפשר לשנות את הערך רק כשהשעון בטיק שערכו 1. דוגמה בהינתן שעון שמחובר ל-G ובית D שערכו משתנה, יוכל לחשב מה יהיה הפלט Q, כפ' של הזמן. השיטה המקורית בהתחלה פירשו שלא ידוע לנו מה הערך של Q שכן הוא לא מוגדר בשלב הזה.



D Flip Flop

בנה מעגל Shifter, שהוא מעגל שבו בית הקלט Z צעד אחד ימינה בכל מחזור. אם נשרשר D-Latch-G-ים שכלה-G-ים שליהם מחוברים לשעון, הקלט יופיע מיד בסוף השרשור (למעט זמן פגיעה של החישמל שהוא זניח).

הפתרון לזה הוא להוסיף מהפץ בין השעון ל-Latch השני כך שכשהשעון ב-1 רק ה-Latch הראשון יוכל לשנות את ערכו וכשהשעון ב-0 רק ה-Latch השני ישנה את ערכו והראשון לא ישנה. שער כזה נקרא .D Flip Flop



לכן רק בעת ירידת השעון מקבל שינוי של הפלט Q , כי לאחר העליה ה-Master ישנה את הפלט ולאחר הירידה ה-Slave ישנה את הערך, הלא הוא פلت המעגל כולו. כל עוד השעון לא יריד, הערך ישאר זהה לערך שקיבל בירידת השעון האחורונה.

הערה ניתן לבנות מעגל שישתנה רק בעליה באמצעות הזזת המהפק לפניו השער של המאסטר ולא העבודה.

תרגול

הגדרה אלגברה בוליאנית היא מבנה אלגברי המוגדר על קבוצת איברים B עם שני אופרטורים בינאריים $\cdot, +$, מתקיימות אקסiomות Huntington: סגירות לכפל וחיבור, קיום איברי ייחידה לכפל וחיבור, קומוטטיביות בחיבור וכפל, דיסטרויטיביות (שני הנוסחים), קיום משלים (כך $x \cdot x' = 0, x + x' = 1$) ו- $|B| \geq 2$.

הגדרה אלגברה בוליאנית דו-ערךית מוגדרת על קבוצה בת שני איברים $B = \{0, 1\}$ עם האופרטורים $x \cdot y = \text{AND}(x, y), x + y = \text{OR}(x, y)$ ו- $x' = \text{Not}(x)$

טענה באלגברה בוליאנית דו-ערךית מתקיימות התכונות הבאות:

- אידempotentיות: $x \cdot x = x + x = x$ לכל x .

- $x \cdot 0 = 0, x + 1 = 1$

- אסוציאטיביות לחיבור וכפל.

- חוק הצמצום : $x(x+y) = x, x+xy = x$

- $(x')' = x$

הערה סדר האופטורים בחישוב ביטויי בוליאני הוא קודם סוגרים, אז NOT, אז AND ואז OR.

דרכים לבטא פונקציה בוליאנית

- ביטוי בוליאני (ביטוי על המשתנים עם שני האופרטורים ושלילה).

- טבלת אמת : לטבלה יהיו 2^n שורות כאשר יש n קלטים.

- סכום מכפלות : מספיק שמכפלה אחת תהיה 1 כדי שהסכום יהיה 1. כדי להציג סכום מכפלות, סוכמים את כל המכפלות הסטנדרטיות (minterm) שמקבלות 1 בטבלת האמת.

המשתנה ה- j מקבל שלילה ב-minterm ה- i אם "ס הבית ה- j " ב- $z_2(i)$ הוא 0.

- מכפלת סכומים : מספיק שסכום אחד יהיה 0 ואז כל המכפלה היא 0. כדי להציג סכום מכפלות עם שלילה על הביטוי ושימוש בשני כללי דה-מורגן.

המשתנה ה- j מקבל שלילה ב-maxterm ה- i אם "ס הבית ה- j " ב- $z_2(i)$ הוא 1.

נרצה לצמצם את מספר הליטרלים שלנו (מספר השערים).

דוגמה נביט ב- $F(x, y, z) = xy' + x'z - F_1(x, y, z) = x'y'z + x'yz + xy'$. את הפ' השנייה ניתן למש עם משמעותית פחות שעריםalogisms (יש הרבה פחות פעולות) אבל הפ' שקוילות ולכן את השנייה תמיד!

את השקוילות אפשר להראות עם טבלת אמת או באמצעות אלגברה בוליאנית :

$$\begin{aligned} xy'z + x'yz + x' &= x'zy' + x'zy + xy' \\ &= x'z(y' + y) + xy' \\ &= x'z + xy' \\ &= F_2(x, y, z) \end{aligned}$$

$$\begin{aligned}
 F(x, y, z) &= (x + y)[x'(y' + z')]' + x'y' + x'z' \\
 &= (x + y)[x + (y' + z')'] + x'y' + x'z' \\
 &= (x + y)(x + yz) + x'y' + x'z' \\
 &= (xx + xyz + yx + yyz) + x'y' + x'z' \\
 &= \dots = 1
 \end{aligned}$$

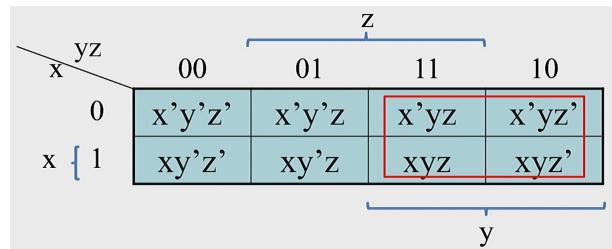
מפות קרנו

מפת קרנו בונים פעם אחת לכל הפ' הبولיאניות ב- n משתנים. ל- n משתנים יש מפת קרנו עם 2^n משבצות.

ראשית נבנה טבלת אמת לפ' הפוליאנית, ואז נציב את ה-minterm-ים בשבלונה של מפת הקרנו אם נצליח לשன אותה.

				y	
		0	0	1	1
	0	m_0	m_1	m_3	m_2
x	1	m_4	m_5	m_7	m_6
		0	1	1	0
		z			

לחולופין נוכל לבנות מחדש את הטבלה עם האינטואיציה שבסיסנים הכהולים למשתנים, באמצעות ערכי x ו- yz ניתן להסיק בקלות את המinterm-ים (יש לשים לב שערכי yz הם לא לפי סדר לקסיקוגרפי)



הערה ארבעת המשבצות ש- z מסומן עליהם נסכום לליטרל ייחיד שהוא z , כך גם על y , וכך גם השורה התוחטונה נסכמת ל- x . נשים לב גם כי כל שני ריבועים סמוכים נבדלים בליטרל אחד בלבד.

כדי לבצע צמצומים נמצא קבוצות של ריבועים סמוכים שערכים בטבלה האמת 1 עבור הפ', כאשר הקבוצה חייב להיות חזקה של 2 (כולל 1) ועלינו לבחור קבוצות גדולות ככל האפשר. קבוצות יכולות לחפות וצריך לכסות את כל הריבועים. הטבלה היא מעגלית ולכן ניתן לבצע חצחות את הקצה מימין ולהמשיך ממשאל.

דוגמה עבור $F(x, y, z) = \sum(2, 3, 4, 5)$ (סכום מכפלות עם אינדקסים). מפת קרנו שלנו היא הבאה, כאשר הגענו אליה או באמצעות השבלונה או באופן הבא: הסימונים של x, y, z אומרים לנו Aiife המשטנה מקבל ערך 1, ולכן במשבצת השנייה מימין בשורה העליונה לדוגמה, גם y וגם z הם 1 אבל x הוא 0, כלומר מדובר במקרה של $(0, 1, 1)$ שבמקרה שלו זה 1 (כי זהו ערכו של המinterm השלישי שמהגדרת הפ' הוא 1).

			z		
		00	01	11	10
$x \backslash y$	0	0	0	1	1
$x \backslash y$	1	1	1	0	0
				y	

כדי למצמצם את הטרבלה עכשו נcosa את הטרבלה עם שני מלבים, אחד משמאלי למיטה ואחד מימין למעלה וכן נקבל ייצוג מינימלי של הפ' הבוליאנית שהוא $F(x, y, z) = x'y + xy'$ (מלבן משמאלי משותף הכל חוץ מ- z וכן גם גס מלבן מימין).

דוגמה ($F(x, y, z) = \sum(0, 2, 4, 5, 6)$). הביטוי הלא מצומצם מכיל 5 ביטויים. נמלא אייכשו את הטרבלה ונקבל

		z			
		00	01	11	10
$x \backslash y$	0	1	0	0	1
$x \backslash y$	1	1	1	0	1
				y	

נשתמש בחפיפות וגם במעגליות ונקבל מלבן אחד משמאלי למיטה ועוד מלבן שכולל את העמודה השמאלית והעמודה הימנית יחד, ואז הביטוי המינימלי הוא $F(x, y, z) = xy' + z' + y$ (בשמאלי למיטה נופל z ובמעגלי נופלים x ו- y , פשוט עוברים על זוג ריבועים אופקי וזוג אנכי ורואים מה משתנה בהם, ומה משותף בהם).

מפת קרנו באربعة משתנים נראה כך, כאשר אפשר לזכור אותה באמצעות העובדה שערכי הצירים שלה (גם אופקיים וגם אנכיים) הם 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 (היצוג הבינארי של המספרים).

		z			
		00	01	11	10
$w \backslash x$	00	0000	0001	0011	0010
$w \backslash x$	01	0100	0101	0111	0110
$w \backslash x$	11	1100	1101	1111	1110
$w \backslash x$	10	1000	1001	1011	1010

דוגמה ($F(x, y, z) = (0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$) מתקבלת מפת קרנו עם הערכים הבאים (לא משנה איך מגיעים לזה)

		00	01	11	10	y
		wx	yz	w	x	
w		00	01	11	10	
w	00	1	1		1	
	01	1	1		1	
	11	1	1		1	
	10	1	1		1	
z						

נבחר את חצי המפה השמאלית כי זו שמנינה והמלבן הגדול ביותר שיש, ובשביל הערכים מימין נבחר שתי רכיביות מעגליות (אי אפשר את כולם ביחד כי זה נותן לא חזקה של 2).

החצי השמאלי נבדל ב- w , z , $-x$, ו- y מקבל 0 כלומר הביטוי הראשון הוא y' .

הרכיביה המעגלית העליונה נבדلت ב- $-x$ ו- y ו- w , z מקבלים ערכים 0 שניהם, כלומר יש לנו $w'z'$ והרכיביה המעגלית התחתונה שונה ב- w ו- y ו- x ו- z מקבלים ערכים 1 ו-0 בהתאם, כלומר הביטוי השני הוא xz' .

$$\text{סה"כ קיבלנו } F(x, y, z, w) = y' + z'w' + xz'$$

הأدירה לפעמים עבור צירופים מסוימים, לא יהיה אכפת לנו מה הוא פלט הפ', צירופים כאלה נקראים **צירופים אדישים** ונitin להשתמש בערך שיותר נוח אליו כך שיבוטלו ליטרלים רבים ככל האפשר. נסמן צירוף כזה ב- \emptyset .

דוגמא ((1, 1, 1)) $F(x, y, z)$ היא הפ' הבוליאנית והצירופים האדישים הם (7) (כלומר רק).

		00	01	11	10	z
		x	yz	w	x	
w		0	1	0	0	1
w	00	1	1		1	
	01	1	1		1	
	11	1	1		1	
	10	1	1		1	
y						

עתה נוכל לבחור מלבנים יותר נוחים (ראו או איור) מאשר בהיעדר הצירוף האדיש כי אז לא היינו יכולים לבחור את השורה התחתונה כמעט והיו לנו אמנים עדין שני ביוטוים אבל עם יותר ליטרלים, כלומר יותר שערירים שזה פחות טוב.

הערה צירוף אדיש מתקבל לדוגמה כsharpכיב אלקטронאי איזשהו מעגל בכל מקרה מחובר להארקה כך שלא משנה ערכו עבור צירוף מסוים כלשהו.

דוגמא נביט בModelProperty הקרה הבאה עם שני צירופים אדישים. הבחירה הכי נוחה היא 0 לשמאלי ו-1 לימני כי כל קומבינציה אחרת הייתה דורשת מאייתנו יותר משני מלבנים או מלבנים קטנים יותר (יותר ליטרלים)* שזה פחות אידיאלי.

		00	01	11	10	y
		wx	yz	w	x	
w		00	01	11	10	
w	00	1	1		1	
	01	1	1		1	
	11	1	1		1	
	10	1	1		1	
z						

דוגמה אפשר להשתמש בມפת קרנו גם כדי לצלצלים מכפלה של סכומים. נביט ב-(5) $F(x,y,z) = \sum (0,1,2,3,4,6,7) = \Pi(5)$. כפי שניתן למטה, צמצום של סכום המכפלות דרוש לפחות 3 נסכומים ואילו מכפלה סכומים דרוש בדיקת ייטרל אחד.

	00	01	11	10
x	1	1	1	1
y	1	0	1	1
z	0	0	1	1

כשמצטאים פ' בוליאנית לפי מכפלה סכומים, מבצעים בדיקת התהיליך של סכום מכפלות רק שמשכים 0-ים במקומות 1-ים. לאחר הcisioi, ממיררים את הcisioi למכפלה של סכומים, כאשר כל סכום מתאים למילון אחד.

הערה ניתן להוכיח נכונות של צמצום לפי מפת קרנו למכפלה סכומים או באמצעות דה-מורגן לצמצום של סכום מכפלות, או פשוט באופן ישיר מטבלת האמת ונכונות ייצוג maxterm-ים.

דוגמה נתונה הפ' הבוליאנית

$$F(w,x,y,z) = \sum (1, 2, 3, 11, 12, 13, 15) + d(w,x,y,z), \quad d(w,x,y,z) = \sum (5, 9, 10, 14)$$

- ציירו את מפת קרנו עבור הפ' F .

המפה תראה כך

	00	01	11	10
w	0	1	1	1
x	0	∅	0	0
y	1	1	1	∅
z	0	∅	1	∅

- כמה פ' שונות מיוצגות ע"י המפה?

$2^4 = 16$ כי אפשר לבחור ערכים שרירוטיים עבור כל אחד מהצירופים האדישים שהוא ב"ת באחרים.

- רשמו סכום מכפלות מינימאי עבור F , האם הסכום ייחיד?

הכי נוח יהיה שהשורה השנייה תהיה כולה 0 והשלישית כולה 1, ובשורה הרביעית כמה שיותר 1-ים כדי שנקבלים מלבים של רביעיות במקומות זוגות. כך נקבל סה"כ את הcisioi הבא

	00	01	11	10
w	0	1	1	1
x	0	∅	0	0
y	1	1	1	∅
z	0	∅	1	∅

שנותנו לנו את הביטוי $z'x'y + x'y + x'w$. זה לא ביוטי מינימלי כי אפשר לבחור שכל הצירופים האדיישים יהיו 1 חוץ מ-(0, 0, 1, 1).

ואז קיבל את הכיסוי הבא עם אותו מספר ליטרלים

		y			
		00	01	11	10
wx		00	1	1	1
w	00	0	Ø	0	0
	01	0	Ø	1	Ø
	11	1	1	1	Ø
	10	0	Ø	1	Ø

פונקציות שלמות

הגדירה קבוצה F של פ' בוליאניות נקראת שלמה אם ניתן למשתכל פ' בוליאנית בעורף F בקבוצת F .

משפט $\{ \cdot, \cdot', +, \cdot' \}$ היא שלמה.

הוכחה: כל פ' בוליאנית ניתנת להציג כסכום מכפלות שדורש רק את שלושת הפעולות הללו.

מסקנה $\{ \cdot, \cdot', +, \cdot' \}$ הן שלמות.

הוכחה: עם דה-מורגן אפשר ליציר $+$ עם \cdot' , ול- \cdot עם \cdot .

דוגמה NAND, קלומר $(y \cdot x)'$ היא פ' שלמה. ראשית ניתן להשיג NOT (\cdot') באמצעות AND-ו- x ($x \cdot x = x'$). לכן ניתן להשיג באמצעות NOT על NAND, שניהם כבר יש לנו.

דוגמה הוכיחו כי $f(x, y, z) = x' + yz$ והפ' הקבועות 0, 1 הן קבוצה שלמה.

ראשית ל-NOT ניתן להגיע באמצעות $x' = f(x, 0, 0)$. f(x, 0, 0) = $x' + 0 \cdot 0 = x' + 0 = x'$. f(x, 0, 0) אפשר להגיע ע"י $f(1, x, y) = f(x, 0, 0)$. לכן ניתן ליציר קבוצה שלמה כלומר הקבוצה המקורית היא שלמה. אפשר גם להגיע ל-OR ע"י $f(f(x, 0, 0), y, 1) = (x')' + y \cdot 1 = x + y = x + y$.

למעשה לא צריך את שני הקבועים כי ברגע שיש NOT עם אחד הקבועים אפשר להגיע לקבוע אחר עם NOT על הקבוע שכן יש לו.

שבוע | IIIII

הרצאה

דוגמה נתונה הפ' עם טבלת האמת הבאה

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

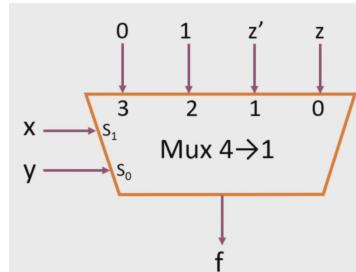
- נממש את הפ' עם 8 MUX → 1 יחיד.

כל מה שצרכי לעשות זה לחבר את z ל- $s_0, s_1, s_2, \dots, s_7$, בהתאם ולהציג בקלטים x_0, x_1, \dots, x_7 של ה-MUX את ערכיו f בטבלת האמת לפיה הסדר. כך נבחר עבור כל צירוף (x, y, z) בדיקות התוצאה מתוך ערכי טבלת האמת של f .

- עתה נממש עם 4 MUX → 1 יחיד ועוד שער אחד בלבד. כאן צריך יותר להתחמץ. ראשית נביט בטבלה כאשר (x, y) מופרדים מ- z , כך שלכל (x, y) יש שני צירופים עם z עם ערכים אפשריים.

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

בנייה MUX יחיד עם סלקטורים y, x ובקלט ה- z - נשים או קבועים שני הצירופים עם z נתונים את אותו ערך, או z או z' בהתאם לערך השערק משטנה (שכנוו עצמכם שאכן אללו כל האפשרויות). כך קיבל את השער הבא



אנחנו מקיימים את הדרישות כי יש MUX אחד ושער אחד שהוא NOT על הקלט השני ל-MUX.

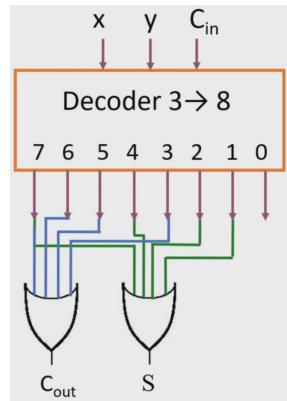
דוגמה נממש FA (שמקבל S, C_{out} ופולט x, y, C_{in}) שיש לו את טבלת האמת הבאה (לצורך נוחות) עם :

x	y	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

• $8 \rightarrow 3$ OR ושערי.

נבחר קלטים למפענה x, y, C_{in} , ואז על הפלטים נצמיד שער OR אחד לפט S ואחד לפט C_{out} . מה"כ זה יראה כך (יראה ב-

(1, 1, 1) מוציא חוט לשני ה-OR-ים, בהתאם לטבלת האמת)



• $8 \rightarrow 1$ MUX .

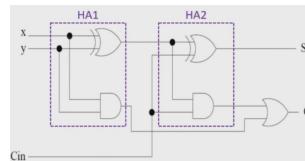
נבחר את הסלקטורים להיות x, y, C_{in} ונשבץ בשמות הקלטים של ה-MUX. מUX את ערכי S בטבלת האמת לכל צירוף של הסלקטורים

(הקלטום), והואתו הדבר שוב עם C_{out} .

• $4 \rightarrow 1$ MUX .

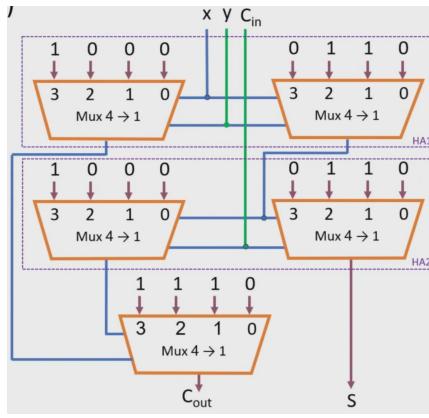
זה כבר יותר מרכיב, והורש פירוק של FA לשני Half Adder-ים שלא הזכירנו כאן, אבל הם שערים שמקבלים x, y ופלטים

. מימוש די פשוט ניתן לראות בתוך המלבן XOR הקלטום והנשא הוא AND על הקלטום.



כך נוכל לחבר יחד שניים כאלה כדי לחשב $C_{in} = (x + y) + C_{in}$.

עם $4 \rightarrow 1$ MUX (משבצים את ערכי טבלת האמת כאמור) וכל שנותר הוא להרכיב שני HA לאחד יותר גדול (ראו איור)

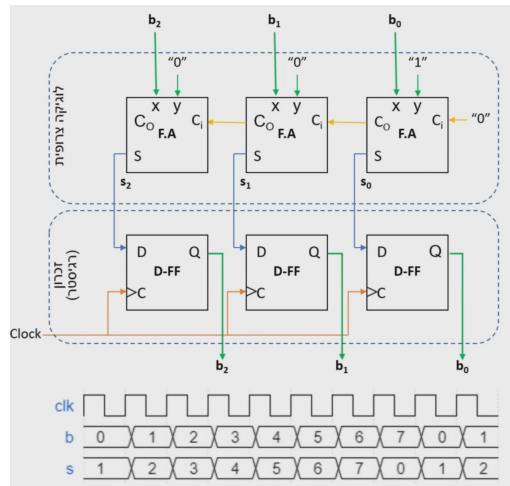


לאחר ראיינו DFF, שמקבל קלט D ושעון ומשנה את פלטו בעת עליית השעון לערך של D (זו וריאצית ה-Edge (Positive Edge)) נוכל על בסיס ייחידה זו לבנות יחידות יותר מורכבות.

דוגמה Toggle FF מקבל T ושעון ומחשב בכל עלייה שעון $Q(t+1) = \text{XOR}(T, Q(t))$ יחד עם T אל תוך XOR $Q(t+1)' = Q(t)$ אם $T = 0$ ו- $Q(t+1)' = T$ אחרת.

דוגמה JK-FF מקבל J, K ושעון ומחשב בכל עלייה שעון $Q(t+1) = JQ'(t) + K'Q(t)$ יחד עם $JQ'(t) + K'Q(t)$ אל תוך JK-FF. מזונת- D של ה-FF הפנימי.

דוגמה נממש ספונ, שסופר את מספר עליות השעון.



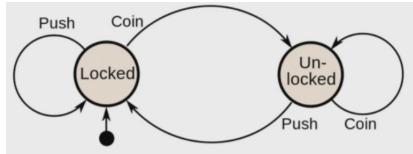
הרעיוון כאן הוא פשוט אבל המימוש לא כל כך: בכל עלייה שעון, ה-FA הימני ביותר מכניס עוד ערך 1 לסכום שמוחזק ע"י כל המעלג. הסכום מופיע בכל עלייה שעון ל-DFF הבא (זה-FA המתאים לו), וכך ה-FFים מחזקים שלושה ביטים שמיצגים מספר שערכו עליה באחד בכל עלייה שעון (הסתודנטית המשקיעה תרים את שלושת המוחזרים בראש/על נייר ותראה שהוא אכן עובד).

Finite State Machine

הגדרה FSM הוא מודל חישובי עם מספר סופי של מצבים, מצב התחלתי, מספר סופי של קלטים ופלטים, פ' מעברים $\rightarrow \{\text{קלטים}\} \times \{\text{מצבים}\}$ ופ' פלט.

דוגמה שער מסטובב (Turnstile) יכול להיות פתוח או סגור, אם הוא מקבל מטבע והוא נסגור, אם הוא לא נסגור ונדחף, הוא נסעל וכוכ'ן. כן המצביעים הם "פתוח" ו"נסע", הקלטים הם מטבע ודחיפה, הפלט עבורו "פתוח" הוא "אפשר לעبور" ובהתאמה עבורו "נסע".

ונוכל לצייר את ה-**FSM** עם גראף

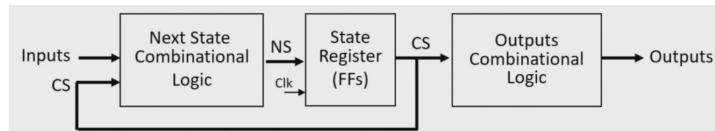


או טבלה (יש כמה דרכים)

State	Output	Input	Next State
Locked (init)	Closed-pass	Coin	Unlocked
	Push		Locked
Unlocked	Open-pass	Coin	Unlocked
	Push		Locked

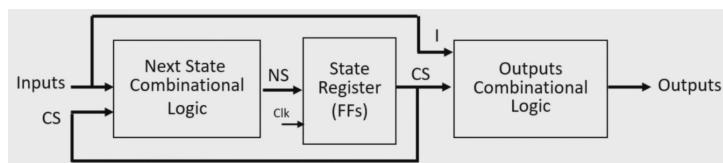
State	Output	Inputs	
		Coin	Push
Locked (Init)	Closed pass	Unlocked	Locked
Unlocked	Open pass	Unlocked	Locked

דוגמה מכונה כבאיור (Moore) היא מכונה שבה מצב הבאה והנוכחי בהתאם, שהיא שמייחד אותה הוא שהפלטים תלויים אך ורק במצב הנוכחי.



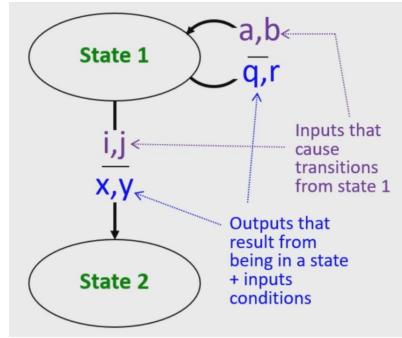
נשים לב שהרגיסטר שמורכב מ-FF-ים מחזיק את המצב הנוכחי, וערכו מחוות חזרה פנימה לחישוב המצב הבא שיכניס לרגיסטר במחזור הבא.

דוגמה מכונה Mealy היא מכונה שבה הפלטים תלויים גם במצב וגם בקלטים, והארQUITטורה שלו היא כבאיור



הערה את הייצוג של הגרפי של מכונת Moore מצירירים כמו אוטומט רגיל, רק שם כל מצב (מעגל) מכיל גם את הפלטים שהוא משרת. את הייצוג הגרפי של מכונת Mealy מצירירים כמו אוטומט רגיל, רק שעל החיצים (המעברים) נוסיף את הפלטים שהקלטים על החץ יחד עם המצב שעוברים אליו משרים (ראו איור).

הערה הפלטים יושפעו מהקלט מהר יותר ב-**Mealy** כי הם מחוברים ישירות לפ' הפלטים, בעוד ב-**Moore** נדרש עלייה שעון כדי שישתנה המצב המשרת פלט.



הערה אפשר לכתוב מכונות Mealy ששוולות למכונות Moore עם פחרות מלבנים, אבל החסרונו הוא ש-Mealy גורם לביעיות עם תזמנונים (שנלמד בהמשך).

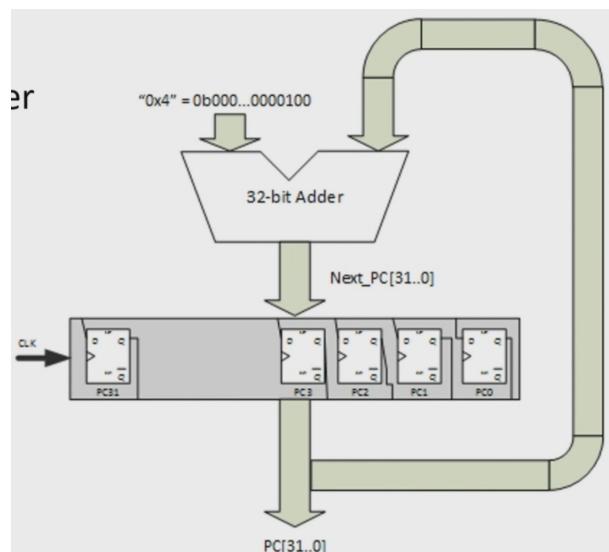
דוגמה נזכיר לדוגמת השער המסתובב. נגידר 0 השער נעל-ו-1 הוא פתוח. לכן המעגל שלפ' הפלט הוא חוט ישיר מהמצב לפלט כי הפלט זהה במצב. את המעברים ניתן לראות בטבלת האמת הבאה

Current State	Coin	Push	Next State
0	0	X	0
0	1	X	1
1	X	0	1
1	X	1	0

והמיוש של המעגל למעבר למצב הבא הוא די פשוט: אם Q הוא המצב הנוכחי ו- D המצב הבא (הקלט ל-DFF) אז $D = \text{Coin} \cdot Q + \text{Push}' \cdot Q'$ שזה ניתן למימוש עם שערים מאוד פשוטים.

דוגמה Program Counter נתונים למעבד בכל פעם את הכתובת בזיכרון ממנה צריך לקרוא את הפקודה הבאה. הספרן פולט כתובות באורך 32 ביטים שkopatzת בקייזות של 4 (אלא אם הייתה קפיצה למקום אחר) בגלל שככל מילה היא באורך 32 ביטים (בית הוא 8 ביט).

המיוש הוא די פשוט: נחזק 32-DFFים שיחזקו את הכתובת, נחוות יישירות את ה-DFF ל-32 הפלטים והמעגל לחישוב המצב הבא הוא FA עם FA $y = Q \cdot x + Q' \cdot 1$ (Q המצב הנוכחי), או באior ברזולוציה נמוכה משום מה זה יראה כך



זהה מכונת Moore, שערכה מתעדכן פעם אחת בכל מחזור.

תזמון מעבד

הגדרה התדר של מעבד הוא מספר המוחזורים של השעון בשנייה (ביחידות Hz).

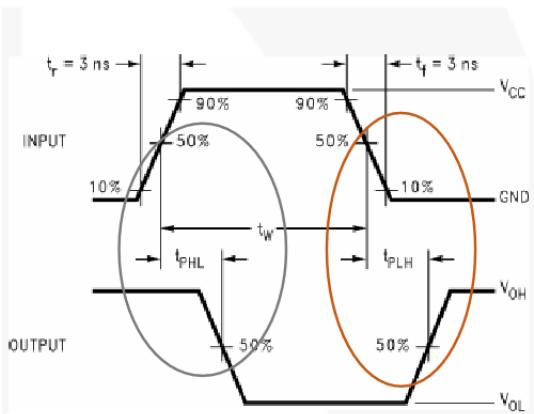
טעינה ופריקה של מטען לוקחים זמן ולכון מעבר של זרם בתוך טרנזיסטור אינם מיידיים (אלא מאוד מהירים). פרק הזמן זהה נקרא Propagation Delay.

פרמטרים של זמן פעוף

- זמן הפעוף מנמוך לגובה (t_{PLH}) : זמן הפעוף כשהפלט עובר מנמוך (0) לגובה (1). מודדים אותו החל משינוי ניכר בקלט ועד לעליית הפלט ל-50% מתח. בפועל מתח נחسب גובה (ומתפרש כ-1) רק כשהוא 90% ומעלה מערכו המקסימלי, וכך יש הנחה סבירה שהעליה והירידה של המתח הם מאד מהירים ולכון מ-50% ל-90% אין הבדל משמעותי.
- זמן הפעוף מגובה לנמוך (t_{PHL}) : זמן הפעוף כשהפלט עובר מגובה לנמוך, מחושב ע"י הפרש הזמנים בין שינוי ניכר בקלט ועד לירידת הפלט למתחת ל-50%.
- זמן עלייה (t_r , Rise) : הזמן שלוקח לעלות מ-10% מתח ל-90% מתח.
- זמן ירידת (t_f , Fall) : הזמן שלוקח לרדת מ-90% ל-10% מתח.
- זמן פעוף (t_{pd}) : כש- $t_{pd} = t_{PHL}$, נקרא להם .

הערה t_r, t_f הם מאוד קטנים (ננו-שניות).

דוגמה בשער NOT כלשהו מקבלים את הגירף הבא של הפלט כתלות בקלט.

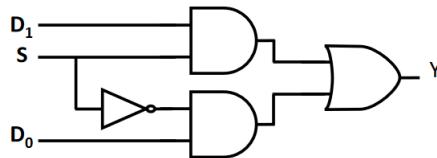


נשים לב שהשינוי הוא לא מיידי. במקרה הזה השינוי הניכר בקלט הוא חצייתו (מלמעלה או מלמטה) של הקלט את רף 50% המתח.

הגדירה עבור t_{pd} יש ערך טיפוסי, ערך מקסימלי (t_{pd_max}) שאומר אחרי כמה זמן בטוח הפלטים כבר ישתנו וערך מינימלי (t_{pd_min}) שمبתייח מתחת לאיזה רף בטוח הערכים לא ישתנו.

הערה $t_{pd_min/max}$ יכולים להיות שונים עבור שינוי בקלטים שונים (קלט x משפיע יותר מהר מאשר y).

דוגמא נתון השער X שmmaומש באופן הבא



וחסמי זמן פעוף לשערים

t_{pd_max}	t_{pd_min}	שער
5ns	2ns	NOT
8ns	4ns	AND
10ns	5ns	OR

- מהו t_{pd_max} של השער כולם?

עבור ההשפעה $Y \rightarrow D_1 \rightarrow Y$ (כמה זמן לאחר שינוי D_1 ישתנה) נctrיך לעבור דרך AND ו-OR כלומר סה"כ $18ns$, וכך גם עבור

$.D_0$

עבור $Y \rightarrow S \rightarrow Y$ זמן הפעוף המקסימלי הוא $\max(AND + OR, NOT + AND + OR) = \max(18, 23) = 23$ ns.

זמן הפעוף של השער כולם הוא המינימום של כל המסלולים, כלומר $23ns$.

- מהו t_{pd_min} של השער כולם?

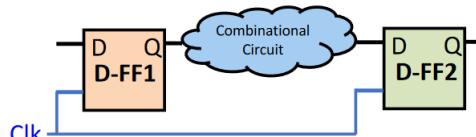
עתה נבחר את המסלול הקצר ביותר, שהוא כמובן $D_0 \rightarrow Y \rightarrow D_1 \rightarrow Y$ ($t_{pd_min} = 4ns + 5ns = 9ns$ וזו)

הגדירה זמן הפעוף של $D \rightarrow Q$ מחושב (מודגר) החל מעלייה של השעון ל-50% ועד לשינוי Q והוא נקרא t_v (או t_{PCQ}) והוא למעשה מגדי אחרי כמה זמן לאחר עליית השעון נוכל להчисיב את הקלט כחוקי. t_{v_min} מבטיח עד מתי Q ישאר בערכו הקודם ו- t_{v_max} מבטיח החל ממתי יהיה הערך החדש.

הגדירה כדי Q יהיה תקין, D צריך להיות יציב ולא להשתנות במשך פרק זמן לפני עליית השעון, זהו ($Setup$), וגם לאחר עליית השעון, זהו ($Hold$) t_h .

הערה $t_s > 0$ כי בתוך ה-DFF-ה Master-slave משנה את ערכו קצר לפני ה-Slave ולכן הערך שם צריך לא להשתנות כדי של-Slave יהיה את הערך הנכון.

דוגמא נביט בكونסטרוקציה הבאה



נניח שזמן מחזור השעון הוא t_{cyc} (זמן בין עליית שעון אחת לשניה), לכן קצב השעון הוא $f = \frac{1}{t_{cyc}}$. נניח כי זמן הפעוף המקסימלי של השער הוא t_{pd-max} . מהו זמן המחזור המינימלי כדי שהמעגל יהיה תקין, כלומר כדי שההתוצאה תגיע ממהקלט ל-DFF1 עד לפולט של 2 תוך שני מחזורים (בראשו הקלטים עוברים את השער ובשני הם כבר מופיעים בצד השני)?

- זמן המחזור צריך לפחות t_{cyc} .

$$t_{cyc} \geq t_{v-max} + t_{pd-max} + t_{setup}$$

כדי שיהיה קודם לחכות שהפלט של DFF1 יהיה חוקי (t_{pd-max}), אז עלת לו לעבור את כל השער (t_{v-max}) ואז שהפלט יהיה ייחד מספיק זמן לפני עליית השעון הבאה כדי שייעבור בהצלחה ל-Q של DFF2 לאחר העליה.

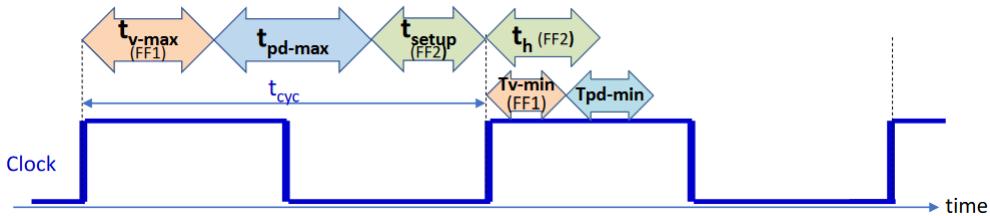
- מה צריך להיות t_h כדי שהשער יהיה תקין?

בפעם השנייה שבה ישנה הערך, נדרש ש- t_h של DFF2 יהיה פחות מזמן שлокח ל-D של DFF2 להשתנות בהשפעת ה- Q . החדש של DFF1. כלומר, נדרש ש- $t_h \leq t_{v-min} + t_{pd-min}$

$$t_h \leq t_{v-min} + t_{pd-min}$$

כדי הזמן שлокח לפחות לעבור מ- D (DFF2) ל- Q (DFF1) חסום מלמטה ע"י הזמן המינימלי שעבורו (t_{pd-min}) לא ישנה לאחר עליית השעון (t_{v-min}) ועוד הזמן המינימלי שעבורו (t_h) לא יכול לקבל ערך חדש כפלט של המעגל (t_{pd-min}).

סה"כ מחלק התזומנים כפ' של השעון הוא באירוע



הערה אם אין לנו דרך לשלוט ב- t_h ונרצה עדין מעגל חוקי, אפשר לחיבר את t_{pd-min} להיות יותר גדול ע"י הוספה שני NOT-ים למסלול הקצר ביותר במעגל (הוא לרוב לא הארוך ביותר) וכך לא להשפיע על התוצאות אבל כן על התזמון המינימלי.

דוגמא נתונה הקונסטרוקציה הבאה עם MUX, שלו (הכלביות ns , $t_{v-min} = 2, t_{v-max} = 10, t_{pd-min} = 9, t_{pd-max} = 23$) ו- D -MUX $t_s = 7, t_h = 5$

- מהו זמן המחזור המינימלי האפשרי?

כדי שהמעגל יהיה תקין, נדרש שמסלול הפעוף הארוך ביותר במבנה יהיה כולל מוכל במחזור אחד. המסלול הארוך ביותר הוא משינויי ב- $(S0, Q)$, דרך חישוב ה-MUX ועד לשינוי הערך ב- D ,ऋציך לקחת בחשבון שלפני תחילת המחזור הבא נדרש ש- D יהיה יציב ל- t_{setup} ננו-שניות. סה"כ נדרש שיתקיים (בדומה לדוגמה הקודמת)

$$t_{cyc} \geq t_{v(max)} + t_{pd(max)} + t_{setup} = 7 + 23 + 3 = 33$$

- מהי הדרישה על t_h כדי לקבל מבנה תקין?

נוצר שערך של D לא ישנה מוקדם מדי לאחר עליית השעון, בפרט שזמן שינוי הערך Q וחישוב ה-MUX יקחו יותר מאשר

$$(t_h = 5 \leq 9 \text{ ו } t_h \leq t_{v(min)} + t_{pd(min)} = 2 + 9) \text{ כלומר } t_h = 9$$

$$\text{סיה"כ} F_{max} = \frac{1}{33ns} = 30MHz \text{ ולכן } T_{cyc(min)} = 33ns$$

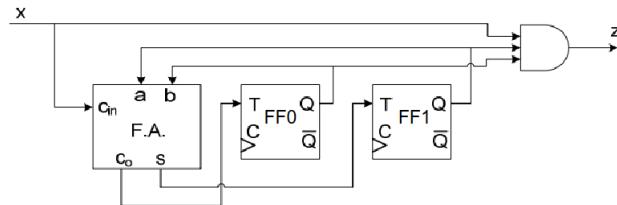
עד כה הטענו מהקלט האמיתי של המודול שמחובר ל-S1. הוא עצמו גם חייב לקיים דרישת זמן לשעון וכך נוכל להתייחס אליו כסיגナル שמסוגל מפלט אחר והניתוח יהיה כן".

כל קלט למעגל אחד הוא פלט של מעגל אחר, וכך יכולם להיות להם ערכי t_v, t_{v-min} שונים ממודול אחר.

כל פלט למעגל כלשהו הוא קלט למעגל אחר וכן הפלטים צריכים לעמוד בדרישות t_h, t_s מסוימות גם כן.

הערה כשנתח זמן של מעגל, נסתכל על כל המסלולים שמתחלים בכניסה לדFF ונגם

דוגמה נסתכל על המבנה הבא, כשהנתון $t_{setup} = 9, t_{hold} = 0$ (ז' נדרש להיות יציב לפחות 9ns לפני עליית השעון, ו-0ns אחרי)



עם הנתונים

Parameter	t_{pd-max}	t_{pd-min}	t_{setup}	t_{hold}	t_v	t_{v-min}
AND 3 inputs	3	1				
FA [a,b,Cin]->S	12	3				
FA [a,b,Cin]->Co	8	3				
T-FF			7	2	4	0

• מהו $T_{cyc-min}$? נתח את כל המסלולים שנגמרים בפלט (כי לפלט יש דרישות ביחס לשעון, בפרט לדFF יש

יש צזה שנדרש מאייתנו)

– מסלולים שנגמרים ב-z : המסלול המקסימלי הוא באורך

$$t_{setup} + t_{pd}^{AND} + \max \{t_v^{FF1}, t_v^{FF0}, t_v^x\} = 9 + 3 + \max \{4, 4, 15\} = 27ns$$

כדי שהזמן יהיה יציב זמן לפני עליית השעון, ערכו מחושב ע"י AND או מוסיפים את זמן הפעוע דרכו, וקלט

ה-AND הם פלטי FF1, FF0 ו-x בתאמה, שערך מתעדכן למחזור הנוכחי לאחר ה- t_v של כל אחד מהם (כאן אנחנו

$$(t_v = t_{v(max)}) \text{ מיסמנים}$$

– מסלולים שנגמרים ב- T : המסלול המקסימלי הוא באורך

$$t_{\text{setup}}^{\text{FF1}} + t_{\text{pd}(\rightarrow S)}^{\text{FA}} + \max \{t_v^{\text{FF1}}, t_v^{\text{FF0}}, x_{\text{valid}}\} = 7 + 12 + \max \{4, 4, 15\} = 34ns$$

– מסלולים שנגמורים ב- T : המסלול המינימלי הוא באורך

$$t_{\text{setup}}^{\text{FF0}} + t_{\text{pd}(\rightarrow C_0)}^{\text{FA}} + \max \{t_v^{\text{FF1}}, t_v^{\text{FF0}}, x_{\text{valid}}\} = 7 + 8 + \max \{4, 4, 15\} = 30ns$$

ולכן סה"כ נצרך $T_{\text{cyc-min}} \geq \max \{27, 34, 30\} = 34ns$

- האם מתקיימת הדרישה על Min Delay .

כן! עבור z מתקיים $t_h^z = 0ns$ והזמן המינימלי לפעוף הוא

$$t_{\text{pd}(min)}^{\text{AND}} + \max \{t_{v(min)}^x, t_{v(min)}^{\text{FF}}\} = 1 + \min \{0, 0\} = 1ns$$

כלומר מתקיימת הדרישה ועבור $t_h^{\text{FF}} = 2ns$ FF0, FF1 יש Min Delay .

$$t_{\text{fpd}(min)}^{\text{FA}} + \min \{t_{v(min)}^{\text{FF}}, t_{v(min)}^x\} = 3ns$$

(כי הערך החדש צריך לעבור דרך FA ולהגיע או משינוי ב- x או משינוי ב- y של אחד ה-FF-ים) ולכן מתקיימת הדרישה גם כן.

תרגול

הגדלה מעגל צירופי (קומבינטורית) הוא מעגל שיש לו כניסה ויציאה כאשר האחרונות תלויות בכל ערך ורגע של הראשונות. מעגל סדרתי הוא מעגל שפלטיו תלויים גם ביחידת זכרון שמחוברת לשעון.

דוגמה כיצד נממש **Full Adder** (כזכור קלטים S, C_{out} ופלטים x, y, C_{in} (מיימן))?

x	y	C_{in}	00	01	11	10
0	0	0	0	1	0	1
0	1	0	1	0	1	0
1	0	1	0	1	0	0
1	1	1	1	1	1	0

x	y	C_{in}	00	01	11	10
0	0	0	0	0	1	0
0	1	0	1	1	1	1
1	0	1	1	1	1	1
1	1	1	1	1	1	1

ולכן אפשר למש את S עם OR על ארבעה פלטי AND (שכלולים קלטים שעוברים דרך מהפץ) ואת C_{out} אפשר קצת יותר עיל. ראיינו בהרצאה שהמיימוש של FA כולל בתוכו שני HA.

הגדרה מוחסרים מחשבים חישור ספורות בינהו. עתנ נפלוט את ההפרש (בערך מוחלט) ו- Out Borrow שיגיד לנו כמה יותר לחסר מעבר להפרש. הסטודנטית המשקיעה תמשח צי-מוחסר ומוחסר מלא.

הגדרה משווים הם מעגלים שבודקים איזה קלט יותר גדול.

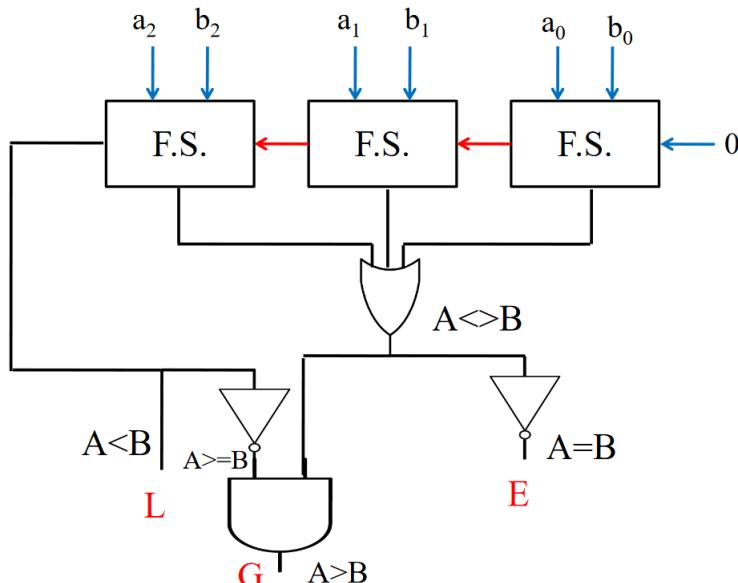
- דרך אחת למשוך זאת היא באמצעות השוואת מה-LSB ל-MSB כאשר בפעם הראשונה שיש אי-שוויון בביטים נבחר את האחד

שקלטו גדול יותר (1 לעומת 0).

- דרך אחרת היא באמצעות מוחסרים : $A > B \iff A - B > 0$ וכו'.

דוגמא נתונים הקלטים G, E, L ממשו עם מוחסרים מעגל שפלט ביטים $A = a_2a_1a_0, B = b_2b_1b_0$ שערך כל אחד מהם אם "1" ממש את המעגל כבאיור $B, A = B, A < E$.

בהתאם.



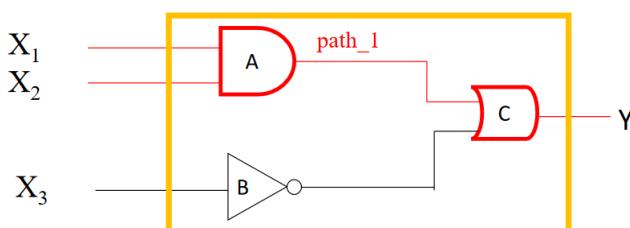
אם $a_2 < b_2$ בהכרח ישאר לנו Borrow Out כי אם BO מ לפני שימוש הלהה ואחרות $a_2 = b_2$ או יהיה BO ובודאי שהוא BO.

אם כל הביטים זהים נדרש NOR על כל הפרשיים (כל החיסורים פולטים 0) ואכן זה מה שנבינו.

בשיטת האלימנציה, G הוא 1 אם $A \geq B$ וגם $A \neq B$, כלומר אם $A = B$ הוא 0 וכן אכן מופיע במעגל.

הערה השימוש ב-10%-ו-90% כרף לחישובי זמן נובע מכך שקשה לאפיין את פריקת הקבל כתהיליך לינארי בקצבות, ולכן מתעלמים מהם.

דוגמא נביט בפ' $X_1 * X_2 + X'_3$ שמנומשת באופן הבא



מתתקיים $t_{pd}(A) = \max\{t_{PLH}(A), t_{PHL}(A)\}$ ו- X_3 מ- X_1, X_2 והשני מ- Y . כלומר $t_{pd}(A) = \max\{t_{PLH}(A), t_{PHL}(A)\}$.

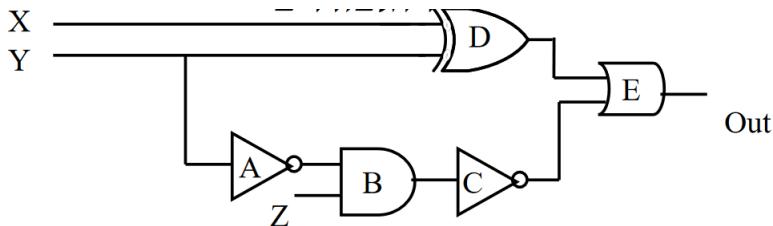
לכן (מסלול ראשון) $t_{pd}(B) + t_{pd}(C)$ ובהתאמה (מסלול שני) $t_{pd}(A) + t_{pd}(C)$ תחת הנ吐ונים הבאים (כאשר t_{cd} מלשון $t_{cd} = t_{tp-min}$)

Data in ns	X_2	A	B	C
t_{PHL}	-	100	90	80
t_{PLH}	-	110	70	100
t_{cd}	-	12	8	10
t_r	14	20	12	18
t_f	15	17	13	19

נחשב את ה- t_{pd} של המעגל כולו. עבור כל שער, וכן $t_{pd} = \max\{t_{PHL}, t_{PLH}\}$ ו- $t_{pd}^{p2} = 90 + 100 = 190ns$.

נחשב את ה- $t_{pd(min)}$ של המעגל. $t_{pd(min)} = 12 + 10 = 22ns$.

דוגמה נתוני המעגל והנת吐ונים הבאים



Data in ns	A	B	C	D	E
t_{pd}	15	25	15	60	20
t_{pd-min}	5	5	5	10	5

המסלולים של שערים מתקלטים לפטיטים הם $B \rightarrow C \rightarrow E$, $A \rightarrow B \rightarrow C \rightarrow E$, $D \rightarrow E$ ו- Z בהתאמה $t_{pd} = 80ns$ ו- $t_{pd(min)} = 15ns$.

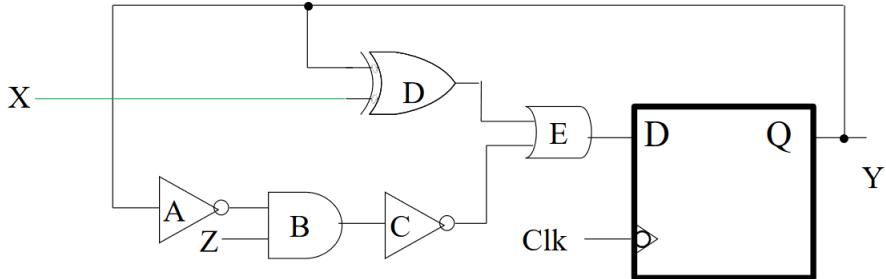
לכן $t_{pd(min)} = 15$ ו- $t_{pd} = 80$.

הערה כל עוד אין מעגלים סדרתיים, חישוב t_{pd} נעשה ע"י מקסימום הזמנים על כל המסלולים מפלטיטים לקליטים ו- $t_{pd(min)}$ ע"י מינימום.

הערה עבור כל מעגל סדרתי חיבר להתקאים $t_{hold} \leq t_{v(min)}^{FF} + t_{pd(min)}^{לגיון}$ כדי שהשינוי הכוי מהיר במעגל יקח יותר מאשר הזמן שהפלט צריך להישאר זהה אחרי עליית השעון.

הערה הקלט למעגל חיבר להטייך לאחר לכל היוטר $+ t_{setup}$ זמן כדי FF יכול לחשב באופן תקין את Q .

דוגמה נתון המעגל הסדרתי הבא עם הנת吐ונים תחתיו



Data in ns	A	B	C	D	E
t_{pd}	15	25	15	60	20
t_{pd-min}	5	5	5	10	5

כדי להשלים את הניתוח של תזמון המעגל נctrיך את האילוצים על ה-DFF. $t_{setup} = 10ns$, $t_{v(min)} = 5ns$, $t_v = 10ns$. $t_{hold} = 25ns$.

• האם המעגל תקין?

לא! חיבר להתקיים $25 = t_{hold} \leq 5 + t_{pd(min)} = 5 + t_{pd(min)}^{D \rightarrow E} = 5 + 10 + 5 = 20$ סתייה.

• כיצד נפתרת הבעיה?

נסיף דילוי על המסלול הקצר ביותר, בפרט נסיף שני NOT-ים על החוט בין Q לקלט ה

לעילו של D ושל A
. עכשו המסלול הקצר ביותר יש לו $25 = t_{hold} \leq 5 + 25 = 30 = t_{pd(min)}$ ועכשו נקבל $25 = t_{hold} \leq 5 + 25 = 30 = t_{pd(min)}$ קלומר זה כן תקין. החסרון הוא שהתדר המקסימלי האפשרי ירד כי זמן המחוור המינימלי עלה כתוצאה מהוספה שני NOT-ים.

MIPS | VII שבוע

הרצאה

הגדירה Instruction Set Architecture היא אוסף כללים שמתכונת צריך לענות להם כשהוא מפתח למעבד.

דוגמה אנחנו נלמד על MIPS, אבל במציאות פופולריים מאוד x86 ו-ARM, וגם V-RISC-Sh הוא פרויקט קוד פתוח.

הגדירה מיקרו-ארQUITקטורה היא מימוש של ISA.

דוגמה המימוש של אינטל ו-AMD ל-x86 הוא מיקרו-ארQUITקטורה.

לצורך האבstrקטציה שלנו, מעבד הוא מכונת מצבים המוחברת ל זיכרון, כאשר המცב כולל רגיסטרים שערכם משתנה על ידי פקודות. הזיכרון הוא מערך שניגשים אליו לפי אינדקס (בית אחד בכל פעם). כל מעבד מרים את התכנית הבאה:

1. קרא את הפקודה הבאה מהזיכרון בכתב שברגיסטר PC (Program Counter).

2. הוסף לרגיסטר PC את מספר הבטים שתופסת פקודה (התקדמות לפוקודה הבאה).
3. בצע את הפוקודה (וישנה את מצב המעבד).
4. חוזר לשלב 1.

בහינתן תוכנה בשפה עילית (שפתקמפלט), נתרגם את הקוד לסדרת פקודות אסמבלי באמצעות קומpileר. לאחר מכן נשתמש באסמבילר כדי לתרגם את קוד האסמביל לבינארי, שאותו המעבר כבר יודע להרץ.

הערה לעיתים הקוד שלנו ישמש בספריות חיצונית או בקבצי קוד אחרים, ועלஇיחוי כל הפקודות לקובץ אובייקט יחיד.
הפקודות באסמביל הן פקודות שהמעבד יודע לבצע (ח-ISA של המעבד), אבל לפני שהן מקודדות ל-1-ים ו-0-ים עבר המעבד.

CISC vs RISC

- Complex Instruction Set Computer זו גישה לפיה פקודות האסמביל יהיו קרובות ככל הנימן לשפה עילית, וכך להוריד את מספר הפקודות בתוכנה. בפועל זה ממומש ע"י פקודות שפותחות למיקר-פעולות ע"י החומרה. ל-ISA הזה יש הרבה מאוד פקודות, בפורמטים שונים והרכבה של פקודות שונות אחת על השניה, ואפשר אפילו להריץ פקודות ישירות על הזיכרון שמסתירות את המעבד בריגיסטרים. אורך הפקודות יכול להשנות, כאשר נקודד פקודות שכיחות באמצעות בית אחד, עד לפקודות הנדירות ביותר שהן באורך 15 בתים.

8x DSP הולכים לפי גישת CISC.

- Reduced Instruction Set Computer היא גישה לפיה יש לשמור את מספר הפקודות מצומצם וכך לפחות את פעולות החומרה, ולתת לקומpileר לעשות את העבודה הקשה של בחירת הפעולות ואופטימיזציה. הפקודות הן די פשוטות והוא רק בין רגיסטרים (ולא על הזיכרון), ורק באופן מפורש לטעון ולשמור נתונים בזיכרון. אורך הפקודות הוא קבוע.

RISC-V MIPS, ARM RISC הולכים לפי גישת

הערה המגמה עם הזמן היא לעבור מ-CISC ל-RISC משום שבמעבר קשה היה כתוב קומpileרים עילאים ולכן נדרשה המורכבות של פקודות האסמביל, ואילו עם חלוף הזמן נהיה קל ויעיל יותר לכתוב קומpileרים (בשפה עילית) שיבצעו את הפעולה המורכבת עצמאם.

דוגמה עבור העתקה של 100 ערכים בין מערך אחד לאחד ב-c, יש ב-8x פקודה אחת שמקבלת את מספר הבטים להעתקה והפוניטרים והחומרה כבר תממש את ההעתקה, לעומת זאת RISC שם צריך למשולאה שמעיטה לרוגיסטר ואז לזכור מילה מילה.

MIPS

במעבד MIPS יש 32 רגיסטרים, \$31, ..., \$0, שכל אחד מהם בגודל 32 ביט, המכונה "מילה" (4 בתים). מספר הרגיסטרים קבוע כך לאחר ניתוח של מספר המשתנים בתוכנות מדגמיות, כאשר אם יש יותר משתנים מרוגיסטרים נשתמש בזכרון הראשי כדי להחזיק את ערכם. לחלק מהרגיסטרים יש יעודיים ספציפיים, כפי שניתן לראות בטבלה הבאה

Name	Register Number	Usage	Preserve on call?
\$zero	0	constant 0 (hardware)	n.a.
\$at	1	reserved for assembler	n.a.
\$v0 - \$v1	2-3	returned values	no
\$a0 - \$a3	4-7	arguments	yes
\$t0 - \$t7	8-15	temporaries	no
\$s0 - \$s7	16-23	saved values	yes
\$t8 - \$t9	24-25	temporaries	no
\$gp	28	global pointer	yes
\$sp	29	stack pointer	yes
\$fp	30	frame pointer	yes
\$ra	31	return addr (hardware)	yes

ניסיונות רבים מהרגיסטרים משמשים פעילות תקינה של המחשבנית (SP, FP, RA), חלק שומרם ארגומנטים וחלק משומשים לצרכים אחרים.

כל פקודה היא בגודל מילה (32 ביט), וכל פקודה מבצעת פעולה פשוטה, בין היתר פעולות אРИתמטיות וЛОגיות, גישה ל זיכרון (load, store) ופקודות ופוקודות מותניות. הפקודות שמורות בזיכרון ובכל פעם נקרא מהזיכרון את הפקודה ונರץ אותה.

פקודות אРИתמטיות

- חיבור/חיסור : נביט בפקודה `add $d, $s, $t` (כאשר שלושת הפרמטרים הם רגיסטרים), ובדומה `sub $d, $s, $t`

ניסיונות להודענו למעבד בשום מקום שאנו מנסים להשתמשים בשיטת המשלים ל-2, כי אנחנו מניחים שניים שקיים קומפלט את הפקודה יודע מהקשר שהוא סוכם רגיסטרים שיש בהם כבר ערכים מיוצגים במשלים ל-2, ואם לא אז זה באג.

$$f = (g + h) - (i + j)$$

הקומפיילר יקצח רגיסטרים למשתנים ונקבל $\$s0 = (\$s1 + \$s2) - (\$s3 + \$s4)$, ואז לתרגם השורה לשפת אסמבלי יש כמה אפשרויות, הנאייה ביותר מתווכן היא

`add $t0, $s1, $s2`

`add $t1, $s3, $s4`

`sub $s0, $t0, $t1`

אבל יש דרכים אחרות (לדוגמה לפתיחת סוגרים). במתמטיקה אמנים התוצאות שקולות, אבל כאן יכול להיות שנתקבל תוצאה אחרת בגלגול overflow-ים.

- חיבור מיידי : הפקודה `i imm + $s addi $t` (כלומר חיבור בין רגיסטר וקבוע והשמה ברגיסטר).
- כאן ומן נטו לנו במשלים ל-2 אבל בגודל 16 ביט כדי שנוכל לכלול אותו בתוך קידוד הפקודה בגודל מילה אחת, ולכן נדרש להרחיב אותו למשלים ל-2 בגודל 32 ביטים, פועלה זו נקראת Sign Extend, וניתן למשה בклות ע"י ריפוד מצד של ה-MSB עם ערך קבוע של 0 לחוביים ו-1 לשיליינים (למעשה ערך ה-MSB לפני הריפוד).

הערה לא צריך `sub` כי אפשר למשה בקלות עם `addi` כאשר ה-`imm` הוא מספר שלילי.

פעולות לוגיות

- הפקודה היא $t = s \text{ and } d$ והוא מחשבת שער AND על כל שני ביטים מתאימים $m-s$ ו- t (בו זמנית על כל הביטים) ושומרת את התוצאה ב- d .

- הזו לוגית: הפקודה $a \ll d$ (מלשון a ביטים שמאל (לכיוון ה-MSB) את הביטים של t ושומרת את התוצאה ב- d .

מימין מכנים אפסים ובנתיים מאבדים ביטים משמאלי, כאשר במקרה שמספרים מיוצגים במשלים ל-2 זה יכול לאבד את בית הסימן, ובכל מקרה נוכל לקבל overflow גם במקרה של טבעים.

- הזו אריתמטית: נשמר על בית הסימן גם לאחר ההזזה במקום להתעלם ממנו. ראו טבלה שמשווה את כל ההוצאות הקיימות ב-

.MIPS

Instruction	Operation	Description
<code>sll \$d, \$t, a</code>	<code>Shift Left Logical \$d = \$t << a</code>	Shifts a register value left by the shift amount listed in the instruction and places the result in a third register. Zeroes are shifted in.
<code>sllv \$d, \$t, \$s</code>	<code>Shift Left Logical \$d = \$t << \$s</code>	Shifts a register value left by the value in a second register and places the result in a third register. Zeroes are shifted in.
<code>sra \$d, \$t, a</code>	<code>Shift Right Arithmetic \$d = \$t >> a</code>	Shifts a register value right by the shift amount (shamt) and places the value in the destination register. The sign bit is shifted in.
<code>srav \$d, \$t, \$s</code>	<code>Shift Right Arithmetic \$d = \$t >> \$s</code>	Shifts a register value right by the value in a second register and places the value in the destination register. The sign bit is shifted in.
<code>srl \$d, \$t, a</code>	<code>Shift Right Logical \$d = \$t >>> a</code>	Shifts a register value right by the shift amount (shamt) and places the value in the destination register. Zeroes are shifted in.
<code>srlv \$d, \$t, \$s</code>	<code>Shift Right Logical \$d = \$t >>> \$s</code>	Shifts a register value right by the amount specified in \$s and places the value in the destination register. Zeroes are shifted in.

הזו אריתמטית של בית אחד שמאליה פרושה הכפלה ב-2 (עד כדי overflow) וימינה פרושה חלוקה ב-2 (עם עיגול כלפי מטה).

פעולות זכרון

הזיכרון הוא מערך, שניגשים אליו באמצעות כתובות, כאשר כל כתובה מביאה לבית אחד של מידע, למרות שבפועל ב-MIPS נקרא ונכתב ביחידות של מילה (ארבעה בתים) מיושרות (כלומר כתובות שמתחלקות ב-4).

- קראת מילה: $(s_i t_i) = \text{כתובת} + imm$ והוא ה-*i*-הbitimm (immediate) ושומרת אותה ב- t .

דוגמה $0($a0), 0($t1), 1($t2)$ קוראת מילה מהכתובת ששמורה ב- $a0$ וככתובת אותו לרגיסטר t .

- כתיבת מילה $(s_i t_i) = \text{כתובת} + imm$ לשומרה בכתובת t בזיכרון.

דוגמה יש לנו מערך A עם שלושה ערכים (מספרים, בגודל ארבעה בתים), שכתובת הבסיס שלו שמורה ב- $s3$. נוכל לגשת אל האיברים במערך באמצעות $0($s3), 4($s3), 8($s3)$.

דוגמה נניח שיש לנו את הפקודות ב-C

```
int A[100];
A[12] = h + A[8]
```

הקוד הזה יתרגם באסמבלי ל-

```

lw $t0, 32($s3) # load A[8] into a temporary register
add $t0, $s2, $t0 # add h to the temporary register
sw $t0, 48($s3) # save the result in A[12]

```

פילוסופיות ארגון זכרון

- ארכיטקטורת ואן-ניומן : זכרון אחד לפקודות התוכנה וגם לששתני התוכנה. כש庫ראים מהזיכרון, משמעות התוכן (פקודה או מידע) תלויה בפעולת השובילה לקריאה. אפ"פ שתיתכן הפרדה פיזית בין החלקים, לוגית הם ממופים לאותו המקום.

יתרונות אזור זכרון מאוחד, וכל לדבג תוכנות ולשנות את אופן הטעינה.

חסרונות קריאת פקודות וקריאת מידע קוראות על אותו המשאב.

ממומש ב-**x86, MIPS, ARM**.

- ארכיטקטורת הארוורד : הזכרון של הקוד מופרד מהזיכרון של המידע, כך ש-**lw** קורא רק מידע ו-**fetch** לפקודות קורא רק פקודות.

יתרונות אין גישה במקביל למשאים שונים על אותו הפס - ביצועים יותר טובים.

חסרונות חוסר גמישות בגודל הסגמנטים של קוד לעומת דאטא וקשה יותר לעורוך את הקוד לדיבוג.

ממומש בעיקר ב-**DSP**.

הערה גם בוואן-ניומן המימוש בפועל יכול להיות באמצעות רכיבים פיזיים שונים, הגישה תהיה באמצעות אותו מרכיב כתובות (אמנם כתובות אחרות, אבל באותו מיפוי).

פקודות קפיצה והתנויות

- קפיצה בלתי-מוותנת : **label j** קופץ לאחר סיום הפקודה לשורת הקוד שמופיע לאחר ה-**label** (כפי שנכתב בקובץ asm-).
- קפיצה מוותנת שוויין : **beq \$s==\$t label** אם **\$s==\$t** קופץ ל-**label** אחריו ממשיכה לפקודה הבאה (ובדומה **bne** שkopatzet אם **.\$s!=\$t**).

דוגמה נתון הקוד הבא ב-**C**

```

if (i == j)
    f = g + h;
else
    f = g - h;

```

הוא יתורגם לקוד אסםבייל באופן הבא

```
bne $s0, $s1, not_eq # s0=i, s1=j  
add $v0, $s2, $s4 # f = g+ h  
j cont  
  
not_eq:  
  
sub $v0, $s2, $s4 # f = g - h  
cont:
```

- **השמה מותנת :** $\$s < imm : \$t \leftarrow \$s$ מושימה ב- $\$d$ אם $\$t < \s ואחרת 0, דוגמה i , בדומה i , אם $\$t < \s ואם $\$t \geq \s שם ב- $\$t$ אמ $\$d$, $\$s$, $\$t$, $\$s < imm$.

ואחרת 0.

IMPLEMENTATION OF CALL BY VALUE

הגדרה הקוראת היא הפ' שקוראת לפ' אחרת, הנקראת היא הפ'. הפרמטרים הם הערכים שMOVEDרים מהקוראת לנקראת, התוצאות הם הערכים שהנקראת מחזירה לקוראת וכתובות החזרה היא הכתובת בזיכרון הקוד של הפקודה שאחורי הקראת לנקראת בקוד של הקוראות.

המחסנית היא אוזר בזיכרון שתוכנה יכולה להשתמש בו, והיא משתמש שמירה של מידע לוקאלית בעט הרצת פ' באופן שמאפשר קראיה מקוונת וחזירה מקרים של פ'. מבנה הנתונים עובד בשיטת LIFO ואפשר או לדחוף (push) אלמנט בראש המחסנית, או להוציא (pop) את הערך העליון במחסנית.

העראה לעיתים אפשר לגשת גם לערכים אקראים במחסנית, ובכל מקרה ניגשים לכתובות ביחס לכתובות ראש המחסנית - Top of Stack, כאשר כל דבר מתחת אינו אלידי. זאת מושם שהמחסנית גדלה נגד כיוון הכתובות, כלומר הוספה ערך תיזי את TOS ארבעה בתים למטה.

שמירת רגיסטרים בעט קראיה וחזירה מפונקציה

הנקראת לא יודעת מה הקוראת עשויה, ורק מצפה לפרמטרים נכונים ולהוציא תוצאות נכונות. לכן אם הקוראת משתמשת ברגיסטרים בלבד, הנקראת תזרוס אותם בלי לדעת שהקוראת צריכה אותם. כדי לשמור את המצב לפני ואחרי קראיה לפ' יש מנגנון ; calling convention (הנקראת) הייתה באמצעות חישוב כשלקראה ל-g (הנקראת). רק חלק מהרגיסטרים נשמרים, וקיימים גורמים שונים :

- $\$t0-\$t9$ הם רגיסטרים זמינים ולבן באחריות f (הנקראת) לשמר אותם במקום אחר במהלך הקראיה ל-g.
- $\$s0-\$s7$ הם רגיסטרים סטטיים ולבן f מצפה שהם לא ישתנו, כך שגם g משתמש בהם היא תצטרכן לשזרז אותם לארכם בטרם קראתה כדי ש-f תתפרק כמו שצריך.

- \$a0-\$a3 משמשים העברת והחזרת ארגומנטים ולכון הקוראת צריכה לשמור אותם אם היא רוצה להשתמש בערכיהם שהוא עצמה קיבלה (או תחזיר) בתור נקראת.
- \$ra הוא הרגיסטר שמחזיק את כתובת החזרה מהפ', והוא נדרשת ע"י הפקודה jal בעת הקראה לפ' הנקרה, כך שאחריות השימוש היא על הקראת.
- \$\$sp, המצביע לראש המחסנית, שנדרש לכל הפ' על מנת לפעול באופן תקין, ולכון הנקרה נדרשת לשזור אותו בעת סיום הקראה לפ'.

את הערכים נשמר תמיד במחסנית.

- כדי לדוחו (ולשמור) את \$ra למחסנית נשימוש בפקודות

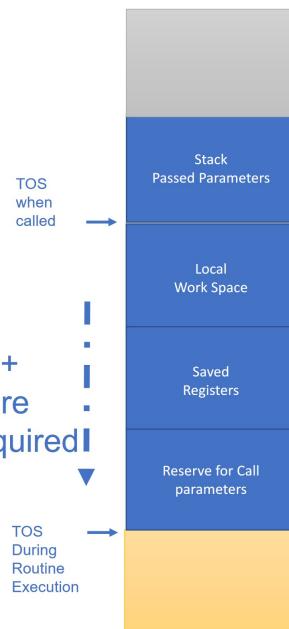
addi \$sp, \$sp, -4

sw \$ra, (\$sp)

כלומר קודם מזיזים את המצביע לראש המחסנית מילה אחת למטה בזיכרון (מעליהם את גובה המחסנית) ואז כותבים לכתובת הבסיס של המחסנית את הערך של \$ra, כך שהוא עכשו בראש העירימה.

- כדי לגשת לערכים אפשר פשוט לבצעlw על כל היסט (חייב) ביחס ל\$sp.
- כדי לעשות kop קודם נקרה את המילה ואז נזיז כלפי מעלה את \$sp.

בעת קימפול יוכל לדעת כמה מקום פ' צריכה במחסנית (מבחינת משתנים מקומיים, שמירת רגיסטרים ומקום לקרוא לפ' אחרות). מבחינה סידור הזכרון בעת קראה לפ', המחסנית תראה כך

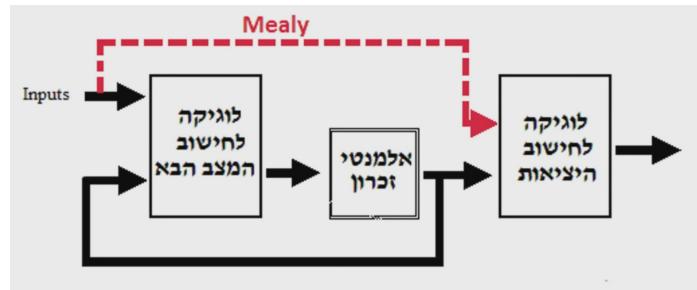


פקודות קראה לפונקציה

- \$s זג קופצת לכתובות שברגייסטר \$.

תרגול

להדגמת ההבדל בין מכונות Moore ל-Mealy, ראו האירור הבא כאשר בשחרור מכונה Moore ובאדום הוטספת השופכת אותה למכונת Mealy.

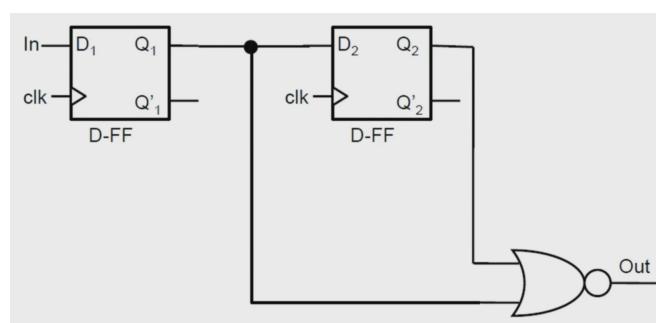


אנליזה של מעגלים סינכרוניים

בהתאם מ Engel, נרצה להבין מה הוא יוצר מוכנה והוא מייצג, אך הוא מתנה). השלבים לאנזה הם:

1. הצגת הקלטמים לזכרוון ופלטי המugal באמצעות הפלטמים מהזכירו והקלטמים למעגל.
 2. כתיבת טבלת מעברים.
 3. כתיבת טבלת מעברים סימבולית (טבלה עם שמות מוצבים).
 4. רישום אוטומט מוצבים.
 5. ניתוח האוטומט באמצעות סדרת בוחרן.

דוגמה נתנו המנגל הבא



- .2. טבלת המעברים תראה כך, כאשר אנחנו עוברים על כל אפשרות ל- D_2

		In=0		In=1		
Q ₁	Q ₂	D ₁	D ₂	D ₁	D ₂	Out
0	0	0	0	1	0	1
0	1	0	0	1	0	0
1	0	0	1	1	1	0
1	1	0	1	1	1	0

כאשר חלק מהקונפיגורציות לא הגיוניות, אבל עדיין נכתבות אותן.

3. ניתן שמות למצבים, כאשר מצב מוגדר ע"י קיבוע ערכיהם של כל פלטי רכיבי הזכרון, במקרה זה יש לנו רק שני DFF-ים

שמות המצבים	Q ₁	Q ₂
A	0	0
B	0	1
C	1	0
D	1	1

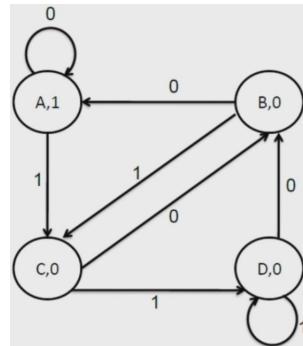
ועכשיו נוכל לחשב את הפלט של המעגל בהינתן הקלט וה המצב הנוכחי בטבלת המצבים הבאה (PS המצב הנוכחי ו-NS המצב

הבא)

	NS		
PS	In=0	In=1	Out
A	A	C	1
B	A	C	0
C	B	D	0
D	B	D	0

4. עתה נבנה אוטומט, כאשר נctrיך להחליט האם לבנות אוטומט Moore או Mealy, ובשלב הראשון עוד יכולנו לשים לב שהמעגל

הelogiy שמנדריך את הפלט תלוי רק בזיכרונו ולא בקלט ולכן נסתפק באוטומט Moore



5. נתח אוטומט, כשם שמענינו אותו בסופו של דבר היא באילו מקרים המכונה פולטה 1. השתמש בסדרת בוחן, ככלומר טבלה עם

שלוש שורות: מצב, פלט, והקלט שאיתו נבעור למצב הבא בשורה. נdag שהמעברים בין כל שני תאים סמוכים בשורת המצבים

יכסו את כל המעברים הקיימים באוטומט (כל הקשתות) לפחות פעם אחת. אין חשיבות לסדר המעבר.

In	0	0	1	0	0	1	1	0	1	0	1	0	0	0
State	A	A	A	C	B	A	C	D	B	C	B	C	B	A
Out	φ	φ	φ	1	0	0	1	0	0	0	0	0	0	1

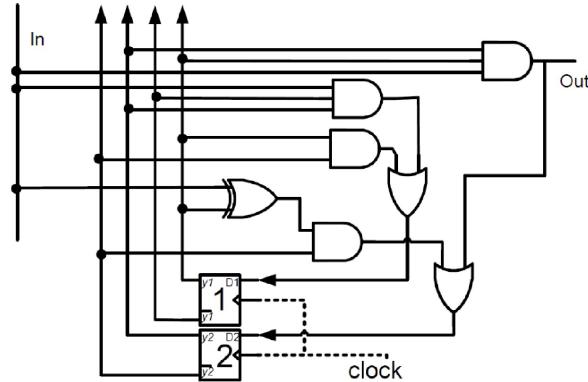
כאשר שני הפלטים הראשונים הם Φ כי כל עוד לא עברו שתי יחידות זמן, לא נוכל לדעת מה הפלט של רכיב הזיכרון השני (כי

הקלט שלו מוגדר רק אחרי המחוור הראשון) שמוביל לפלט ולכן הפלט לא מוגדר.

עתה נחפש את ה-1ים בפלטים, ונשים לב שכדי לקבל 1 בפלט, צריך שהקלטים בשני מחזורי השעון הקודמים יהיו 0, וזהו כלל השינוי! נשים לב שהקלט בזמן המחוור הנוכחי לא משנה כי מדובר במכונת Moore ולא Mealy.

איך נדע שחייב רק את שני הביטים שלפני המחוור הנוכחי ולא יותר או פחות? אפשר לבדוק כללי شيئا מורכבים יותר (שמורכבים משלושה ביטים לדוגמה) ולשים לב שהם לא מתקיים, כי אפשר לקבל 1 גם עם 100 (זה קורה בסדרת הבדיקה) וגם באמצעות 000 (ע"י היישאות ב- A שוב ושוב). הכלל האופטימלי וה邏輯י הוא זה שמענין אותנו.

דוגמה נתון המודול הבא



כאשר הפלט היחיד הוא Out (והחיצים מעלה חסרי משמעות).

1. ניצג את הפלטים והקלטים לזכרון. ובנוסף $D_2 = y_2 \cdot y_1 \cdot In + y_2' \cdot (y_1 \oplus In)$ ו- $D_1 = y_1 \cdot y_2' + y_2 \cdot y_1' \cdot In$.

2. מהצבת הערכים קיבל את בטבת המצבים

	In=0			In=1			
y_1	y_2	D_1	D_2	Out	D_1	D_2	Out
0	0	0	0	0	0	1	0
0	1	0	0	0	1	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1

3. נבחר שמות למצבים (קייבוע פלטי הזכרון) כרגע

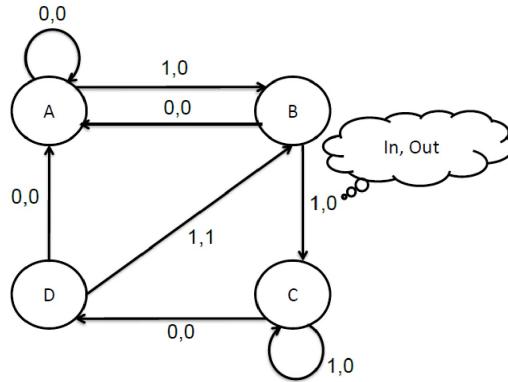
שמות המצבים	Q_1	Q_2
A	0	0
B	0	1
C	1	0
D	1	1

ועתה באמצעות הצבה של השמות בטבת המצבים קיבל כאשר הקלט כן משפיע על הפלט, כלומר מדובר במכונת Mealy.

	NS,Out	
PS	In=0	In=1
A	A,0	B,0
B	A,0	C,0
C	D,0	C,0
D	A,0	B,1

4. כדי לבנות את האוטומט נצטרך עכשו לשימוש באוטומט Mealy, ככלומר שעל הקשת נכתוב איזה קלט מעביר אותו למצב

הבא ואיזה פלט הוא מספק לנו



5. עתה נרשום סדרת בוון שתראה אותנו הדבר, רק שהפעם בחיפוש אחר כל השינוי נצטרך להתחשב גם בערך הנוכחי של הקלט

In	1	1	0	1	1	0	1	0	1	1
State	A	B	C	D	B	C	D	B	A	B
Out	φ	φ	0	1	0	0	1	0	0	0

כאשר 1 מתקבל רק כאשר הקלט בזמןים $t+3, t+2, \dots, t+1$ היה .1, 1, 0, 1.

סינטזה של מעגלים סינכרוניים

בහינתן אפיון, נרצה למשתמש מעגל סינכרוני שמקיים את הדרישות. נשתמש בסכמה הבאה :

1. בניית אוטומט בהתאם לדרישות.

2. טבלת מצבים.

3. קידוד מצבים ובחירה סוג FF.

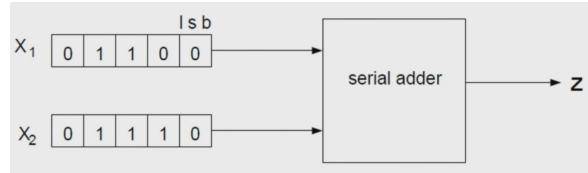
4. טבלת מעברים ופלט.

5. הגדרת פ' הכניסות של רכיבי הזיכרון ויציאת המעגל.

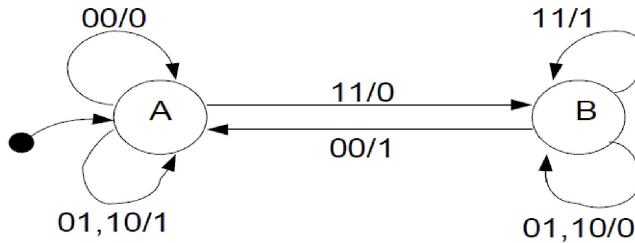
6. בניית המעגל.

דוגמה נרצה לבנות מסכם בינארי טורי, כולם מוגדר שמקבל קלטיהם x_1, x_2 וזמן t_i מחשב את הביט $h-i$ (מכיוון ה-LSB) בסכימת המספרים המתואימים על הסרט הנע של x_1, x_2 (כל מהזור מקבלים שני ביטים חדשים, שכайлו מתווסףים כ-MSB למספרים שאנו שוכנים).

ראו איור (כל פעם סוכנים את הביטים המתואימים, כמוגן עם נשא מהסכימות הקודמות)



- ראשית נבנה אוטומט שמקיים את הדרישות, כאשר הקלט (מיימן ל-/-) הוא שני הביטים $m-1-x_1$ ו- x_2 בהתאם. נבחר שהמצבים שלנו ייצגו האם יש לנו נשא מהחישוב הקודם, וכך נשמר את המידע הזה בזיכרון למשעה. A מייצג מצב שאינו בו נשא מהחישוב הקודם ו- B מייצג מצב שבו יש נשא מהחישוב הקודם. בכל פעם נחשב את הסכימה יחד עם הנשא (אם יש כזה), ונבחר מה הפלט של התוצאה ובנוסף האם יש לנו נשא ונעביר מצב בהתאם.



- نبנה טבלה מצבים, ש מכילה גם את הפלט במצב אליו עוברים כי הפלט תלוי בקלט הנוכחי כי זו מכונת Mealy

		NS (Next State), z (Output)			
		Input			
PS (Present State)		00	01	11	10
A	00	A,0	A,1	B,0	A,1
	01	A,1	B,0	B,1	B,0

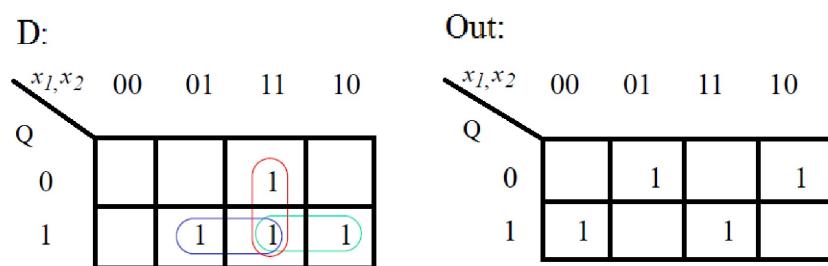
כאשר נשים לב שהקלטים לא מסודרים לקסיקוגרפיה אלא כמו בטבלה קרנו!

- קידוד המצבים דורש בחירות ייצוג בינארי למצבים, כאשר יש שניים זה די קל - מספיק ביט אחד שייצג את A כשהוא 0 ואת B כשהוא 1.
- נחליף את A ו- B בטבלה בביטויים המתואימים להם ונקבלת טבלה מעברים ופלט ש מכילה רק ביטים

		NS (Next State), z (Output)			
PS (Present State)		Input			
		00	01	11	10
0	0	0,0	0,1	1,0	0,1
	1	0,1	1,0	1,1	1,0

5. נפצל את הטבלה לשתי מפות קרנו (אחת ל-D, הקלט ל- DFF שיחזיק את המצב הבא ואחת לפולט) ונכסה את הטבלה כמו שעשינו

בתרגול 2



$$\text{כך שנתקבל בסופו של דבר } Out = Qx'_1x'_2 + Q'x'_1x_2 + Qx_1x_2 + Q'x_1x'_2 \text{ ו- } D = x_1x_2 + Qx_1 + Qx_2$$

6. הסתודנטית הערנית תשים לב שהחישובים האלה הם בדיקת הפלטים של FA ולכן (באופן שדי מתאים לאפיון) נקבל שنمמש את המעגל

כז

