

Preparation Report LAB4

ADVANCED CPU ARCHITECTURE AND HARDWARE ACCELERATORS LAB

361.1.4693

Roy Kislev 206917064

Michael Grenader 208839845

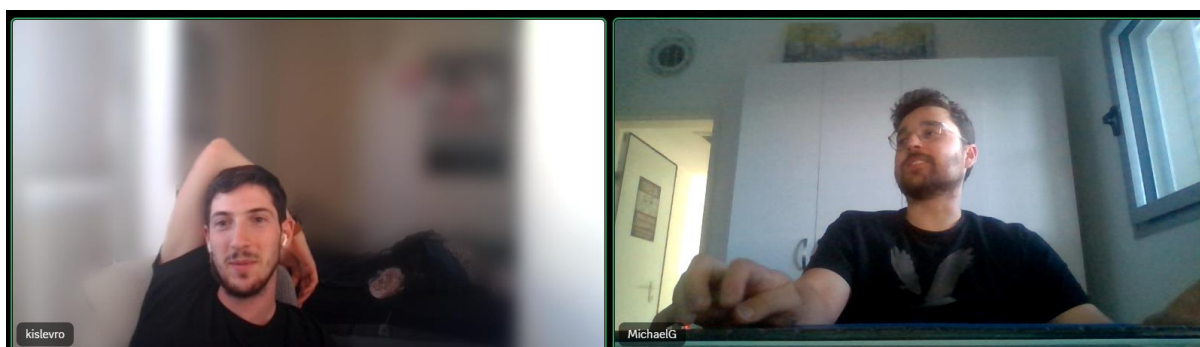


Figure 1 עבודה משותפת מרחוק

תוכן עניינים

2	מטרת המעבדה
2	בדיקת ביצועים
3	סימולצית ModelSim
4	מציאת תדר מקסימלי
6	פירוט המערכת
6	מערכת ה-ALU
9	מודול AdderSub
12	מודול Shifter
14	מודול Logic
16	אופטימיזציה חומרית
16	Signal Tap

מטרת המעבדה

במעבדה זו למדנו להשתמש ביכולות של תוכנת Quartus ובפרט לבצע סינתזה עבור מודלים שפיתחנו בעבר במעבדה 1. את הסינתזה ביצענו על גבי Cyclone V FPGA של כרטיס DE10 standard.

בדיקת ביצועים

בדיקה זו נעשתה על מנת לבדוק ביצועים ראשוניים של ה-ALU שפיתחנו במעבדה 1, אך הפעם בשילוב של סינתזה על גבי כרטיס FPGA אמיתי. מכיוון שהמערכת שפיתחנו הינה א-סינכרונית, כלומר ללא שעון, נצטרך לחבר למערכת רגיסטרים סינכרוניים לכניסה ולמוצא המערכת על מנת לבצע אנליזה בזמן.

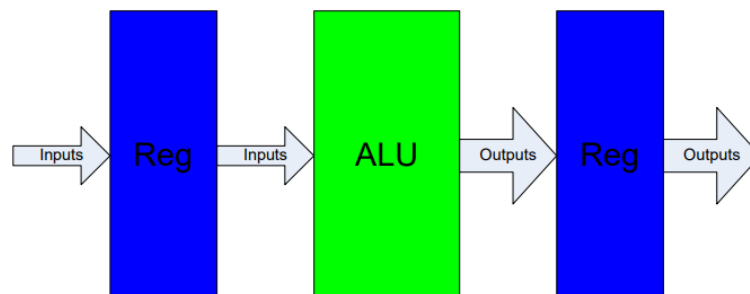


Figure 2 ALU performance test case

כאשר אל רגיסטרים אלו נחבר את שעון מגביש הכרטיס –

<i>Signal Name</i>	<i>FPGA Pin No.</i>	<i>Description</i>	<i>I/O Standard</i>
CLOCK_50	PIN_AF14	50 MHz clock input	3.3V

סימולצית ModelSim

בסימולציה זו תחילה נבצע TB ונראה את הגלים על גבי המערכת כולה ולאחר מכן עבור כל תתי המודולים. להלן כל הגלים עבור המערכת כולה, כאשר TB שלנו מאתחל את X להיות 0 ואת Y להיות 1- כאשר X קופץ ב2 וY יורד ב1. – כמו כן, ALUFN לוקח פקודות שונות בכל 100ns מתוך *icache* שמאותחל בקובץ.

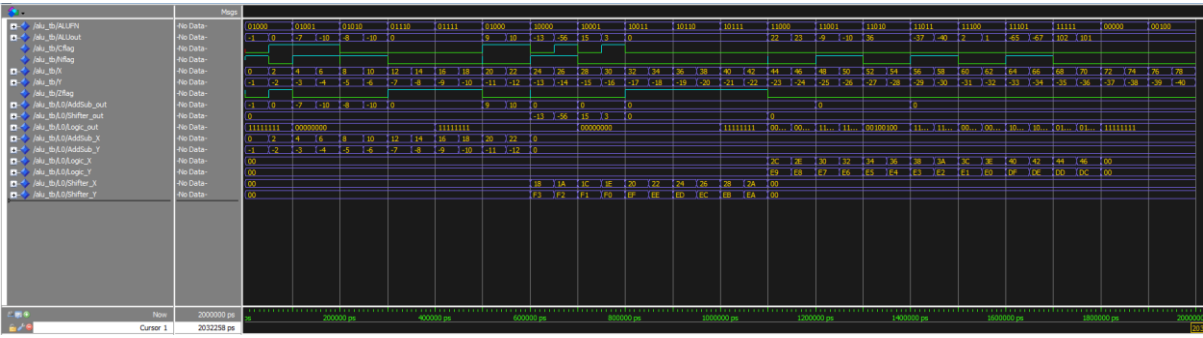


Figure 3 סימולצית גלים של המערכת כולה

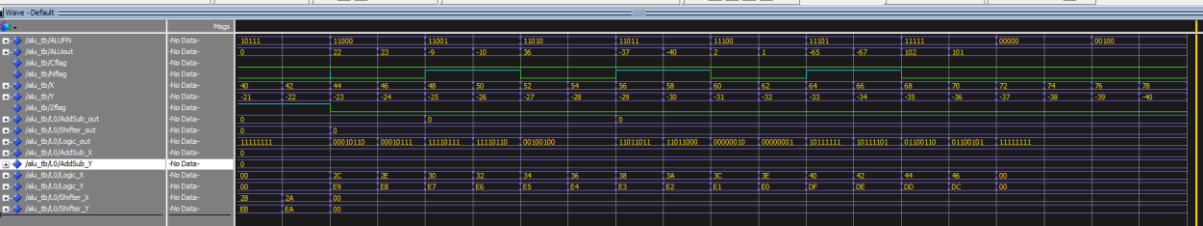


Figure 4 סימולצית גלים של המערכת כולה - זום

עבור המודול *AdderSub*

בתוצאת סימולציית הגלים עבור מודול זה התקבל כי –

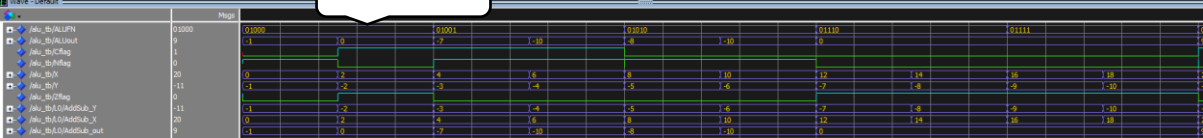


Figure 5 סימולציית גלים של מודול AdderSub

ניתן לראות ש 2 ביטים MSB ב ALUFN הם 01 ולכן המערכת משתמשת במודול *AdderSub*. כמו כן, ניתן לראות שמפני שהביטים התחתונים הם 000 אזי מתבצעת פעולת חיבור בין 2 ל-2² – ו-בין 0 ל-1 – ולכן התוצאה ב 1 – $ALUout = 0$ בהתאם.

עבור המודול *Shifter* התקבל כי –

בתוצאת סימולציית הגלים עבור מודול זה התקבל כי –

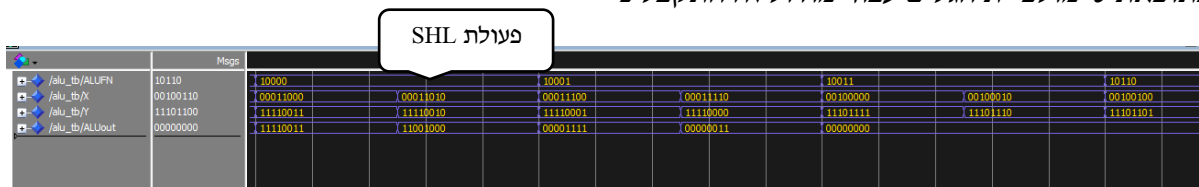


Figure 6 סימולצית גלים של מודול Shifter

ניתן לראות כי $Y = b'11110010$ בעוד ששלושת ביטי ה- LSB של X הם 010 לכן נצפה לקבל הזזה כפולה שמאלה ל- Y ולקבל **11001000** כאשר הביטים המודגשים הם הביטים שהתווספו.

עבור המודול *Logic* התקבל כי –

בתוצאת סימולציית הגלים עבור מודול זה התקבל כי –

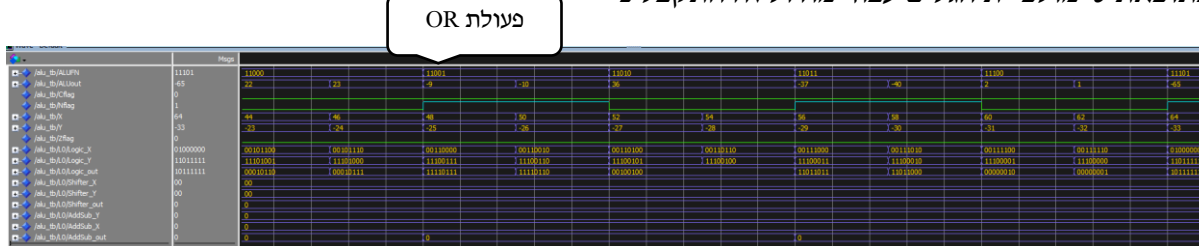


Figure 7 סימולצית גלים של מודול Logic

ניתן לראות שתחת הענן מתקבלת פעולת OR מפני שהמודול הרלוונטי הוא 11 והפונקציה המתאימה היא 001, כלומר פעולת OR לפי ISA בין סיגנל $X = 00110000$ ל- $Y = 11100111$ כאשר נצפה לתוצאה שאכן התקבלה $.res = 11110111$.

מציאת תדר מקסימלי


בכדי למצוא תדר מקסימלי של המערכת שלנו, כאמור נצטרך להוסיף רגיסטרים בכניסה ובמוצא ALU, לכן יצרנו קובץ VHDL חדש שעוטר את מערכת הALU עם הרגיסטרים שמוזנים מאותו השעון. נוסיף את קובץ הSDC הבא:


```
# Constrain clock port clk with a 20-ns requirement
create_clock -period 20 [get_ports clk]

# Automatically apply a generate clock on the output of phase-locked loops (PLLs)
# This command can be safely left in the SDC even if no PLLs exist in the design
derive_pll_clocks
```

Figure 8 הוספת שעון של 50MHz ב SDC

נבצע את הקימפול והסינתזה ללא השמה לפינים כפי שמבוקש (כולל השעון עצמו). לאחר ביצוע קימפול וסינתזה, תוכנת Quartus יודעת להביא לנו את התדר המקסימלי של המערכת כך שהתקבל –

Slow 1100mV 85C Model Fmax Summary				
 <<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	149.01 MHz	149.01 MHz	clk	

Slow 1100mV 0C Model Fmax Summary				
 <<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	150.31 MHz	150.31 MHz	clk	

נרצה לבצע אופטימיזציה של הקוד על מנת לקבל תדר טוב יותר, לכן ננסה להימנע מ Latches בקוד ובנוסף נשים בבחירת הפינים לכניסות והמוצאים את הפינים הקרובים ביותר שהתוכנה מציעה ב-fitter location. לאחר ביצוע השלבים הנ"ל נקבל את התדר המקסימלי הבא –

Slow 1100mV 85C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	158.48 MHz	158.48 MHz	clk	

Slow 1100mV 0C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	159.26 MHz	159.26 MHz	clk	

שמנו לב שכאשר נבצע השמה רק לCLK ושאר הפינים יישארו ללא השמה, נקבל תדר גבוה יותר (כ170 מגה).

בנוסף כפי שנלמד, ניתן להגדיל את תדר שעון המערכת באמצעות רכיב חומרה PLL שקיים בכרטיס עליו אנחנו מפתחים. לחומרת הPLL הכנסנו את שעון המערכת בעל תדר 50MHz והוצאנו שעון בעל תדר 100MHz. כך שהתקבל –

Slow 1100mV 85C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	216.83 MHz	216.83 MHz	PLLModule..ER divclk	

Slow 1100mV 0C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	218.77 MHz	218.77 MHz	PLLModule pl..UNTER divclk	

פירוט המערכת

בסעיף זה נסביר על המערכת שלנו ככלל ועל תתי המודולים שלה בפרט. עבור כל אחד ניתן סקירה קצרה על אופן פעילותו, נציג את הRTL שלו לאחר ביצוע הסינתזה, נפרט את הלוגיקה בה הוא משתמש, נמצא נתיב קריטי של פעילותו ונציג אותו בעזרת תוכנת Quartus.

מערכת הALU

סקירת פעולת המודול

נרצה לממש מערכת שמבצעת מספר מודולים שונים כאשר כל מודול מתבצע בנפרד לאחרים וקיים לו chip select שניתן על ידי המשתמש ובוחר באיזה מודול להשתמש. להלן המערכת שנרצה לממש –

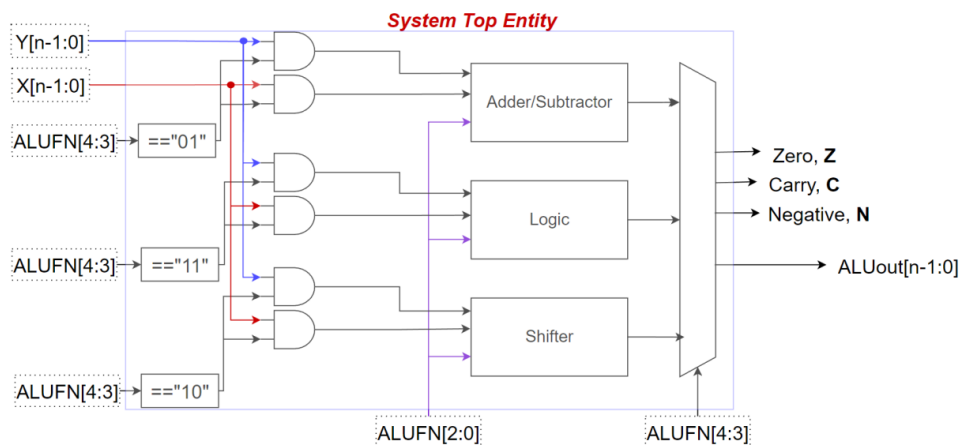


Figure 9 מודל המערכת

כאשר נסביר על כל חלק בנפרד בחלקים הבאים. המערכת שלנו תקבל 3 כניסות –

- אות כניסה X.
- אות כניסה Y.
- קו בקרה ALUFN כאשר ביטים 3,4 קובעים את המודול הנבחר כך ש –
 - 01 הוא מודול AdderSub.
 - 10 הוא מודול Shifter.
 - 11 הוא מודול Logic.

בנוסף לכך, כל מודול יכול לפעול במספר אופנים לפי הקו הבקרה המתאים שנכנס אליו והוא מורכב מהביטים 0,1,2 של קו הבקרה ALUFN.

המערכת שלנו תוצא 4 מוצאים –

- דגלים Z (Zero), C (Carry), N (Negative) כאשר תוצאת כל אחד מהדגלים תהיה בהתאם לשם שלו.
- מוצא המערכת בהתאם למודול הנבחר בסיגנל ALUOut.

לפירוט נוסף ניתן להסתכל על הISA של המערכת שלנו עבור כניסות שונות, כאשר עבור כניסת מודול שלא קיימת (00) נרצה לשמור על אותו הערך כמו מקודם ועבור בחירת מודול קיים אך ביצוע פעולה לא קיימת נרצה שכל המוצאים יהיו 0.

שרטוט הRTL

להלן הentity של המודול –

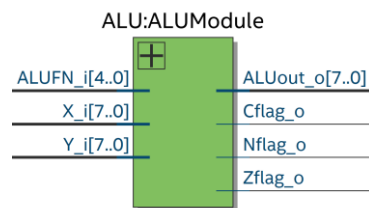


Figure 10 דיאגרמת בלוק של הALU

שרטוט הRTL של מודול זה –

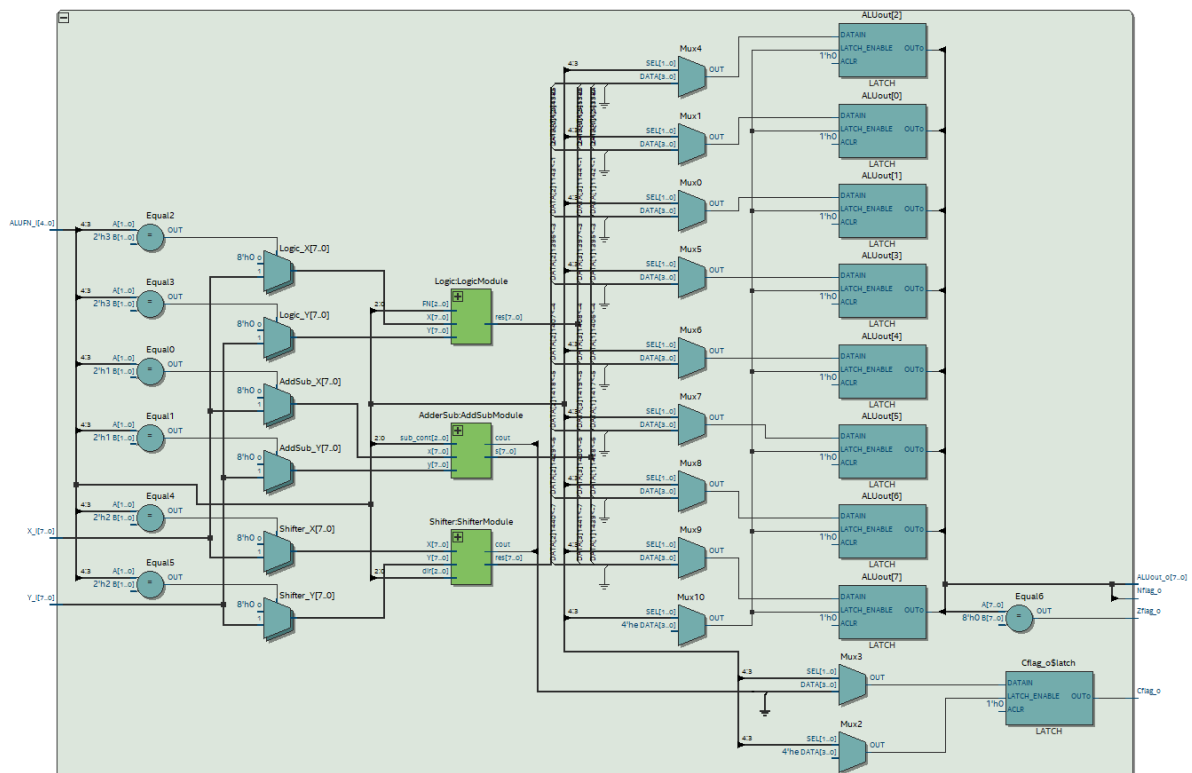


Figure 11 RTL של הALU

בשימוש בלוגיקה

כידוע אנחנו מפתחים קוד לוגי אשר משתמש באלמנטים לוגיים שניתן לקנפג אותם בהתאם לקוד, דבר שנעשה כחלק מתהליך הסינתזה. בניתוח זה נציג את הלוגיה עבור כל מודול כנדרש –

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	88
2		
3	Combinational ALUT usage for logic	120
1	-- 7 input functions	1
2	-- 6 input functions	51
3	-- 5 input functions	26
4	-- 4 input functions	11
5	-- <=3 input functions	31
4		
5	Dedicated logic registers	32
6		
7	I/O pins	33
8		
9	Total DSP Blocks	0
10		
11	Total PLLs	1
1	-- PLLs	1
12		
13	Maximum fan-out node	ALUFN_i[4]
14	Maximum fan-out	47
15	Total fan-out	684
16	Average fan-out	3.12

Figure 12 System Logic Usage

נתיב קריטי

הנתיב הקריטי חשוב ביותר להבנת זרימת המידע במערכת. מפני שלרכיבים יש השהייה גם אם הם מקביליים אזי עלינו לקחת בחשבון את אורך הזמן שלוקח למערכת להתייצב לפני הכנסת כניסה חדשה. להלן מציאת הנתיב הקריטי במודול זה –

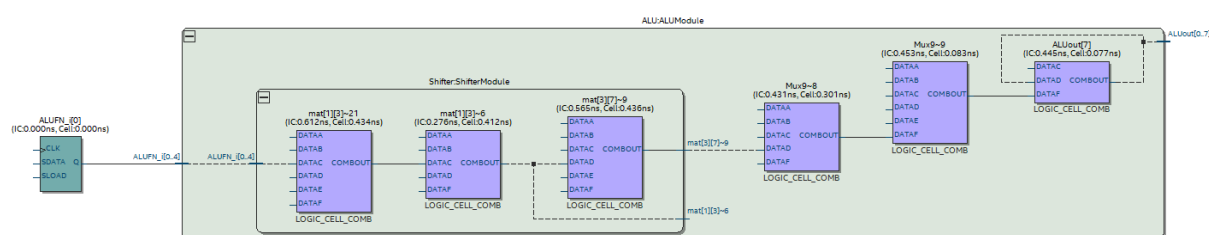


Figure 13 ALU Critical Path

נשים לב כי קיבלנו שהנתיב הקריטי עבור דרך המודול SHIFTER כפי שהיינו מצפים עקב הלוגיקה הרבה שהוא לוקח ביחס למודולים האחרים.

סקירת פעולת המודול

מודול זה מבצע בהתאם לכניסת קו הבקרה ALUFN א חיבור א חיסור בין 2 סיגנלים X,Y באורכים שווים על ידי FA א פעולת NEG המבצעת היפוך ביט-ביט עבור הסיגנל של X. הפעולות מתבצעות בהתאם לקו הבקרה באופן הבא –

פעולה	ALUFN[2: 0]
חיבור הכניסות X,Y	000
חיסור הכניסות X,Y	001
פעולת NEG על כניסת X	010
הוצאת וקטור אפסים	others

ביצענו את המודל הזה בVHDL על ידי הגדרת X_temp, Y_temp שהם יהיו הכניסות לכל אחד מהFA והם יוגדרו בצורה מקבילית כך שX_temp יהיה תוצאת הXOR כאשר נרצה לבצע חיבור או חיסור, יהיה היפוך כאשר נרצה לבצע NEG ויהיה 0 אחרת (כדי שהמוצא יהיה 0 כי התקבלה כניסה שגויה). כמו כן, Y_temp יהיה מוגדר להיות Y כאשר נרצה לבצע חיבור או חיסור ווקטור אפסים אחרת, מפני שגם בNEG וגם כאשר מתקבלת כניסה שגויה נרצה להוציא וקטור אפסים. לאחר מכן, נבצע שרשור של n פעמים של FA עם כניסות מתאימות מתוך וקטורים X_temp, Y_temp.

שרטוט הRTL

להלן entity של המודול –

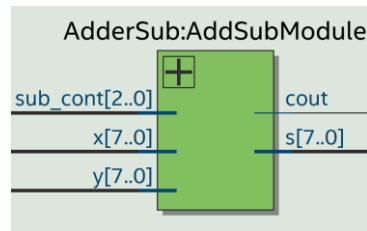


Figure 14 דיגאגרמת בלוק של AdderSub

שרטוט הRTL של מודול זה –

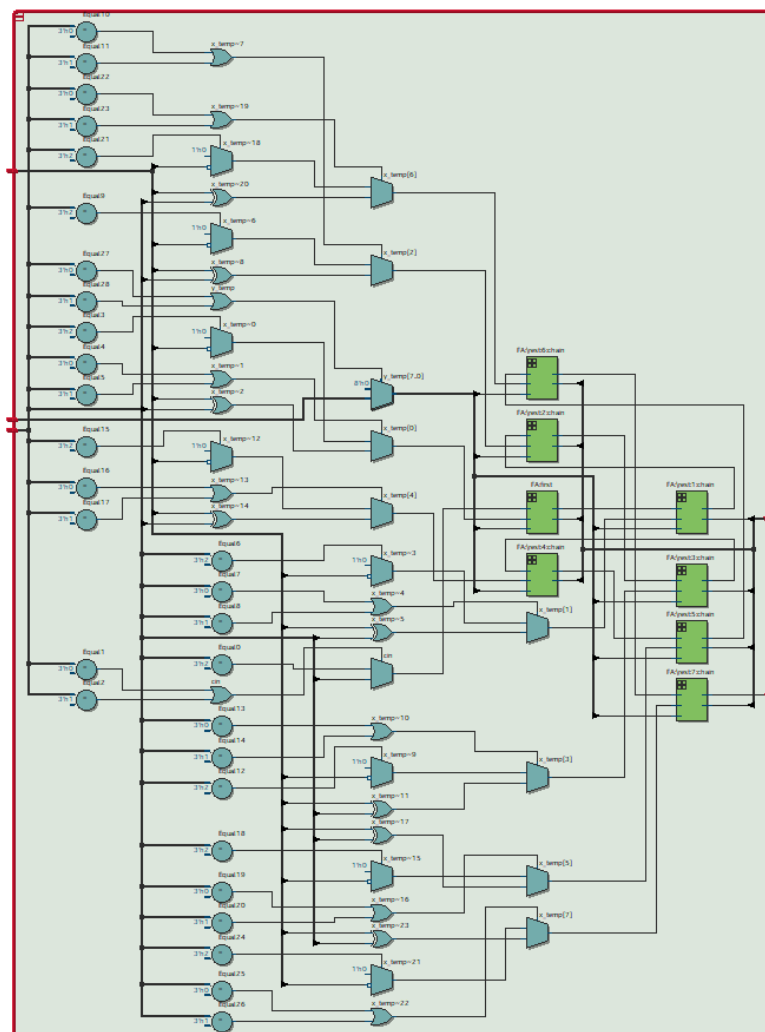


Figure 15 ה-RTL של AdderSub

בשימוש בלוגיקה

כידוע אנחנו מפתחים קוד לוגי אשר משתמש באלמנטים לוגיים שניתן לקנפג אותם בהתאם לקוד, דבר שנעשה כחלק מתהליך הסינתזה. בניתוח זה נציג את הלוגיה עבור כל מודול כנדרש –

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	17
2		
3	▼ Combinational ALUT usage for logic	28
1	-- 7 input functions	0
2	-- 6 input functions	5
3	-- 5 input functions	9
4	-- 4 input functions	7
5	-- <=3 input functions	7
4		
5	Dedicated logic registers	0
6		
7	I/O pins	28
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	sub_c...input
12	Maximum fan-out	18
13	Total fan-out	161
14	Average fan-out	1.92

Figure16 AdderSub Logic Usage

נתיב קריטי

הנתיב הקריטי חשוב ביותר להבנת זרימת המידע במערכת. מפני שלרכיבים יש השהייה גם אם הם מקביליים אזי עלינו לקחת בחשבון את אורך הזמן שלוקח למערכת להתייצב לפני הכנסת כניסה חדשה. להלן מציאת הנתיב הקריטי במודול זה –

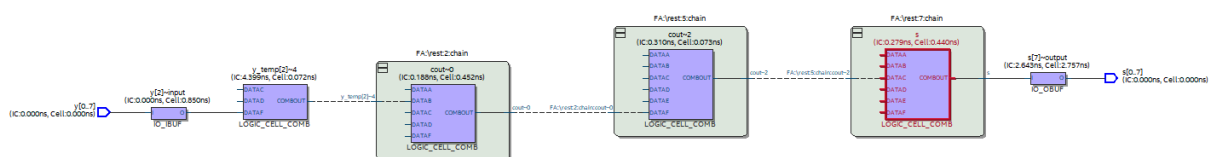


Figure 17 הנתיב הקריטי של ה-AdderSub

מודול Shifter

סקירת פעולת המודול

מודול זה מבצע בהתאם לכניסת ALUFN הזזה מבוססת Barrel-Shifter, כאשר אם ALUFN בביטים 0,1,2 הם '000' אזי נבצע הזזה שמאלה ואם '001' נבצע הזזה ימינה. החלטנו לממש את המודול באמצעות מעבר על כל k השכבות (כאשר $k = \log_2 n$) ועבור כל שכבה נבצע הזזה בהתאם כך שאם הביט בשכבה הרלוונטית ב X הוא 0 לא נבצע הזזה ואז מוצא השכבה יהיה כמו הכניסה ואחרת נבצע הזזה בהתאם לשכבה.

נדגיש שבמידה והוגדר לבצע הזזות ימינה אזי נרצה לבצע הפיכה עבור השורה הראשונה והאחרונה וכך להשיג את התוצאה המתאימה. כמו כן, בכדי למצוא את carry שיוצא מהמודול יצרנו סיגנל מתאים שמכיל את carry בכל שלב ונחזיר את האיבר האחרון שלו.

שרטוט הRTL

להלן entity של המודול –

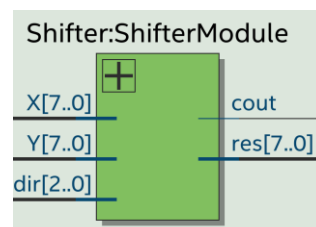


Figure 18 דיאגרמת בלוק של ה-Shifter

שרטוט הRTL של מודול זה –

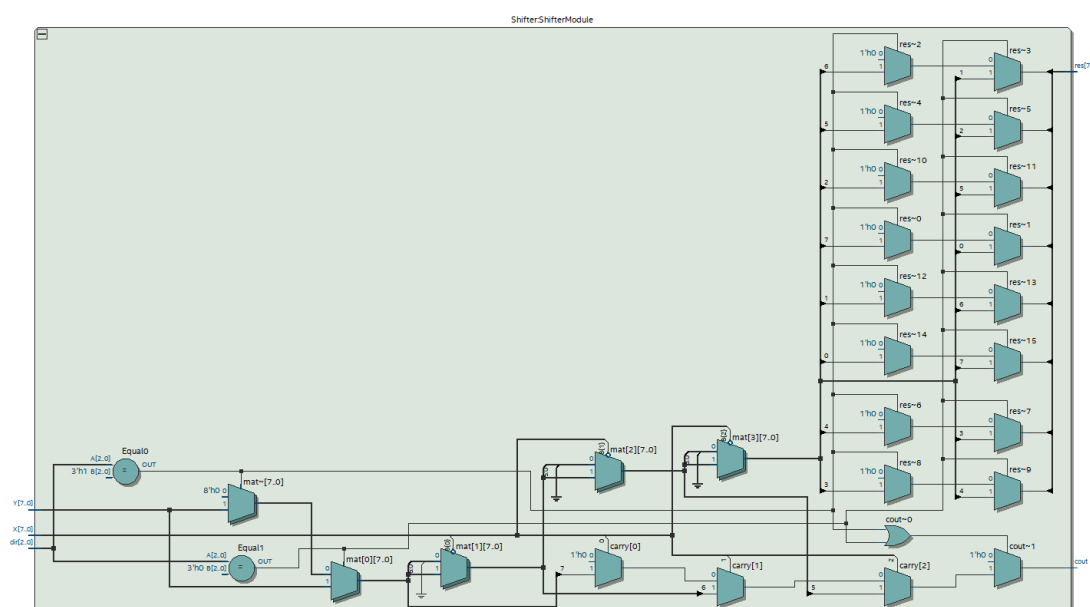


Figure 19 ה-RTL של ה-Shifter

בשימוש בלוגיקה

כידוע אנחנו מפתחים קוד לוגי אשר משתמש באלמנטים לוגיים שניתן לקנפג אותם בהתאם לקוד, דבר שנעשה כחלק מתהליך הסינתזה. בניתוח זה נציג את הלוגיה עבור כל מודול כנדרש –

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	24
2		
3	Combinational ALUT usage for logic	35
1	-- 7 input functions	1
2	-- 6 input functions	11
3	-- 5 input functions	6
4	-- 4 input functions	14
5	-- <=3 input functions	3
4		
5	Dedicated logic registers	0
6		
7	I/O pins	28
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	dir[0]~input
12	Maximum fan-out	20
13	Total fan-out	203
14	Average fan-out	2.23

Figure 20 Shifter Logic Usage

נתיב קריטי

הנתיב הקריטי חשוב ביותר להבנת זרימת המידע במערכת. מפני שלרכיבים יש השהייה גם אם הם מקביליים אזי עלינו לקחת בחשבון את אורך הזמן שלוקח למערכת להתייצב לפני הכנסת כניסה חדשה. להלן מציאת הנתיב הקריטי במודול זה –

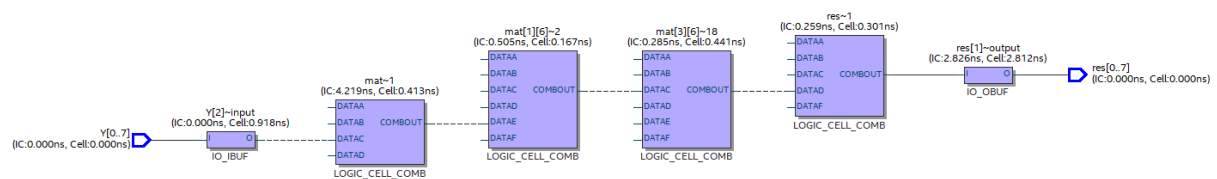


Figure 21 הנתיב הקריטי של ה-Shifter

מודול Logic

סקירת פעולת המודול

מודול זה נדרש לבצע פעולות לוגיות שונות על הסיגנלים X, Y בהתאם לכניסת ALUFN. את הפעולות הלוגיות האלו ניתן לבצע בVHDL כאשר הן יכולות להיות מסונתזות ולכן השתמשנו בהן בקלות כאשר בחרנו את הפעולה המתאימה בהתאם לביטים 0,1,2 בALUFN בהתאם ISA המצורפת.

שרטוט הRTL

להלן entity של המודול –

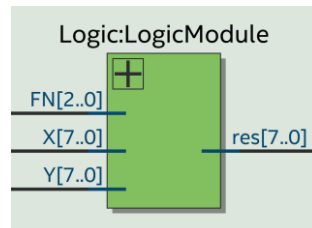


Figure 22 דיאגרמת בלוק של ה-Logic

שרטוט הRTL של מודול זה –

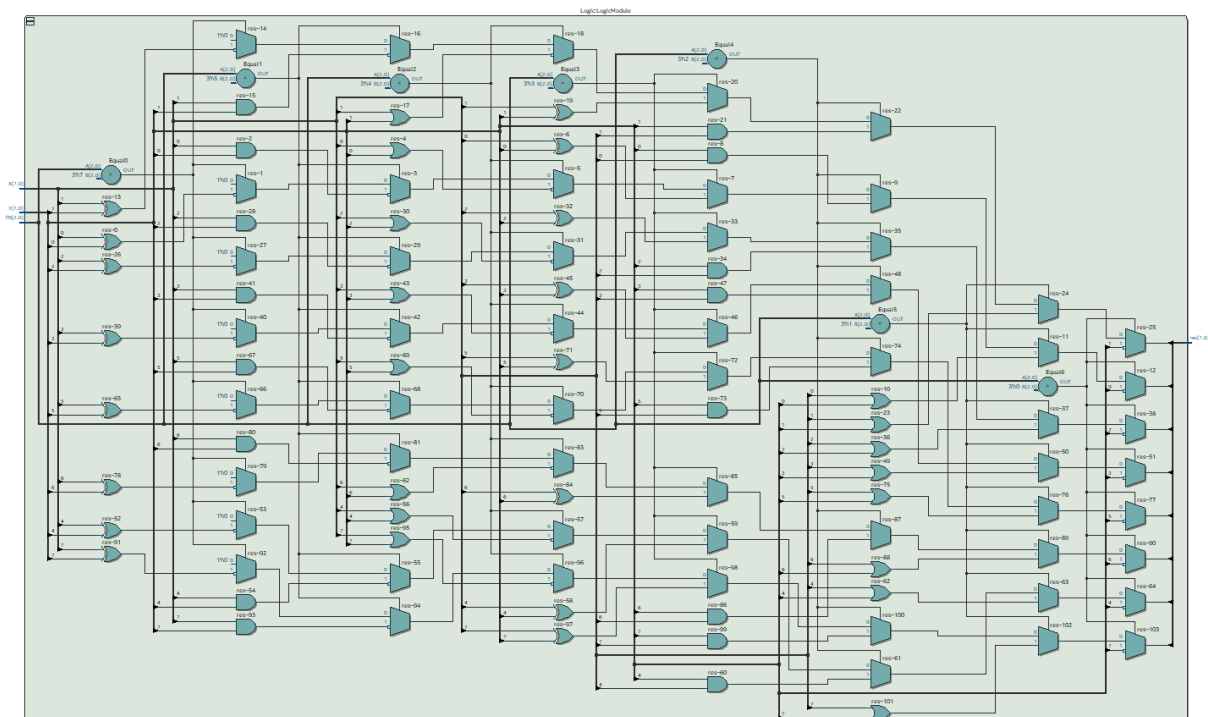


Figure 23 RTL של ה-Logic

בשימוש בלוגיקה

כידוע אנחנו מפתחים קוד לוגי אשר משתמש באלמנטים לוגיים שניתן לקנפג אותם בהתאם לקוד, דבר שנעשה כחלק מתהליך הסינתזה. בניתוח זה נציג את הלוגיה עבור כל מודול כנדרש –

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	4
2		
3	Combinational ALUT usage for logic	8
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	8
4	-- 4 input functions	0
5	-- <=3 input functions	0
4		
5	Dedicated logic registers	0
6		
7	I/O pins	27
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	FN[0]~input
12	Maximum fan-out	8
13	Total fan-out	75
14	Average fan-out	1.21

Figure24 Logic Module Logic Usage

נתיב קריטי

הנתיב הקריטי חשוב ביותר להבנת זרימת המידע במערכת. מפני שלרכיבים יש השהייה גם אם הם מקביליים אזי עלינו לקחת בחשבון את אורך הזמן שלוקח למערכת להתייצב לפני הכנסת כניסה חדשה. להלן מציאת הנתיב הקריטי במודול זה –

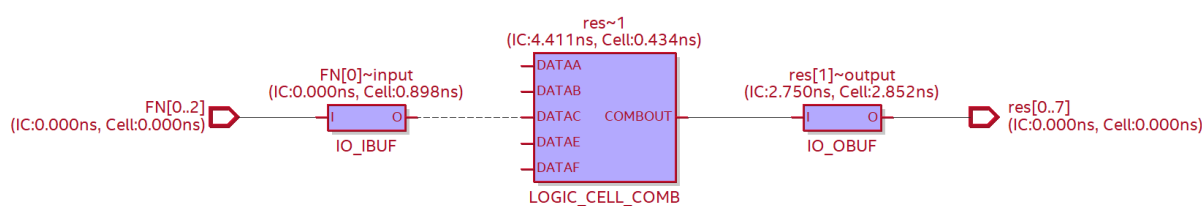


Figure 25 הנתיב הקריטי של ה-Logic

אופטימיזציה חומרית

בשלב האופטימיזציה עבור הקוד שלנו, הוספנו את מודול ה-PLL שקיים בכרטיס שלנו על מנת להגדיל את תדר העבודה וכך גם את תדר העבודה המקסימלי. כפי שהוזכר במציאת תדר מקסימלי, הוספת ה-PLL הביא לעליית התדר המקסימלי בכ-60MHz.

נזכור שהגדרנו את זמן המחזור בקובץ ה-SDC להיות $T = 20ns$ ולכן $f = 1/T = 1/20ns = 50MHz$. תדר שעון הכניסה ל-PLL הוא 50MHz בעוד שקבענו כי תדר המוצא אותו נזין את השעונים יהיה 100MHz.

Signal Tap

נרצה לבצע וורייקציה של החומרה על ידי פונקציית ה-*signal tap* של ה-*quartus*. נתפוס בזמן אמת את מצב הסיגנלים של הרכיב, ובהתאם לסיגנל שאותו נרצה לתפוס, ברגע שהסיגנל ישתנה למה שאנחנו רוצים נקבל את תוצאות הסיגנלים שנדפיס למסך.

נשים את הסיגנלים הבאים, כאשר הסיגנלים שאנחנו לוכדים הם ה-*Keys* שהם במצב *Pull Down* לכן נתפוס אותם בירידת מתח, ונדפיס את הכניסות והמוצאים של המערכת. בנוסף נשים את תנאי הלכידה כ-*Basic*, Or, כלומר שרק מספיק שאחד משתנה ולא כל הסיגנלים.

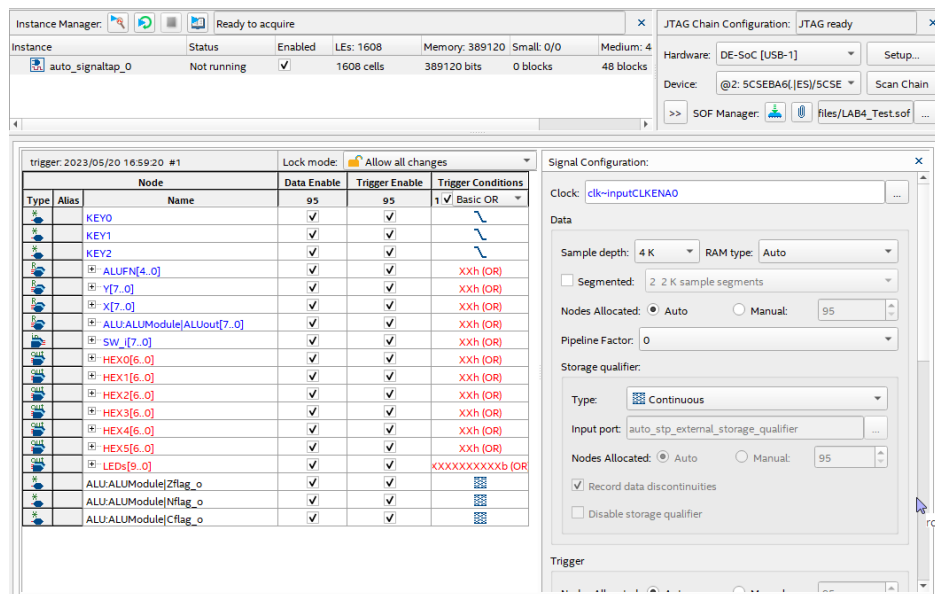


Figure 26 Signal Tap Configuration

תחילה עבור שינוי הערך של X :

Type	Alias	Name	-512	-256	0	256	512	768	1024	1280	1536
		KEY0									
		KEY1									
		KEY2									
		ALUFN[4..0]					00h				
		Y[7..0]					00h				
		X[7..0]			00h			06h			
		ALU:ALUModule ALUout[7..0]					00h				
		SW_i[7..0]					06h				
		HEX0[6..0]			40h			02h			
		HEX1[6..0]					40h				
		HEX2[6..0]					40h				
		HEX3[6..0]					40h				
		HEX4[6..0]					40h				
		HEX5[6..0]					40h				
		LEDs[9..0]					0000000100b				
		ALU:ALUModule Zflag_o									
		ALU:ALUModule Nflag_o									
		ALU:ALUModule Cflag_o									

Figure 27 שינוי X

כלומר אנחנו רואים, שברגע שלחצנו על $Key2$, ערך X ישתנה מ-0 ל-6 כפי שהגדרנו במתגים (SW_i)

נראה בדומה עבור Y :

Type	Alias	Name	-512	-256	0	256	512	768	1024	1280	1536
		KEY0									
		KEY1									
		KEY2									
		ALUFN[4..0]					00h				
		Y[7..0]			00h			02h			
		X[7..0]					06h				
		ALU:ALUModule ALUout[7..0]					00h				
		SW_i[7..0]					02h				
		HEX0[6..0]					02h				
		HEX1[6..0]					40h				
		HEX2[6..0]			40h			24h			
		HEX3[6..0]					40h				
		HEX4[6..0]					40h				
		HEX5[6..0]					40h				
		LEDs[9..0]					0000000100b				
		ALU:ALUModule Zflag_o									
		ALU:ALUModule Nflag_o									
		ALU:ALUModule Cflag_o									

Figure 28 שינוי Y

כלומר אנחנו רואים שבמתגים יש לנו את הערך 2, ולכן כאשר נלחץ על $Key0$ הערך של Y ישתנה ל-2.

כעת נראה שתי פעולות של ה- ALU , אחת חיבור והשנייה $Shift Left$:

פעולת חיבור על ידי 0100 :

Type	Alias	Name	-512	-256	0	256	512	768	1024	1280	1536
		KEY0									
		KEY1									
		KEY2									
		ALUFN[4..0]			00h			08h			
		Y[7..0]					02h				
		X[7..0]					06h				
		ALU:ALUModule ALUout[7..0]			00h			08h			
		SW_i[7..0]					06h				
		HEX0[6..0]					02h				
		HEX1[6..0]					40h				
		HEX2[6..0]					24h				
		HEX3[6..0]					40h				
		HEX4[6..0]			40h			00h			
		HEX5[6..0]					40h				
		LEDs[9..0]			0000000100b			0100000000b			
		ALU:ALUModule Zflag_o									
		ALU:ALUModule Nflag_o									
		ALU:ALUModule Cflag_o									

Figure 29 פעולת חיבור

ביצענו פעולת חיבור על ידי ה- $Opcode$: 0100, בין הערכים 2 ו-6 וקיבלנו במוצא אכן את הערך 8

פעולת *shifter* :

log: Trig @ 2023/05/20 17:56:30 (0:0:4.4 elapsed)			click to insert time bar								
Type	Alias	Name	-512	-256	0	256	512	768	1024	1280	1536
		KEY0									
		KEY1									
		KEY2									
		+ ALUFN[4..0]	00000b					10000b			
		+ Y[7..0]					00000100b				
		+ X[7..0]					00000001b				
		+ ALU.ALUModule[ALUout[7..0]	00000000b					00001000b			
		+ SW_[7..0]						10h			
		+ HEX0[6..0]						78h			
		+ HEX1[6..0]						40h			
		+ HEX2[6..0]						18h			
		+ HEX3[6..0]						40h			
		+ HEX4[6..0]	40h						00h		
		+ HEX5[6..0]						40h			
		+ LEDs[9..0]	0000000100b						1000000000b		
		ALU.ALUModule[Zflag_o									
		ALU.ALUModule[Nflag_o									
		ALU.ALUModule[Cflag_o									

Figure 30 פעולת SHL

בפעולה זו, נרצה לבצע הזזה שמאלה (לא ציקלית) על ידי ה- $Opcode = 10000$ של הווקטור Y כמות הפעמים של $X/2..0$. כאן כמות ההזזות היא 1 והערך של Y הוא 00000100 ולכן נצפה לקבל במוצא 00001000 ואכן קיבלנו תוצאה זו בסימולציה



Figure 31 ג'ייסון אחרי המעבדה

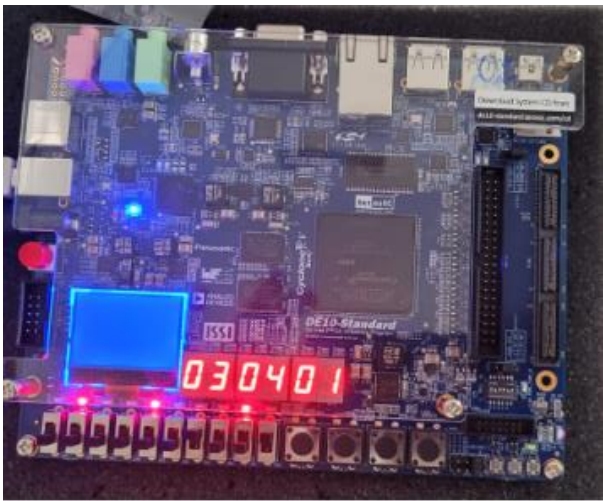


Figure 32 פעולת חיבור Y-X