מערכות ספריות – סיכום

<u>בסיסי ספירה שונים:</u>

ייצוג בבסיס b כלשהו: $b^n \cdots b^3\ b^2\ b^1\ b^0\ .\ b^{-1}\ b^{-2} \cdots$

המרה מבסיס 10 לבסיס 2: שיטת החלוקה החוזרת. נחלק ב2 ונרשום את השארית עד שנגיע ל- 0. מימין לשמאל.

729	364	182	91	45	22	11	5	2	1	0
שארית:	1	0	0	1	1	0	1	1	0	1

הספרה האחרונה (כשמחלקים את 1 ומגיעים ל-0) תמיד תהיה 1. זאת הספרה השמאלית, ה- MSB בייצוג הבינארי.

Most Significant Bit = MSB, הספרה המשמעותית ביותר – החזקה הכי גדולה.

כדי להמיר שבר מבסיס 10 לבסיס 2: הכפלה חוזרת - משמאל לימין. נכפיל ב-2 את ה**שבר בלבד** עד שנקבל 0, או עד שנראה שאנחנו בלולאה (אם הגענו לשלם ושבר שכבר ראינו בדרך):

חלק שלם	0	1	1	0	1	
שבר	0	0	0.5	0.75	0.375	0.6875

מעבר בין בסיס 10 לבסיסים אחרים – בדומה לבסיס 2, פשוט נכפיל / נחלק בבסיס השונה. כך אפשר לעבור בין כל שני בסיסים דרך בסיס 10.

אם מתקיים $b_1^n = b_2$, אפשר לעבור בין הבסיסים ע"י פריסת / הלחמת ביטים: לדוגמה $3^2 = 9$:

במעבר מבסיס 9 לבסיס 3, נפרוס כל ספרה ל-2 ביטים (או יותר, לפי היחס):

8	2	7
22	02	21

בכיוון השני, נלחים כל שני ביטים (או יותר, לפי היחס) לביט אחד. נתחיל מימין, ואם צריך "נרפד" עם אפסים בשמאל:

לדוגמה 10,1110,1110,0110 מבסיס 2 לבסיס 16. נוסיף 2 אפסים משמאל.

0	1	1	0	0	1	1	1	0	1	1	1	0	1	0	0
6				E				E				2			

אם מתקיים: $b_1 = b^n, b_2 = b^m$, אפשר לעבור דרך בסיס b.

בסיס הקסדצימאלי – 16 ספרות. ספרות 10-15 מיוצגות ע"י A=10, B=11, C=12, D=13, E=14, F=15

גלישה: אם מנסים לייצג מספר שגדול יותר מטווח הייצוג. לדוגמה אם נחבר 110+101 במערכת של 3 ביטים.

<u>ייצוג מספרים בינאריים חיוביים ושליליים:</u>

שיטת **גודל וסימן:** הביט השמאלי מייצג סימן: 0- חיובי, 1- שלילי. טווח ייצוג: $2^{n-1} - 1$ מספרים. $-2^{n-1} + 1 \to 2^{n-1} - 1$

שיטת **משלים ל-1:** חיובי – כמו גודל וסימן. שלילי – כותבים את החיובי והופכים את כל הסיביות. טווח ייצוג כמו גודל וסימן.

שיטת **משלים ל-2:** חיובי – כמו גודל וסימן. שלילי – כותבים את החיובי, הופכים את כל הסיביות, ועושים 1+.

הביט השמאלי הוא בעצם הערך השלילי, כל השאר חיובי: לדוגמה עבור 4 ביטים: המספר 5-

ביט:	1	1	0	1
ערך:	$2^0 = 1$	$2^1 = 2$	$2^2 = 4$	$-2^3 = -8$

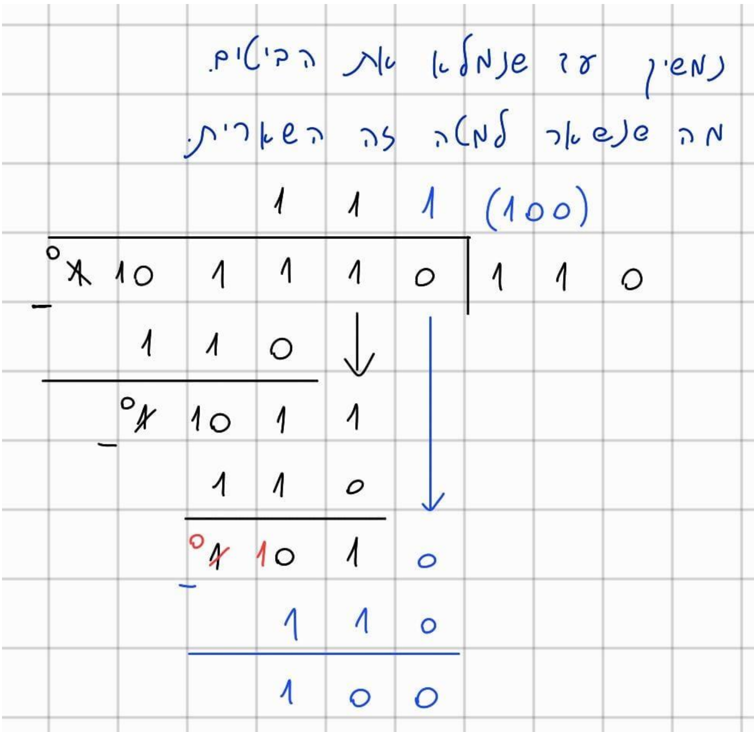
טווח ייצוג: 2^n מספרים. $-2^{n-1} \to 2^{n-1} - 1$

חיבור: חיבור בינארי רגיל. זורקים את הנשא. אם $C_{in} = C_{out}$ (הנשא האחרון שווה לנשא שלפניו) זה אומר שלא הייתה גלישה והמספר תקין. אחרת, הייתה גלישה (חרגנו מטווח הייצוג).

חיסור: עושים חיבור עם המספר הנגדי. השוואה: $A \geq B \ \leftrightarrow A - B \geq 0$

כפל: כמו כפל ארוך רגיל. כל שורה תהיה או אפסים, או המספר העליון ב"הזזה" שמאלה.

חילוק: חילוק ארוך.



בעיות טווח ייצוג – דוגמה: כדי לייצג את הטווח 10- עד 10+ עם רזולוציה של 1/2, במשלים ל-2: בשביל החלק השלם צריך 5 ביטים – טווח $-2^{5-1} \rightarrow 2^{5-1}$ (זה נותן יותר, אבל 4 סיביות היה נותן רק בין 8- ל- 8+). כדי לייצג חצאים, צריך ביט שמייצג פחות מ 2^{-1} (או בדיוק). במקרה שלנו ביט אחד מספיק. סה"כ 5 מימין ואת משמאל לנקודה.

עוד דוגמה: אם צריך טווח של 5- עד 25+, אפשר לעשות *offset* – נייצג 15- עד 15+ ונוסיף קבוע לכל מספר.

שיטת נקודה צפה – floating point:

נותן אפשרות למספרים גדולים או קטנים מאד, ורמת דיוק גבוהה. מתבסס על ייצוג מדע של מספרים: $5783 = 5.783 \times 10^3$

או בבינארי: $1011100101 = +1.011100101 \times 2^9$ מנטיסה כפול (2 בחזקת אקספוננט) ייצוג ב- 32 ביט:

0	1	0	0	0	1	0	1	0	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
סימן	אקספוננט – 8 ביט								מנטיסה – 23 ביט																						

המנטיסה מרופדת באפסים מימין, אם צריך. האקספוננט מיוצג עם offset של 127+, כדי שנוכל לרשום מספרים שליליים. אז פה כתוב 136 שזה בעצם 9. אם נרשום 0 זה 127-. טווח הייצוג באקספוננט הוא $2^{-127} \to 2^{128}$

קוד גריי: ייצוג בינארי שבו בין שני מספרים סמוכים יש הבדל של ביט אחד. $(11101101)_2 = (10110110)_{gray}$

כדי להמיר מבינארי לגריי:

הMSB נשאר כמו שהוא. לכל ביט אחר, נבדוק אותו עם הביט משמאל: אם זהות – 0, אם שונות – 1.

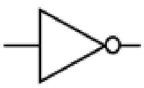
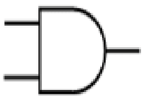
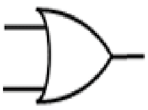
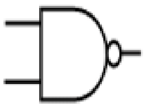
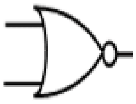
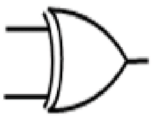
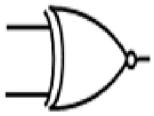
פורמלית: עבור n ביטים: $G_n = B_n\,,\quad G_i = B_i \oplus B_{i+1}$

מעבר מגריי לבינארי: הMSB נשאר כמו שהוא. לכל ביט אחר, הולכים משמאל לימין וסופרים את מספר האחדות שיש משמאל. אם זוגי – נרשום את הביט. אם אי-זוגי – נרשום את המשלים.

$$B_n = G_n\,,\quad B_i = G_i \oplus G_{i+1} \oplus G_{i+2} \oplus \cdots \oplus G_n$$

<u>אלגברה בוליאנית:</u>

שערים לוגיים:

Gate Name	NOT	AND	OR	NAND	NOR	XOR	XNOR
Algebraic Expressions	$F = \overline{A}$	$F = A\,.\,B$	$F = A + B$	$F = \overline{A\,.\,B}$	$F = \overline{A + B}$	$F = A \oplus B$	$F = \overline{A \oplus B}$
Gate Symbol							
Truth Table	Input, Output A F 0 1 1 0	Inputs, Output A B F 0 0 0 1 0 0 0 1 0 1 1 1	Inputs, Output A B F 0 0 0 1 0 1 0 1 1 1 1 1	Inputs, Output A B F 0 0 1 1 0 1 0 1 1 1 1 0	Inputs, Output A B F 0 0 1 1 0 0 0 1 0 1 1 0	Inputs, Output A B F 0 0 0 0 1 1 1 0 1 1 1 0	Inputs, Output A B F 0 0 1 0 1 0 1 0 0 1 1 1

עקרון דואליות: אם נחליף את כולם: $0 \leftrightarrow 1, OR \leftrightarrow AND$ נקבל ביטוי שקול: $X + 0 = X \to X \cdot 1 = X$

1 ניטרלי לכפל ושולט בחיבור, 0 ניטרלי לחיבור ושולט בכפל. נגדיים: $X + \bar{X} = 1,\quad X \cdot \bar{X} = 0$

כפילות: $X \cdot X = X,\quad X + X = X$ משלים: $\overline{(\bar{X})} = X$

קיבוץ: $(X + Y) + X = X + (Y + Z),\quad X(YZ) = (XY)Z$

פילוג: $X(Y + Z) = XY + XZ,\quad X + YZ = (X + Y)(X + Z)$

דה-מורגן: $\overline{(X + Y)} = \bar{X} \cdot \bar{Y},\quad \overline{(XY)} = \bar{X} + \bar{Y}$

בליעה: $X + XY = X,\quad X(X + Y) = X,\quad X + \bar{X}Y = X + Y,\quad X(\bar{X} + Y) = XY$

דוגמה: שימוש ב*XOR* להצפנה: לכל אחד יש קוד, הקוד האמיתי הוא *XOR* לכל ביט. ככה רק שלושתם ביחד יכולים לדעת את הקוד האמיתי:

S1	1	0	1	1	0	1	1	0
S2	0	1	1	0	1	1	1	1
S3	0	0	0	0	1	1	0	0
C	1	1	0	1	0	1	0	1

שימוש ב*XOR* לגילוי שגיאות: ביט בקרה:

	0	1	1	0	1	1	0	1	1
XOR		1	0	0	1	0	0	1	1

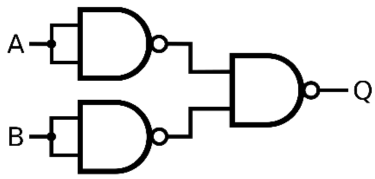
שגיאה:

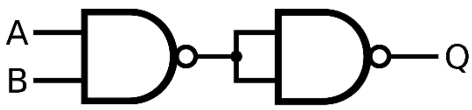
	0	1	1	0	1	1	1	1	1
XOR		1	0	0	1	0	0	1	0

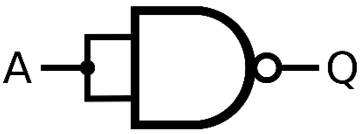
הביט בקרה הוא 1 אבל יצא לנו בחישוב 0.

מערכת פעולות שלמה: אם יש לנו את הפעולות: {AND, OR, NOT}. השער *NAND* הוא מערכת שלמה כי אפשר לייצג איתו את שלושת השערים:

$$X\ OR\ Y = (NOT\ X)\ NAND\ (NOT\ Y), \qquad X\ AND\ Y = (X\ NAND\ Y)\ NAND\ (X\ NAND\ Y), \qquad NOT\ X = X\ NAND\ X,$$

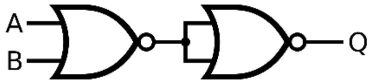


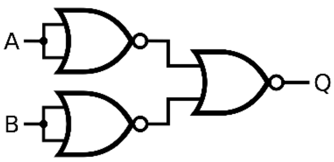


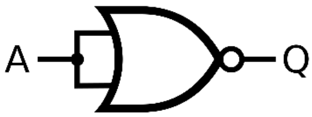


גם *NOR* הוא מערכת פעולות שלמה:

$$X\ OR\ Y = (X\ NOR\ Y)\ NOR\ (X\ NOR\ Y) \qquad X\ AND\ Y = (X\ NOR\ X)\ NOR\ (Y\ NOR\ Y), \qquad NOT\ X = X\ NOR\ X,$$







פונקציות בוליאניות:

A	*B*	*C*	*F*
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

$$\begin{aligned} f(a,b,c) &= \bar{a}bc + a\bar{b}c + ab\bar{c} + abc \\ &= \bar{a}bc + abc + a\bar{b}c + abc + ab\bar{c} + abc \\ &= (\bar{a}+a)bc + \left(\bar{b}+b\right)ac + (\bar{c}+c)ab \\ &= ab + bc + ac \end{aligned}$$

ליטרל: משתנה או משלים. $X, \bar{X}$

מינטרם: מכפלה שבה כל משתנה מופיע בתור ליטרל.

מקסטרם: סכום שבו מופיע כל משתנה בתור ליטרל.

צורות קנוניות: SOP = Sum Of Products, סכום מינטרמים. ערך הפונקציה יהיה 1 אם קיים לפחות מינטרם אחד שערכו 1.

כל מינטרם מתאים לשורה בטבלת אמת (משבצת במפת קרנו). דוגמה עבור 3 משתנים: $f = \sum(1,2,7) = \bar{x}\bar{y}z + \bar{x}y\bar{z} + xyz$

המספרים הם מספרי השורות שבהן ערך הפונקציה הוא 1. המרת המינטרם למספר שורה: משתנה זה 1, משלים זה 0:

$\bar{x}$	y	$\bar{z}$
0	1	0

שורה 2: $(010)_2 = 2_{10}$

POS = Product Of Sums, מכפלה של מקסטרמים. ערך הפונקציה יהיה 0 אם קיים לפחות מינטרם אחד שערכו 0.

כל מקסטרם מתאים לשורה בטבלת אמת (משבצת במפת קרנו). דוגמה:

$$f = \prod(1,3,5) = (x + y + \bar{z}) \cdot (x + \bar{y} + \bar{z}) \cdot (\bar{x} + y + \bar{z})$$

המספרים הם מספרי השורות שבהן ערך הפונקציה הוא 0. המרת המינטרם למספר שורה: משתנה זה 0, משלים זה 1:

x	y	$\bar{z}$
0	0	1

שורה 1: $(001)_2 = 1_{10}$

נשים לב ש*SOP* ו- *POS* הן משלימות (מבחינת מספרי שורות):

$$f = \sum(1,2,7) = \Pi(0,3,4,5,6), \qquad \bar{f} = \sum(0,3,4,5,6) = \Pi(1,2,7)$$

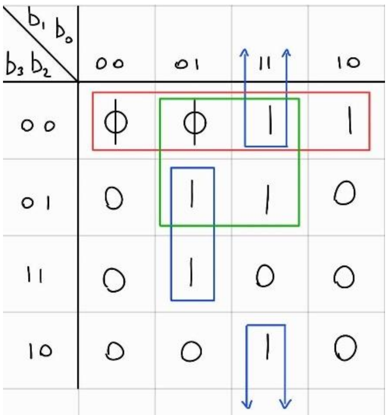
צמצום פונקציות ע"י **מפת קרנו**:

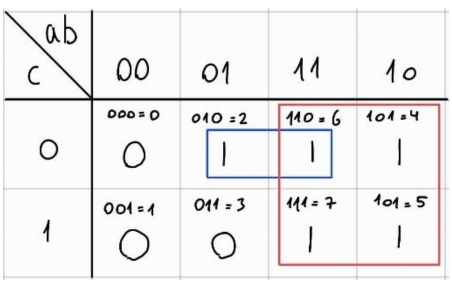
כיסוי מלבנים: מלבן צריך להיות בגובה ורוחב שהם חזקות של 2. מלבן מקסימלי נקרא **גורר.** יש 3 סוגים:

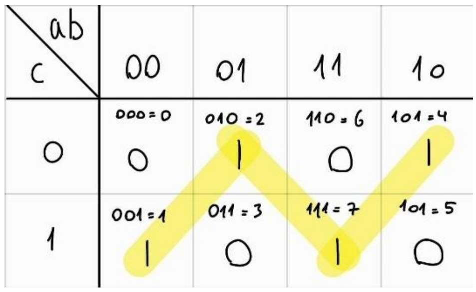
1) גורר רגיל – ליטרל או מכפלת ליטרלים שמכוסה ע"י הפונקציה.
2) גורר ראשוני – גורר שכל השמטה של ליטרל ממנו יוצרת מכפלה שלא מכוסה ע"י הפונקציה.
3) גורר ראשוני הכרחי – גורר ראשוני שמכסה מינטרם שלא מכוסה ע"י אף גורר ראשוני אחר.

לדוגמה: $f(x,y,z) = xy + xz + xy\bar{z}$ המסומנים הם ראשוניים. השלישי לא, כי אם נמחק את z' נקבל את xy.

דוגמה למפות קרנו ב3 ו-4 משתנים: המלבנים הם גוררים ראשוניים. כל מינטרם מסומן 0 או 1. אם יש מינטרם שלא משנה לנו מה הוא יוצא – הוא נקרא *don't care*, ומסומן $\emptyset$. הוא מצטרף ל0 או 1, לפי מה שעוזר לנו. לדוגמה מפת קרנו של מספרים ראשוניים, שבה 0,1 לא מוגדרים. נשים לב שמפת קרנו היא ציקלית (*cyclic*), המלבן יכול להיות מפוצל בין תאים קיצוניים:







בדוגמה הימנית אין מלבנים, אבל כשיש סדר כזה של זיגזג, זה מבטא $A \oplus B \oplus C$

מפת קרנו בנויה לפי קוד גריי – בין משבצות סמוכות רק ליטרל אחד משתנה.

בניית צורה קנונית לפי מפת קרנו: לוקחים את הגורמים הראשוניים ההכרחיים, כל אחד מיוצג ע"י הליטרלים **שלא** משתנים בתוך המלבן. אם הוא 1 רושמים את הליטרל, אם הוא 0 רושמים את הנגדי.

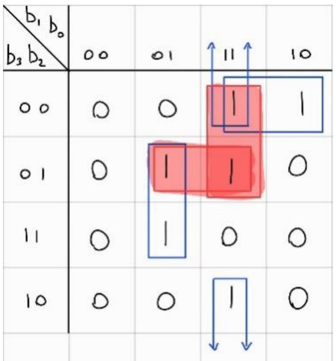
בדוגמה האמצעית: $f(a,b,c) = a + bc$

בדוגמה של המספרים הראשוניים: $f(b_3,b_2,b_1,b_0) = \overline{b_3}\,\overline{b_2} + \overline{b_3}\,b_0 + b_2\,\overline{b_1}\,b_0 + \overline{b_2}\,b_1 b_0$

ככל שהמלבן גדול יותר, הוא מיוצג ע"י פחות ליטרלים. ריבוע בודד- כל הליטרלים. כל פעם שנכפיל את הגודל, ליטרל אחד יורד.

בדוגמה הזאת, הגורמים האדומים הם לא הכרחיים, כי כל מינטרם מכוסה ע"י לפחות עוד גורם אחד. אנחנו עדיין צריכים לקחת אחד מהם, זה לא משנה איזה:

$$f(b_3,b_2,b_1,b_0)=\overline{b_3}\,\overline{b_2}b_1+\overline{b_3}\,b_1b_0+\overline{b_2}\,b_1b_0+b_2b_1b_0$$

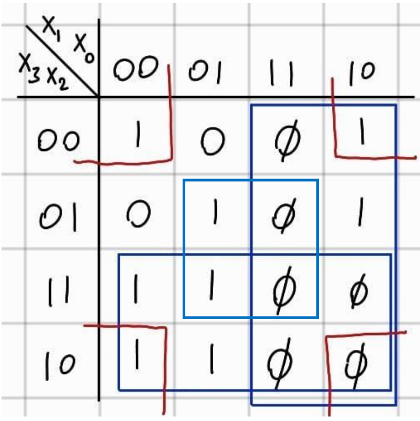


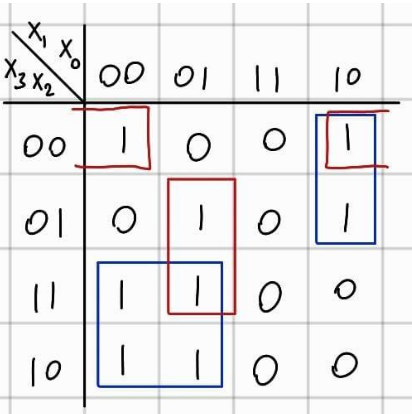
Seven segment:

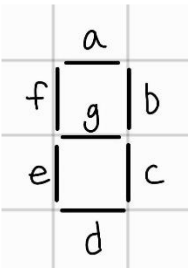
רכיב של מספר דיגיטלי: נתמקד ב-A: השורה הימנית בטבלה מייצגת מתי A דלוק.

מה קורה בשורות 10-15? יש שני פתרונות: אם אמורים לא להדליק את A, צריך לרשום 0. אם זה לא משנה, זה *don't care*.

	X1	X2	X3	X4	A
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
	1	0	1	0	Ø
	1	0	1	1	Ø
	1	1	0	0	Ø
	1	1	0	1	Ø
	1	1	1	0	Ø
	1	1	1	1	Ø







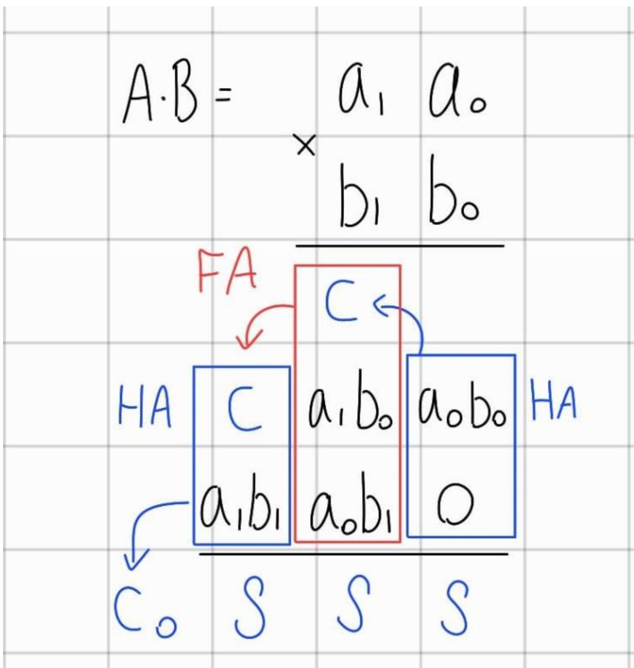
במקרה עם don't care:

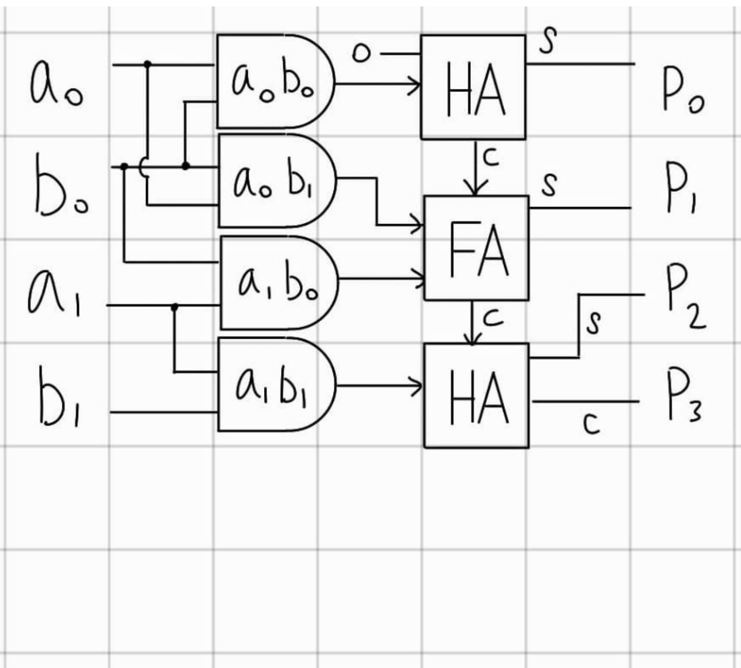
$$A=x_1+x_3+x_2x_4+\overline{x_2}\,\overline{x_4}$$

במקרה עם האפסים:

$$A=\overline{x_1}\,x_3+\overline{x_1}\,x_2x_4+x_1\overline{x_2}\,x_3+\overline{x_2}\,\overline{x_3}\,\overline{x_4}$$

מימוש מכפל:



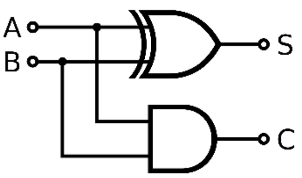


<u>תכן לוגי:</u>

חצי מחבר – *HA - half adder*

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

הסכום הוא פשוט XOR, הנשא הוא AND



מחבר שני ביטים. נרצה לחבר 2 מספרים בינאריים ע"י סדרה של חצי מחברים. הבעיה היא שנצטרך לחבר 3 ביטים – אחד מכל מספר, פלוס נשא.
הפתרון – מחבר מלא, full adder.
מקבל גם נשא נכנס.

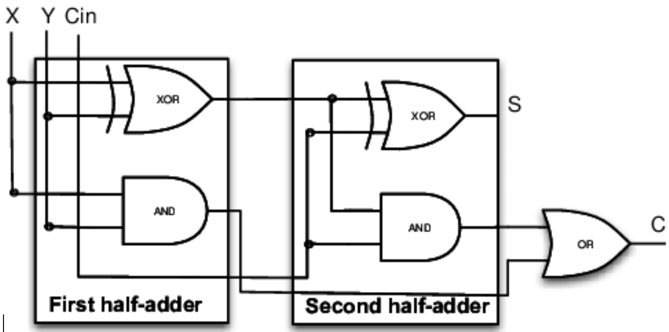
מחבר מלא – *FA - full adder*

C in	A	B	C out	S
1	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

מפת הקרנו של S נותנת את הזיגזג שראינו בעמוד הקודם, שזה $A \oplus B \oplus C_{in}$, מאד נוח למימוש ע"י שערים. נעשה מפת קרנו ל- C ונקבל: $C_{out}(A,B,C_{in}) = AB + AC_{in} + BC_{in}$ שזה לא נוח. נשים לב ששניים מהגוררים נותנים לנו רק ליטרל אחד, אז ניקח אותו לבד (ולא את כל הגורר) ונקבל: $C_{out} = AC_{in} + \bar{A}BC_{in} + AB\overline{C_{in}} =$
$= AC_{in} + B(\bar{A}C_{in} + A\overline{C_{in}}) = AC_{in} + B(A \oplus C_{in})$
שזה משהו שיותר נוח לממש.

$C_{in}\backslash AB$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

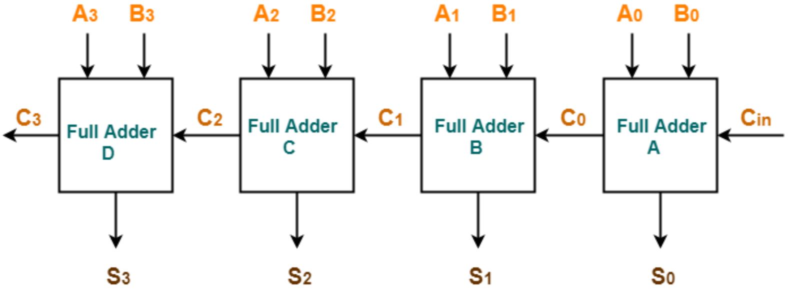
בסך הכל, מחבר מלא ניתן למימוש ע"י שני חצי מחברים ועוד שער *OR*:



כדי לחבר מספרים בינאריים, נשתמש ב *ripple carry adder*:
כל מיקום הוא *FA*:

					C_{in}
A	a_n	a_{n-1}	…	a_1	a_0
B	b_n	b_{n-1}	…	b_1	b_0
C_{out}					

כדי לעשות $A-B$, נעשה $A+(-B)$. נרשום את B+, נהפוך את כל הסיביות ונוסיף 1 ב- C_{in}



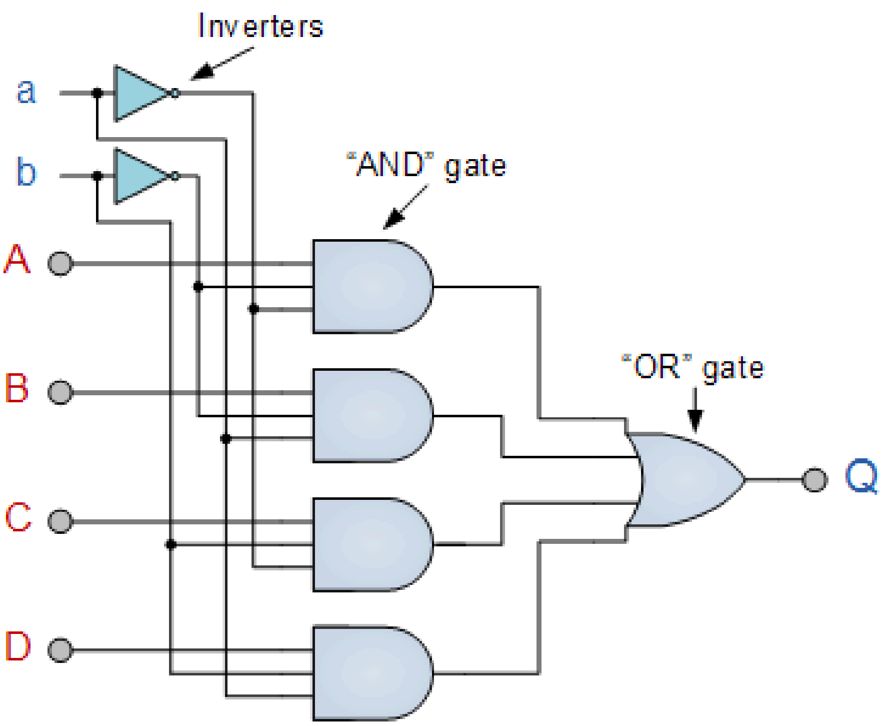
4-bit Ripple Carry Adder

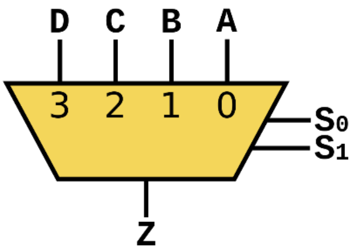
MUX – multiplexer – מרבב

מקבל כניסות מידע וכניסות כתובת.

הכתובת קובעת איזה מידע להעביר:

S_1	S_0	Z
0	0	A
0	1	B
1	0	C
1	1	D

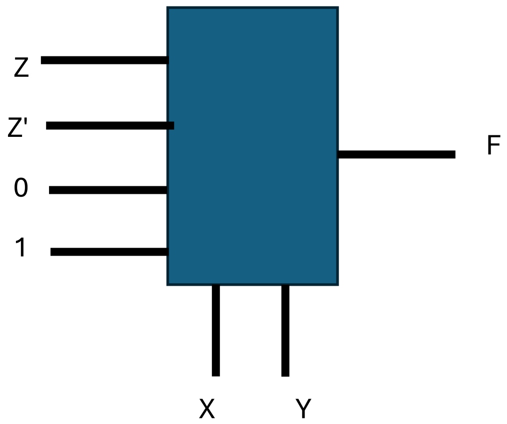




שימוש במרבב למימוש פונקציות בוליאניות:

- נבנה טבלת אמת
- נחלק את הטבלה לזוגות של שורות שבהן *X,Y* לא משתנים
- נראה מה הקשר בין *Z* ל- *f*

x	y	z	f	
0	0	0	0	$f=z$
0	0	1	1	
0	1	0	1	$f=\bar{z}$
0	1	1	0	
1	0	0	0	$f=0$
1	0	1	0	
1	1	0	1	$f=1$
1	1	1	1	



כל פונקציה בוליאנית של *n* משתנים ניתנת למימוש ע"י מרבב של *n-1* בקרות.

לדוגמה Cout של FA:

1	1	1	0	1	0	0	0	
7	6	5	4	3	2	1	0	
1	0	1	0	1	0	1	0	A
1	1	0	0	1	1	0	0	B
1	1	1	1	0	0	0	0	Cin

כשיש 2 או יותר אחדות, נרצה להוציא 1.

אפשר גם בצורה יותר יעילה:

1	Cin	Cin	0	
3	2	1	0	
1	0	1	0	A
1	1	0	0	B

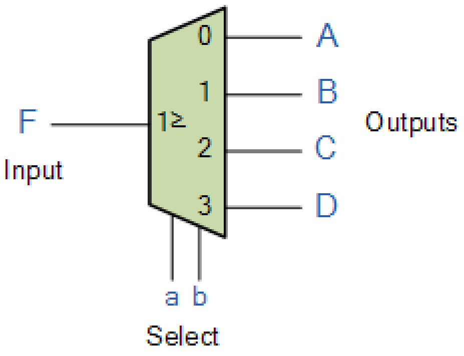
מפלג – ***DEMUX – demultiplexer***

הפוך ממרבב. מקבל כניסת מידע אחת, m כניסות בקרה,

ושולח את המידע לאחת מ- 2^m יציאות. כניסות הבקרה

מייצגות מספר בינארי של היציאה שתיבחר.

לפעמים יש כניסת *Enable* – מאפשר. קובע אם יעבור משהו בכלל או לא.

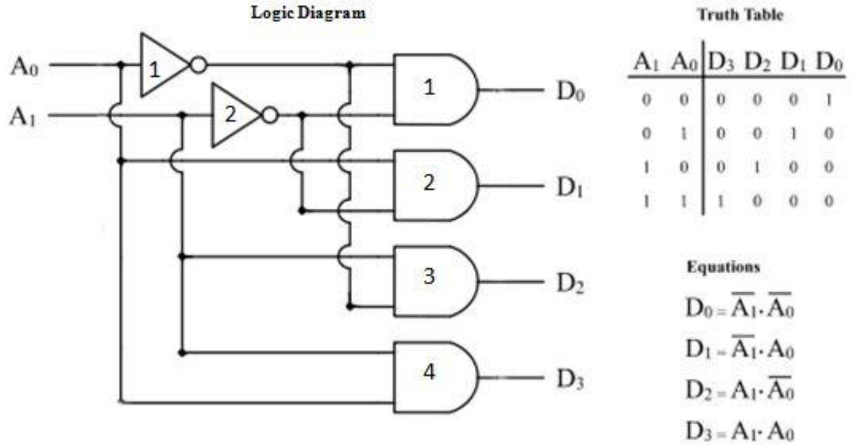


מפענח – *decoder*

ממיר מספר בינארי לפלט. n כניסות, 2^n יציאות. לדוגמה אם נכנס 101, יציאה 5 תוציא 1.

מקודד – *encoder*

הפוך ממפענח



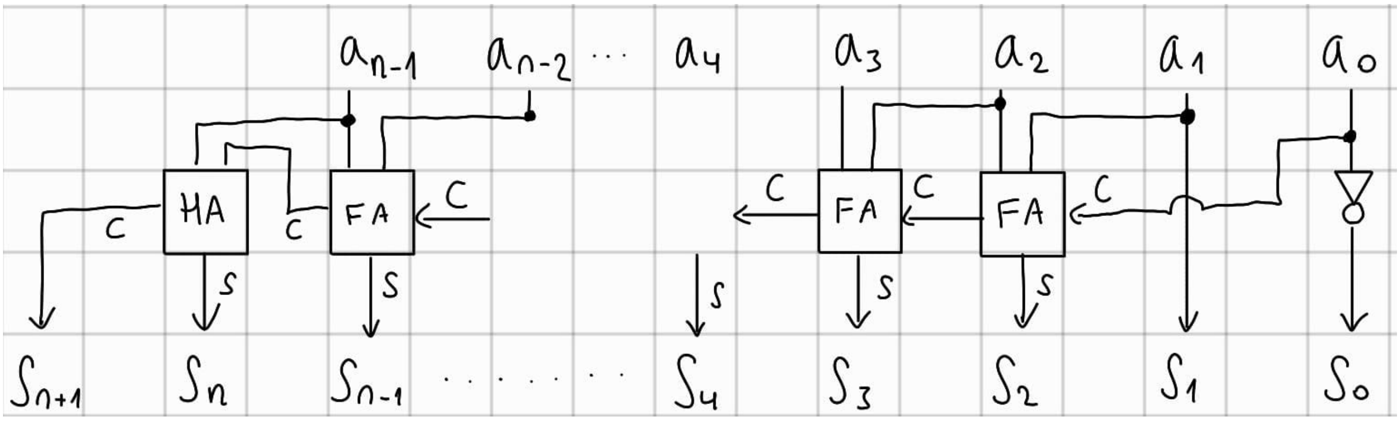
תרגיל כיתה: רכיב שמבצע *3A+1*: (קשור להשערת קולץ?)

קל לכפול מספר ב2 בבינארי – פשוט מוסיפים 0 מימין. אז אפשר לעשות:

A =	0	a_{n-1}	$\cdots$	a_1	a_0
2A =	a_{n-1}	a_{n-2}	$\cdots$	a_0	0
2A+1 =	a_{n-1}	a_{n-2}	$\cdots$	a_0	1

ואז לחבר את זה עם *A* באמצעות מחברים:

	C_{in}	C_{in}	C_{in}	a_0	a_0	
A =	0	a_{n-1}	$\cdots$	a_2	a_1	a_0
2A+1 =	a_{n-1}	a_{n-2}	$\cdots$	a_1	a_0	1
C_{out}	*HA*	*FA*	*FA*	*FA*	$a_0 \oplus a_1 \oplus a_0 = a_1$	$a_0 \oplus 1 = \overline{a_0}$

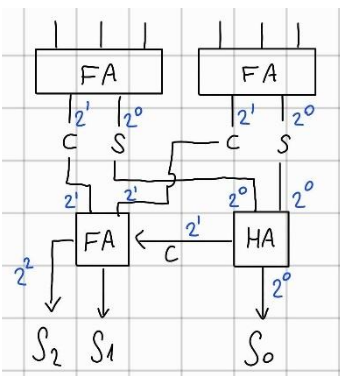


עוד תרגיל – משקל המינג: מספר האחדות בווקטור, בייצוג בינארי:

נשים לב ש*FA* בעצם סופר כמה אחדות יש בכניסות.

אז כל *FA* יכול לטפל ב3 ביטים. לדוגמה עם 6 ביטים:

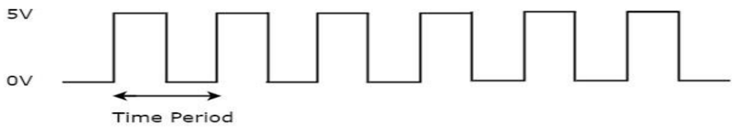
המספרים הכחולים באמצע זה כמה אחדות הסיב הזה מוסיף לסכום.



מעגלים צירופיים – *combinational logic*: המוצא הוא פונקציה של הכניסות הנוכחיות בלבד – אין זיכרון.

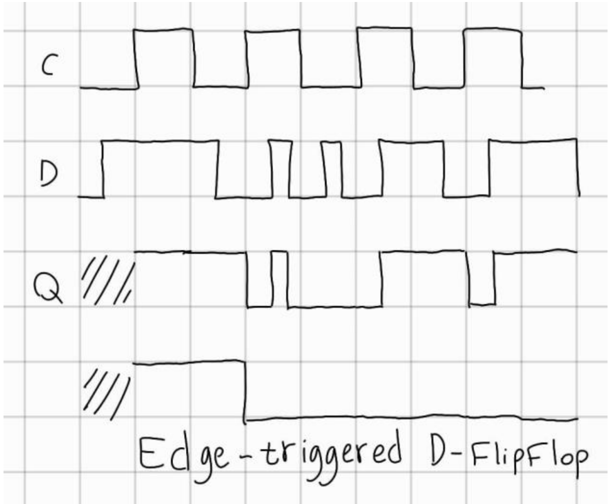
מערכת עקיבה – *sequential logic*: היציאה והמוצא הבא הם פונקציה של המצב הנוכחי.

שעון: עולה ויורד בתדירות קבועה. מרגע העלייה עד רגע לפני העלייה הבאה, זה מחזור אחד.



רכיב *D-latch*: כאשר C=0, *Q* לא משתנה

כאשר C=1, *Q* מקבל את הערך של *D*:



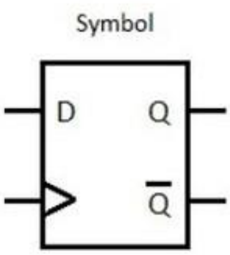
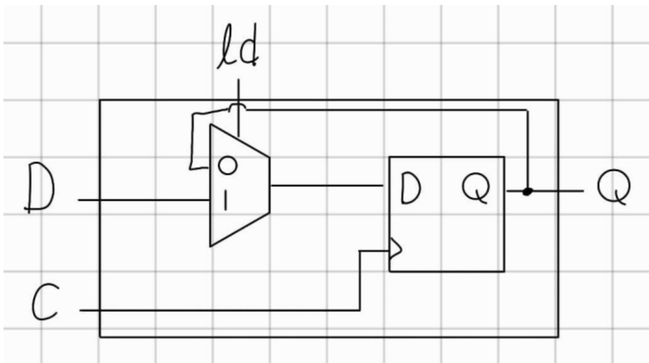


Table of truth:

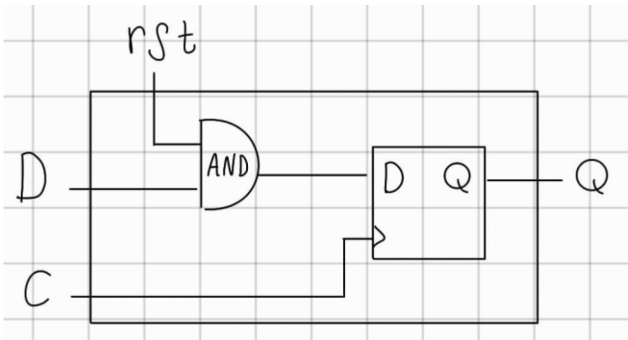
clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

הסוג האחרון – ***edge triggered D flipflop*** – הוא מה שנמצא בשימוש בפועל. מה שנכנס בעליית השעון, יישאר עד עליית השעון הבאה. כלומר אם *D* השתנה באמצע מחזור, זה לא ישפיע.

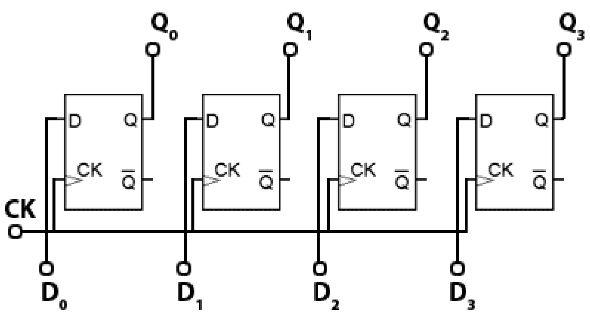
דלגלג עם כניסת אפשור – אם בעליית השעון $ld = 1$, המוצא במחזור הבא יהיה *D*. אחרת, המצב יישאר.



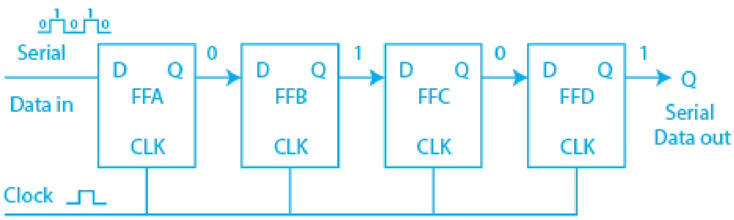
דלגלג עם כניסת איפוס – אם בעליית השעון rst=0, המוצא במחזור הבא יהיה 0. אחרת, המוצא יהיה D



אוגר – *register*: כמו הקודם, רק שיש כמה כניסות נתונים.

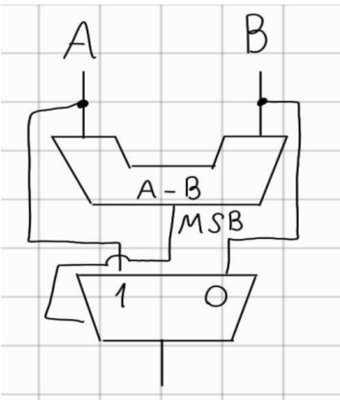


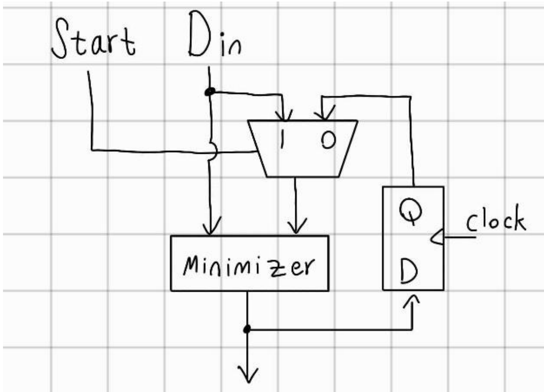
אוגר הזזה – *shift register*:
כל מוצא מחובר לכניסה הבאה. לדוגמה כדי לחלק ב-2, כל ביט מועבר למיקום הבא.

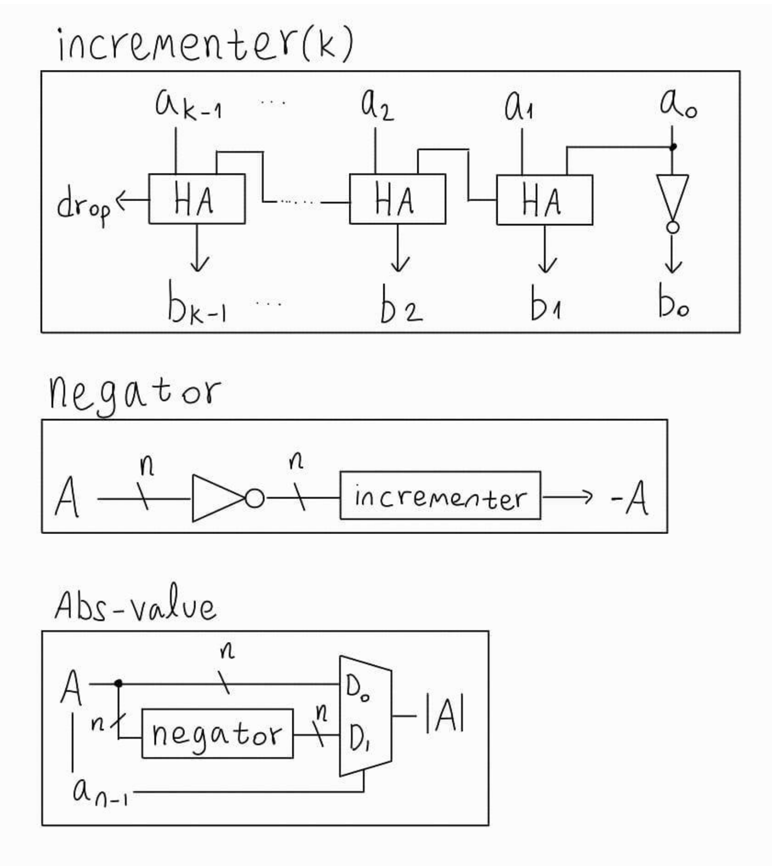


תרגיל כיתה – ***serial minimizer***: בכל מחזור מוציא את המינימום מבין הקלטים בסדרה הנוכחית. יש כניסת בקרה שמייצגת קלט חדש. אם $A \geq B$, ההפרש חיובי וה-*MSB* יהיה 0 (בשיטת משלים ל- 2). אם *start* דלוק, מה שנכנס ב- *D* ייכנס לשני הצדדים של ההשוואה. אם *start* כבוי, אחד הצדדים יהיה מה שהיה במחזור הקודם.

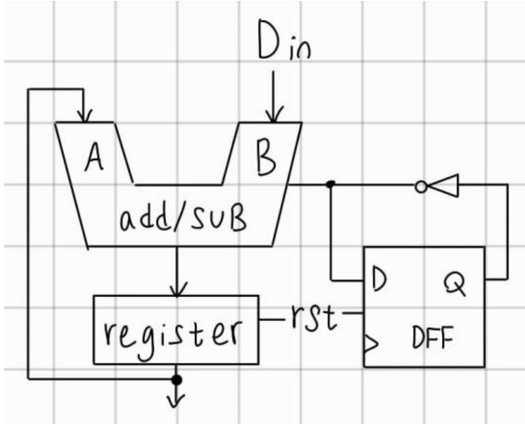
minimizer





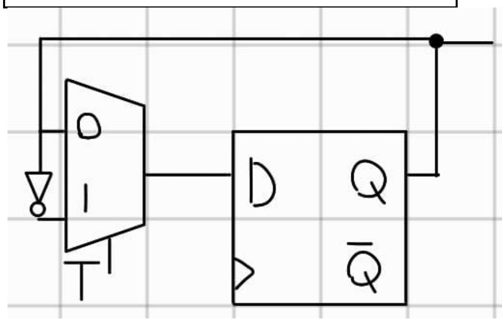


Add-sub accumulator: $D_0 - D_1 + D_2 - \cdots$



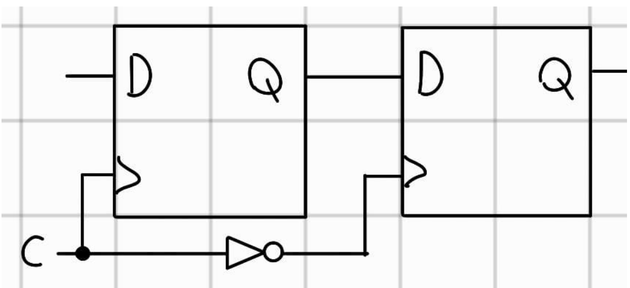
Toggle FlipFlop:

$$out(t+1) = \begin{cases} \overline{out(t)}, & T = 1 \\ out(t), & T = 0 \end{cases}$$



Master-slave:

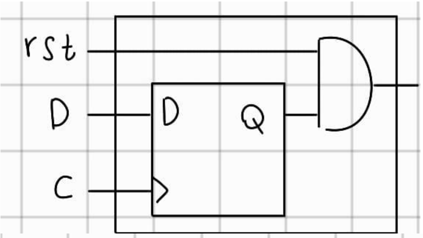
Flip-flop פעיל בירידת שעון. זה בעצם מה שיש בתוך D flip-flop. כשהשעון למטה, הוא זוכר את המידע. כשהשעון עולה הוא יכול לקבל מידע חדש אבל מוציא אותו רק בירידת השעון הבאה



אם נחליף את המיקום של המהפך (שער NOT) נקבל דלגלג שפעיל בעליית שעון.

מערכות סינכרוניות מתקדמות:

דלגלג עם איפוס אסינכרוני: אם נכנסת פקודת איפוס, היא תשפיע בעליית שעון הבאה:



תרגיל כיתה: **מונה ריבועים סינכרוני:**

בכל עליית שעון:

- n יגדל ב- 1
- $Square\_n$ יראה את n^2

$A = 2n + 1$

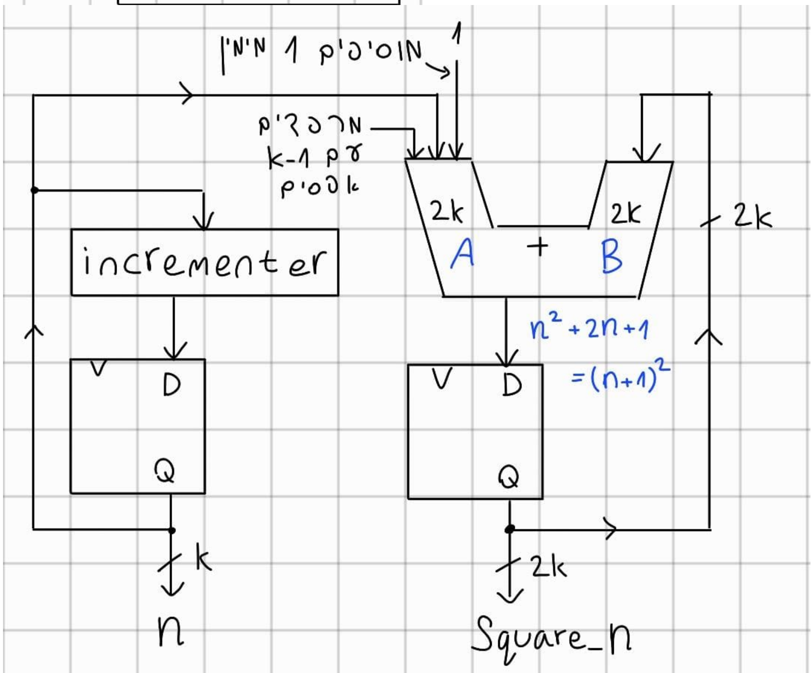
$B = n^2$

כל מחזור משתמש בn^2 מהמחזור הקודם כדי

לחשב את $(n+1)^2$. כשמוסיפים את ה1 מימין,

בעצם הוספנו ביט מימין שזה מכפיל ב2, והאחד

נותן לנו +1.

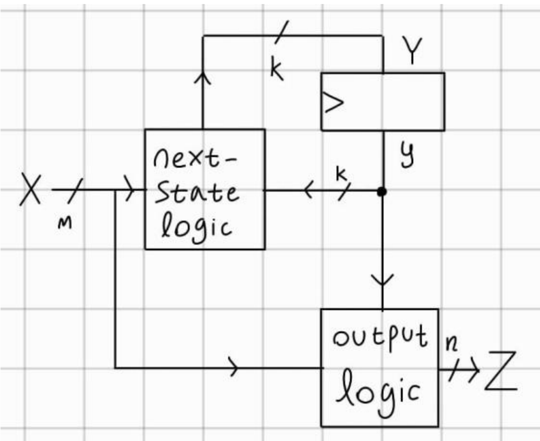


מעגלים סדרתיים:

יש שני סוגים:

סינכרוני – מצב משתנה לפי הקלט בהתאם למחזור שעון.

אסינכרוני – שינוי מיידי.



מכונת מצבים סופית:

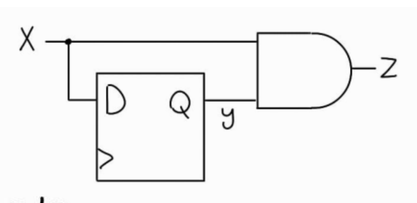
מערכת שיש לה מספר סופי של מצבים אפשריים. לדוגמה:

$Sequential\ input: x_0, x_1 \ldots$

$initial\ output: z_0 = 0$

$for\ i > 0: z_i = x_i \cdot x_{i-1}$

X	0	1	0	1	1	1	0	1
Z	0	0	0	0	1	1	0	0



נתאר אותה ע"י **דיאגרמת מצבים:**

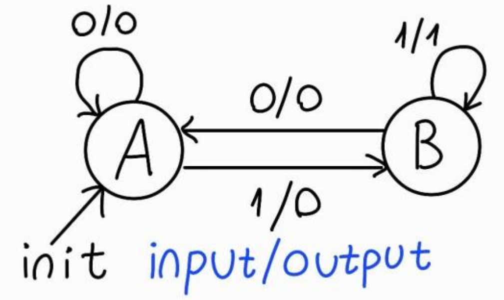
נגדיר: $A := x_{i-1} = 0, \quad B := x_{i-1} = 1$

המצב ההתחלתי הוא A. אם נקבל 0, הפלט הוא 0 ונישאר בA.

אם נקבל 1, נעבור למצב B אבל הפלט הוא עדיין 0.

אם המצב B וקיבלנו 1, הפלט הוא 1 ונשארים במצב B.

אם המצב B וקיבלנו 0, הפלט 0 ועוברים למצב A. אפשר גם:



בטבלת מצבים:

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

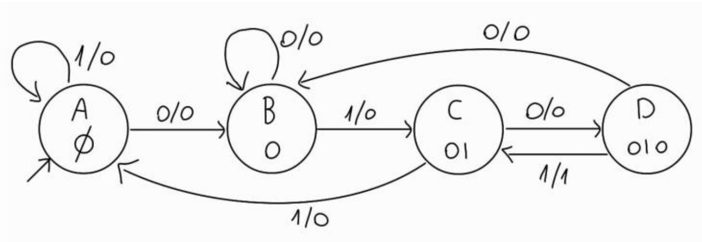
או טבלת מעברים:

	X=0	X=1
0	0,0	1,0
1	0,0	1,1

בטבלת המעברים: Y,z כאשר Y הוא המצב הבא (במקרה שלנו, זוכר מה היה הקלט הקודם) , ו- z הוא הפלט.

דוגמה 2: מכונת מצבים שמחזירה 1 אם קיבלנו "0101",
עם חזרות. נגיד בדוגמה הזאת, קיבלנו "0101" פעמיים.
נגדיר 4 מצבים ונבנה דיאגרמה: נשים לב שאם נכנס משהו
שהורס את הסדר שמחפשים, מבחינתנו הקלט ריק וזה חוזר
למצב ההתחלתי. כלומר אם קיבלנו 11, לא צריך לשמור
את זה. פשוט חוזרים למצב ההתחלתי. אם היה לנו 010
וקיבלנו עוד 1, הפלט הוא 1 וחוזרים למצב של 01..

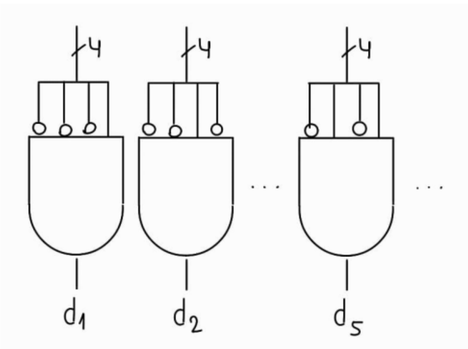
X	0	1	0	1	0	1
Z	0	0	0	1	0	1



	X=0	X=1
A	B,0	A,0
B	B,0	C,0
C	D,0	A,0
D	B,0	C,1

Y1	Y2	X=0	X=1	X=0	X=1
0	0	01	00	0	0
0	1	01	11	0	0
1	1	10	00	0	0
1	0	01	11	0	1

תרגיל כיתה - מפענח decoder עבור ספרה אחת.
מקבל ספרה בינארית כקלט ומדליק את סיבית הפלט המתאימה:
אנחנו צריכים שהקלט יהיה בדיוק הייצוג הבינארי של המספר, אז
בכל מקום שצריך להיות 0 נשים מהפך ואז השער AND יקבל 4 אחדות.



תרגיל כיתה – משקל המינג של 9 ביטים:
המשמעות של המשקלים: אם הסיב הזה
דלוק, הוא מייצג את הכמות הזאת של
אחדות בקלט.
הפלט מתאים לייצוג בינארי של 4 ביטים.