

Preparation Report LAB4

**ADVANCED CPU ARCHITECTURE AND HARDWARE
ACCELERATORS LAB**

361.1.4693

Danel Barsheshet 209471242

Nachman Mimoun 321730558

תוכן עניינים

3.....	מטרת המעבדה.....
3.....	בדיקת ביצועים.....
4.....	סימולציית MODELSIM.....
6.....	מציאת תדר מקסימלי.....
7.....	פירוט המערכת.....
7.....	מערכת Top.....
11.....	מערכת ALU.....
13.....	מערכת PWM.....
16.....	מודל ADDERSUB.....
19.....	מודל SHIFTER.....
22.....	מודל LOGIC.....
25.....	חומרה.....
25.....	SINGAL TAP.....

מטרת המעבדה

במעבדה זו רכשנו מיומנויות בשימוש בתוכנת Quartus, תוך התמקדות בביצוע סינתזה למודלים שפיתחנו במעבדה הקודמת ובמעבדה הזו. את תהליך הסינתזה יישמנו על כרטיס DE10-Standard המצויד ב Cyclone V-FPGA.

בדיקת ביצועים

מטרת בדיקה זו הייתה לבחון את הביצועים הראשוניים של ה-ALU שפיתחנו במעבדה הקודמת ובנוסף ל-PWM שפיתחנו במעבדה זו, אך הפעם תוך שילוב סינתזה על כרטיס FPGA אמיתי. מאחר שמערכת ה-ALU שפיתחנו היא א-סינכרונית, כלומר פועלת ללא שעון, נדרשנו להוסיף למערכת רגיסטרים סינכרוניים בכניסה ובמוצא כדי לאפשר ניתוח ביצועים מבוסס זמן.

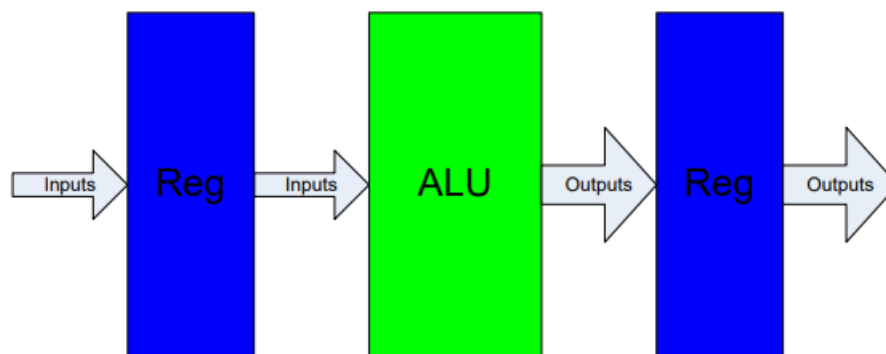


Figure 1 - ALU performance test case

כאשר לרגיסטרים אלו נחבר את השעון המותקן על הכרטיס-

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

Figure 2 - Clock Description

סימולציית ModelSim

בסימולציה זו נתחיל בביצוע (TB) Test Bench ובבחן את צורות הגל עבור המערכת השלמה, ולאחר מכן נתמקד בכל אחד מתתי-המודולים בנפרד. להלן מוצגים הגלים עבור המערכת כולה,

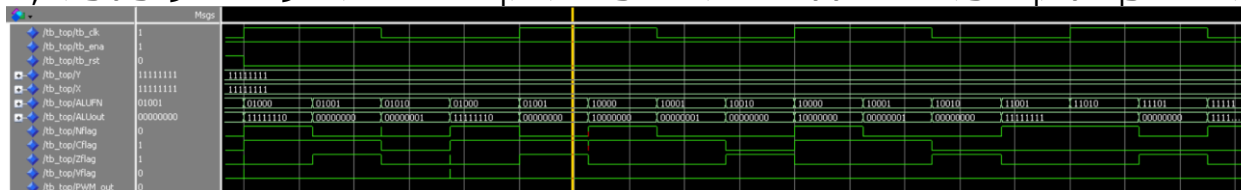


Figure 3 - סימולציה של כל המערכת

עבור המודול ALU

ניתוח תוצאות סימולציית הגלים עבור מודול זה מראה כי-

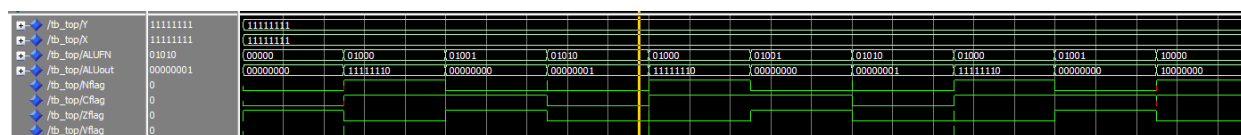


Figure 4- סימולציה של ה- ALU

כאשר נאתחל את X לערך 1- ואת Y לערך 1-1. במהלך הסימולציה ה- ALUFN מקבל פקודות שונות כל 100 ננו-שניות מתוך זיכרון המטמון (cache) שאותחל מראש בקובץ חיצוני.

לדוגמא – עבור ALUFN השווה ל- '01001' פעולת חיסור נקבל תוצאה של 0 כמצופה.

עבור המודול PWM

ניתוח תוצאות סימולציית הגלים עבור מודול זה מראה כי-

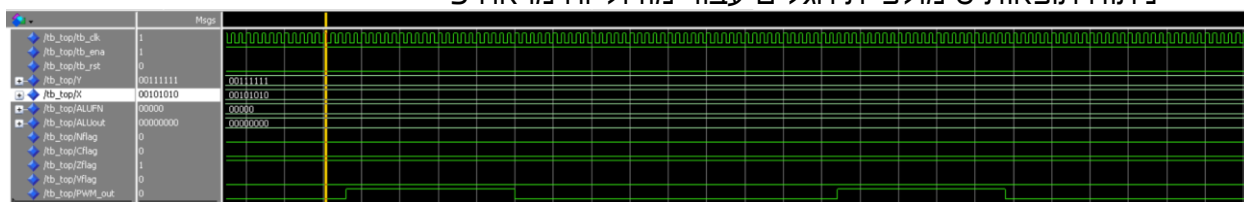


Figure 5- סימולציה של ה- PWM

כאשר נאתחל את X ל- 00101010 ואת Y ל- 00111111. וכאשר במהלך הסימולציה ה- ALUFN מקבל פקודת '00000' (PWM Output) נקבל במוצא גל ריבועי PWM_out אשר נדלק כאשר ה- counter מגיע לערך X ונכבה כאשר הוא מגיע לערך Y.

עבור המודול AdderSub

ניתוח תוצאות סימולציית הגלים עבור מודול זה מראה כי-

	Msgs	
+ /addersub_tb/x	00000100	00000100 00000010 00000100 11111111 00000000
+ /addersub_tb/y	00000010	00000010 00000100 00000010 00000001 00000000
+ /addersub_tb/sub_...	000	000 001 000 001
+ /addersub_tb/cout	0	
+ /addersub_tb/s	00000110	00000110 00000010 11111110 00000000 00000000 00000000

Figure 6– סימולציית ADDERSUB

ניתן לראות כי שלושת הביטים האחרונים ב- ALUFN הם '001' \ '000', המציינים כי מתבצעת פעולת חיבור\חיסור. הפעולות המתבצעות הן חיבור בין 8 ל-4, וכן בין 1 ל-(-1). כתוצאה מכך, הפלט ALUout מציג את התוצאה 8 או 0, בהתאם לפעולות אלו.

ניתוח תוצאות סימולציית הגלים עבור מודול זה מראה כי-



עבור המודול *Logic* התקבל כי –

ניתוח תוצאות סימולציית הגלים עבור מודול זה מראה כי-



מציאת תדר מקסימלי

```
1 # Constrain clock port clk with a 20-ns requirement
2
3 create_clock -period 20 [get_ports clk]
4
5 # Automatically apply a generate clock on the output of phase-locked loops (PLLs)
6 # This command can be safely left in the SDC even if no PLLs exist in the design
7
8 derive_pll_clocks
9
10 # Constrain the input I/O path
11
12 #####set_input_delay -clock clk -max 3 [all_inputs]
13
14 #####set_input_delay -clock clk -min 2 [all_inputs]
15
16 # Constrain the output I/O path
17
18 #####set_output_delay -clock clk 2 [all_outputs] 2
```

הוספת שעון של 50MHZ

6

Slow 1100mV 0C Model Fmax Summary					Slow 1100mV 85C Model Fmax Summary				
<<Filter>>					<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note		Fmax	Restricted Fmax	Clock Name	Note
1	168.66 MHz	168.66 MHz	clk		1	167.45 MHz	167.45 MHz	clk	

Figure 10 – f_{max} ALU

ועבור ה- PWM:

Slow 1100mV 0C Model Fmax Summary					Slow 1100mV 85C Model Fmax Summary				
<<Filter>>					<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note		Fmax	Restricted Fmax	Clock Name	Note
1	216.97 MHz	216.97 MHz	clk		1	210.66 MHz	210.66 MHz	clk	

Figure 11 – f_{max} PWM

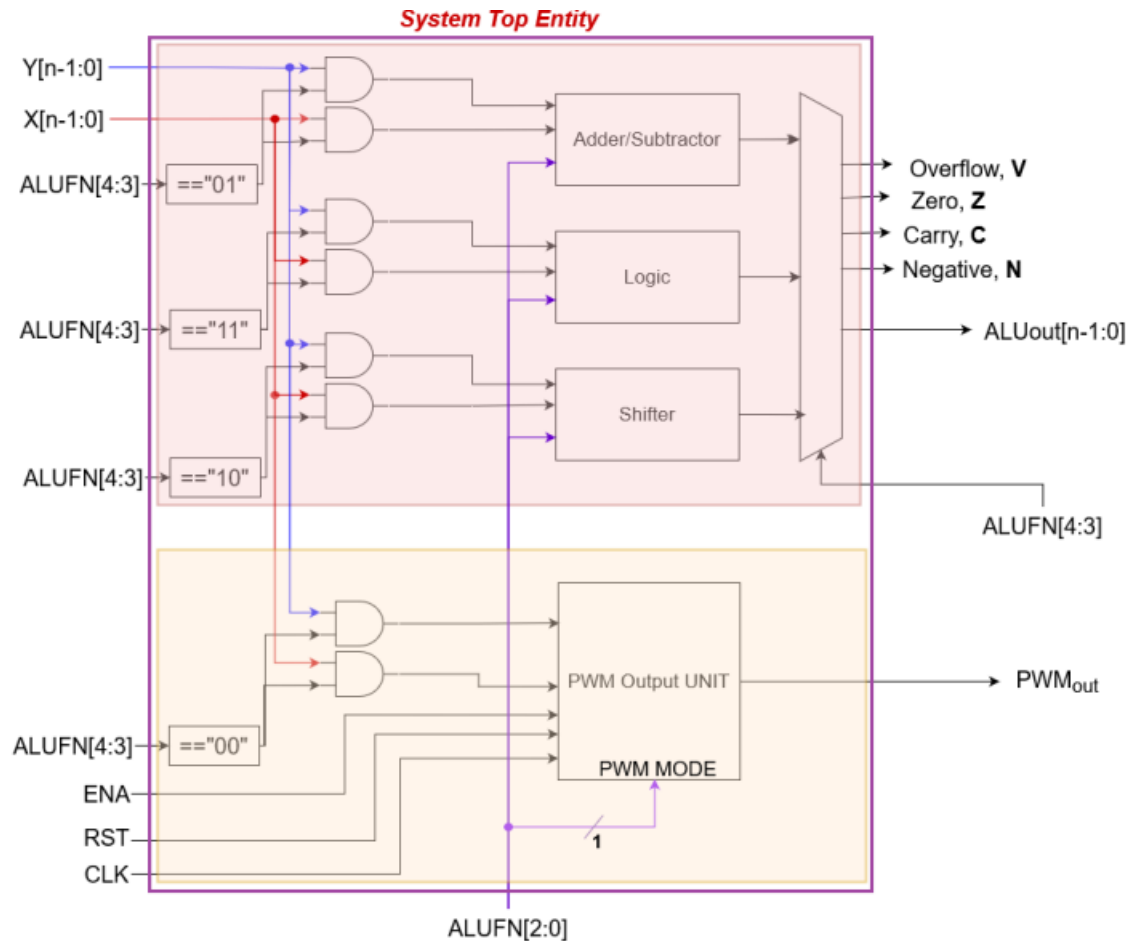
פירוט המערכת

בחלק זה נספק הסבר מקיף על המערכת שלנו באופן כללי, ולאחר מכן נתמקד בכל אחד מתתי-המודולים שלה. עבור כל רכיב במערכת, נציג סקירה תמציתית של אופן פעולתו, תצוגת ה-RTL שלו לאחר ביצוע הסינתזה, פירוט הלוגיקה המיושמת בו, זיהוי והצגת הנתבי הקריטי בפעולתו, וכן המחשה ויזואלית של הנתבי הקריטי באמצעות תוכנת Quartus.

מערכת ה- TOP

סקירת פעולת המודול

מטרתנו היא לפתח מערכת המורכבת ממספר מודולים שונים, כאשר כל מודול פועל באופן עצמאי ונפרד מהאחרים. המערכת תכלול מנגנון בחירה הנשלט על ידי המשתמש, המאפשר לבחור איזה מודול יופעל בכל רגע נתון. להלן תיאור המערכת שאנו מתכוונים לממש-



מודל כל המערכת - Figure 12

נפרט כעת את מבנה המערכת, כאשר הסבר מעמיק על כל חלק יינתן בסעיפים הבאים. המערכת שלנו מקבלת שלוש כניסות:

- אות כניסה X
- אות כניסה Y
- קו בקרה ALUFN, כאשר הביטים 3 ו-4 קובעים את המודול הנבחר באופן הבא :

- 00 מפעיל את מודול PWM
- 01 מפעיל את מודול AdderSub
- 10 מפעיל את מודול Shifter
- 11 מפעיל את מודול Logic

כל מודול יכול לפעול במספר אופנים, הנקבעים על ידי שלושת הביטים התחתונים (0, 1, 2) של קו הבקרה ALUFN.

המערכת מפיקה ארבעה פלטים:

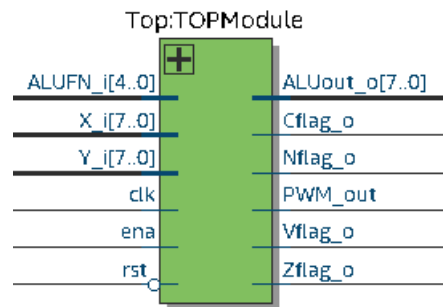
ארבעה דגלים V(Overflow), Z (Zero), C (Carry), N (Negative): כאשר כל דגל מציין את המצב המתאים לשמו.

פלט המערכת (ALUout) בהתאם למודול הנבחר.

פלט המערכת (PWMout) בהתאם למודול הנבחר ול- PWM Mode שנבחר .

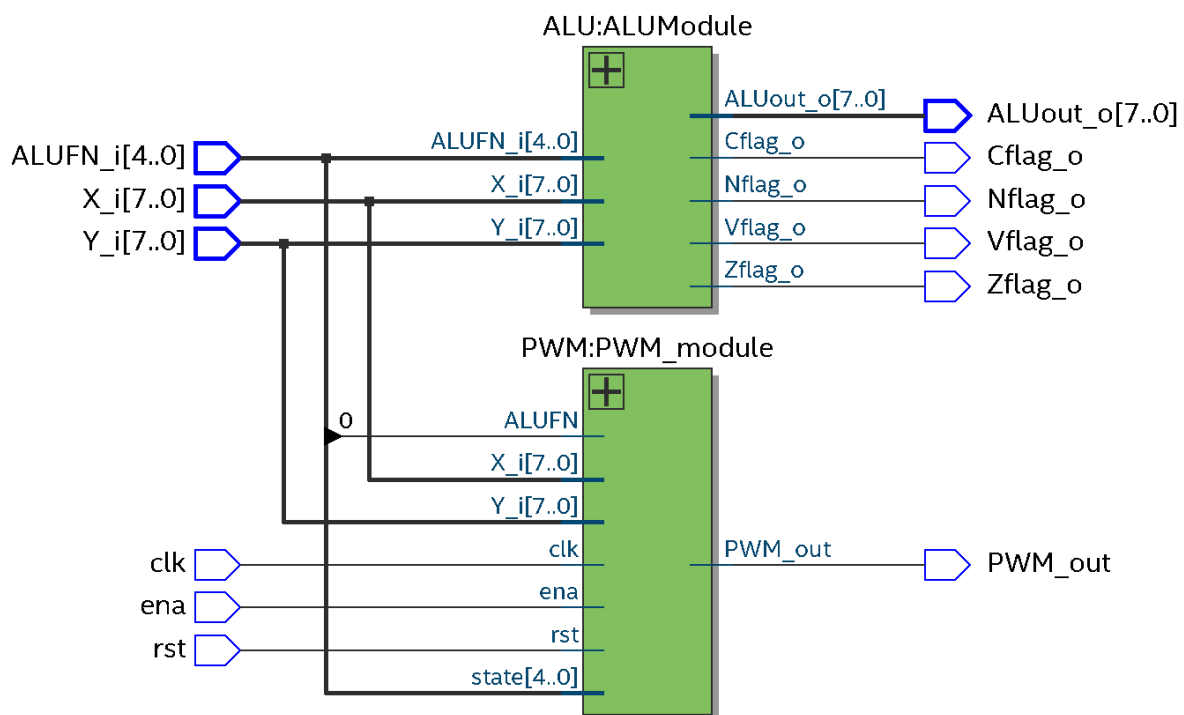
אם נבחר מודול קיים אך מבוצעת פעולה לא מוגדרת, כל הפלטים יהיו 0.

להלן תיאור ה- entity של המודול-



דיאגרמת בלוק של ה- TOP-13 Figure

להלן שרטוט ה- RTL של המודול -



RTL של ה- TOP-14 Figure

בשימוש בלוגיקה

כפי שידוע, אנו מפתחים קוד לוגי המשתמש באלמנטים לוגיים הניתנים לקונפיגורציה בהתאם לקוד שנכתב. תהליך זה מתבצע כחלק מתהליך הסינתזה. בניתוח הבא, נציג את הלוגיקה עבור כל מודול כנדרש-

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	99
2		
3	▼ Combinational ALUT usage for logic	161
1	-- 7 input functions	1
2	-- 6 input functions	35
3	-- 5 input functions	49
4	-- 4 input functions	24
5	-- <=3 input functions	52
4		
5	Dedicated logic registers	33
6		
7	I/O pins	37
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	ALUFN_i[1]~input
12	Maximum fan-out	51
13	Total fan-out	889
14	Average fan-out	3.32

Figure 15 - System Logic Usage

נתיב קריטי

הנתיב הקריטי הוא מרכיב חיוני להבנת זרימת המידע במערכת. מכיוון שלרכיבים יש השהיות, גם כאשר הם פועלים במקביל, עלינו להתחשב בפרק הזמן הדרוש למערכת להתייצב לפני הכנסת כניסה חדשה. להלן נציג את תהליך זיהוי הנתיב הקריטי במודול זה-

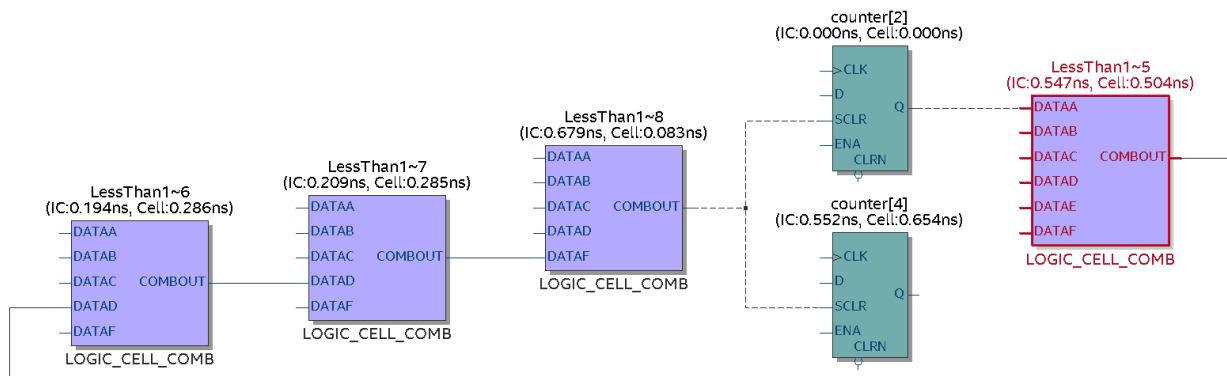


Figure 16 – TOP Critical Path

ראוי לציין כי הנתיב הקריטי שזוהה עובר דרך מודול ה-PWM, תוצאה התואמת את ציפיותינו. זאת בשל מורכבות ה-Counter- המיושמת במודול זה, אשר משמעותית יותר בהשוואה למודולים האחרים במערכת.

מודול ALU

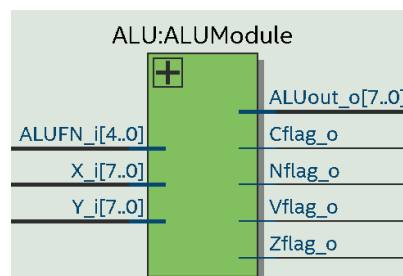
סקירת פעולת המודול

מודל זה מבצע פעולות אריתמטיות, לוגיות ופעולות הזהה בהתאם לכניסת ALUFN. כאשר השני ביטים הראשונים (MSB):

- '01' - פעולות חיבור, חיסור ו-neg.
- '10' - פעולות Shift ימינה ו-שמאלה.
- '11' - פעולות לוגיות.

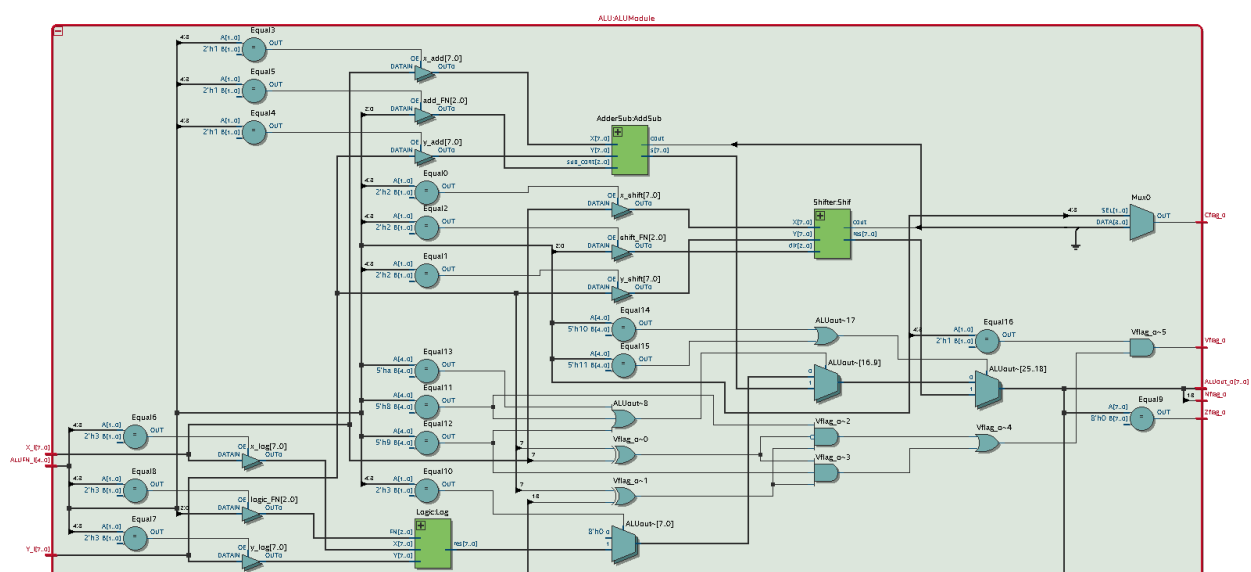
שרטוט ה-RTL

להלן תיאור ה-entity של המודול-



דיאגרמת בלוק של ה-ALU Figure 17-

להלן שרטוט ה-RTL של המודול -



RTL של ה-ALU Figure 18-

בשימוש בלוגיקה

כפי שידוע, אנו מפתחים קוד לוגי המשתמש באלמנטים לוגיים הניתנים לקונפיגורציה בהתאם לקוד שנכתב. תהליך זה מתבצע כחלק מתהליך הסינתזה. בניתוח הבא, נציג את הלוגיקה עבור כל מודול כנדרש-

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	71
2		
3	▼ Combinational ALUT usage for logic	110
1	-- 7 input functions	0
2	-- 6 input functions	31
3	-- 5 input functions	37
4	-- 4 input functions	22
5	-- <=3 input functions	20
4		
5	Dedicated logic registers	0
6		
7	I/O pins	33
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	ALUFN_i[1]~input
12	Maximum fan-out	51
13	Total fan-out	560
14	Average fan-out	3.18

Figure 19 - System Logic Usage

נתיב קריטי

הנתיב הקריטי הוא מרכיב חיוני להבנת זרימת המידע במערכת. מכיוון שלרכיבים יש השהיות, גם כאשר הם פועלים במקביל, עלינו להתחשב בפרק הזמן הדרוש למערכת להתייצב לפני הכנסת כניסה חדשה. להלן נציג את תהליך זיהוי הנתיב הקריטי במודול זה –

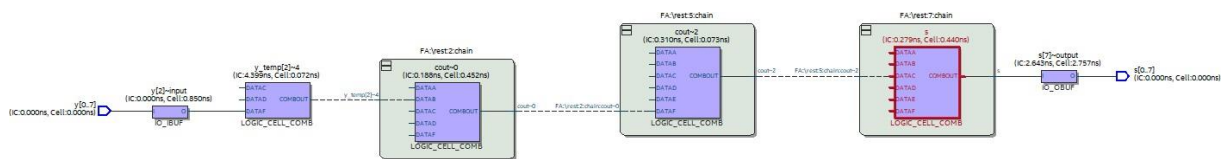


Figure 20 – TOP Critical Path

מודול PWM

סקירת פעולת המודול

- מודל זה מבצע הוצאת אות PWM בהתאם למס כניסות:
- **ALUFN** – כאשר שני הביטים הראשונים (MSB) הינם '00' המערכת תיכנס ל- Mode PWM. בנוסף כאשר הביט האחרון (LSB) היינו '0' ניכס למצב `set\rest` וכאשר הוא '1' נכנס למצב `rest\set`.
 - **X ו-Y** – כאשר כל אחד מהם הוא `Set\Rest` בהתאם ל- Mode שאנחנו נמצאים כשנגיע ל- `Conter` יתאפס.
 - **ENA** – מאפשר הוצאת אות PWM כאשר הוא '1' ולא מאפשר ב-'0'.
 - **RST** – מאפס את המוצא וה- `counter` כאשר הוא לחוץ ('1') ולא עושה דבר כשהוא '0'.
 - **CLK** – השעון שבו תעבוד המערכת וה- `Counter`.
- לפי הכניסות הנ"ל נדע איך לתת אות מוצא יחיד `PWM_out` המערכת מתוארת בגרף הנ"ל:

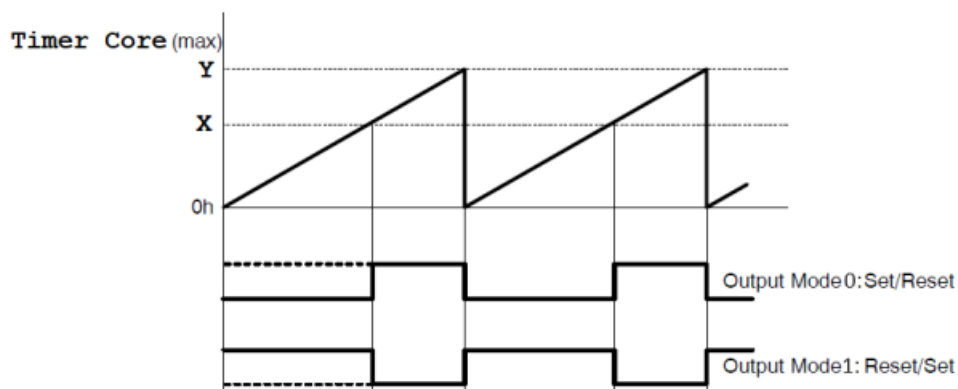
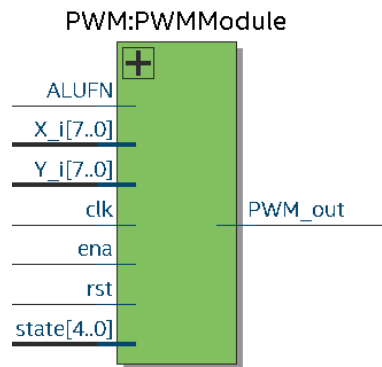


Figure 5: PWM output modes

Figure 20 - PWM output modes

שרטוט RTL



דיאגרמת בלוק של ה-PWM

להלן שרטוט ה-RTL של המודול -

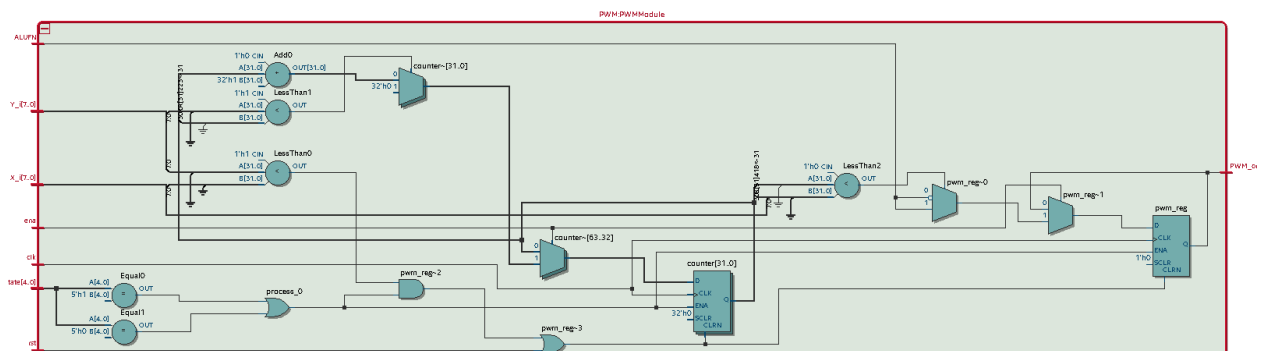


Figure 22-PWM של ה-RTL

בשימוש בלוגיקה

כפי שידוע, אנו מפתחים קוד לוגי המשתמש באלמנטים לוגיים הניתנים לקונפיגורציה בהתאם לקוד שנכתב. תהליך זה מתבצע כחלק מתהליך הסינתזה. בנייתו הבא, נציג את הלוגיקה עבור כל מודול כנדרש-

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	33
2		
3	▼ Combinational ALUT usage for logic	59
1	-- 7 input functions	0
2	-- 6 input functions	7
3	-- 5 input functions	11
4	-- 4 input functions	9
5	-- <=3 input functions	32
4		
5	Dedicated logic registers	33
6		
7	I/O pins	26
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	pwm_reg~0
12	Maximum fan-out	33
13	Total fan-out	386
14	Average fan-out	2.68

Figure 23 - System Logic Usage

נתיב קריטי

הנתיב הקריטי הוא מרכיב חיוני להבנת זרימת המידע במערכת. מכיוון שלרכיבים יש השהיות, גם כאשר הם פועלים במקביל, עלינו להתחשב בפרק הזמן הדרוש למערכת להתייצב לפני הכנסת כניסה חדשה. להלן נציג את תהליך זיהוי הנתיב הקריטי במודול זה-

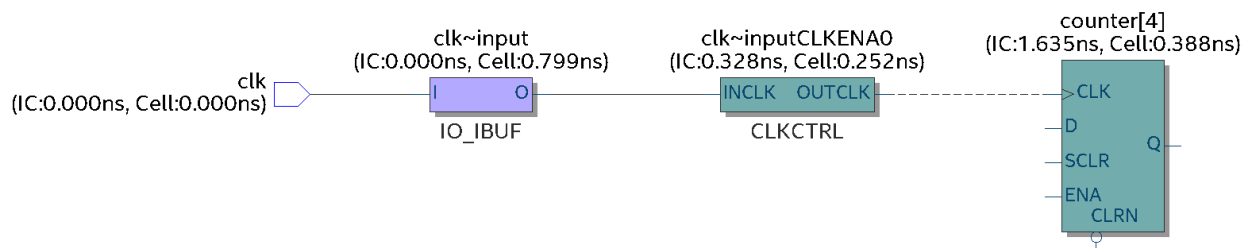


Figure 24 – PWM Critical Path

מודול AdderSub

סקירת פעולת המודול

מודול זה מבצע, בהתאם לכניסת קו הבקרה ALUFN, חיבור או חיסור בין שני האותות X ו Y, ו-שהם באורך זהה, באמצעות Full Adder (FA) או פעולת NEG המבצעת היפוך ביט-ביט עבור האות X. הבחירה בין הפעולות נעשית על פי קו הבקרה באופן הבא-

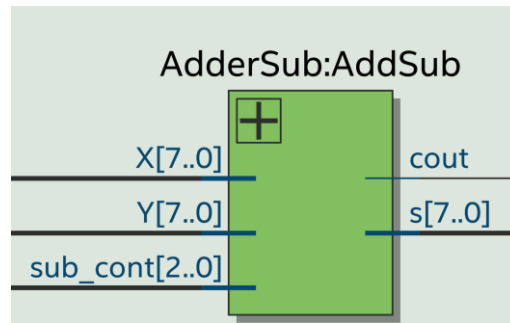
פעולה	ALUFN[2: 0]
חיבור הכניסות X,Y	000
חיסור הכניסות X,Y	001
פעולת NEG על כניסת X	010
הוצאת וקטור אפסים	others

במימוש המודול ב-VHDL, הגדרנו שני אותות זמניים X_sig ו Y_sig: אלה משמשים ככניסות לכל אחד מרכיבי ה- Full Adder (FA) ומוגדרים באופן מקבילי כדלהלן:

X_sig מוגדר כתוצאת פעולת XOR כאשר נדרש חיבור או חיסור, כהיפוך ביטים כאשר נדרשת פעולת NEG, וכאפס בכל מקרה אחר (להבטחת פלט אפס במקרה של כניסה שגויה). Y_sig מוגדר כזה ל- Y כאשר נדרש חיבור או חיסור, וכוקטור אפסים בכל מקרה אחר (כולל פעולת NEG וכניסה שגויה).

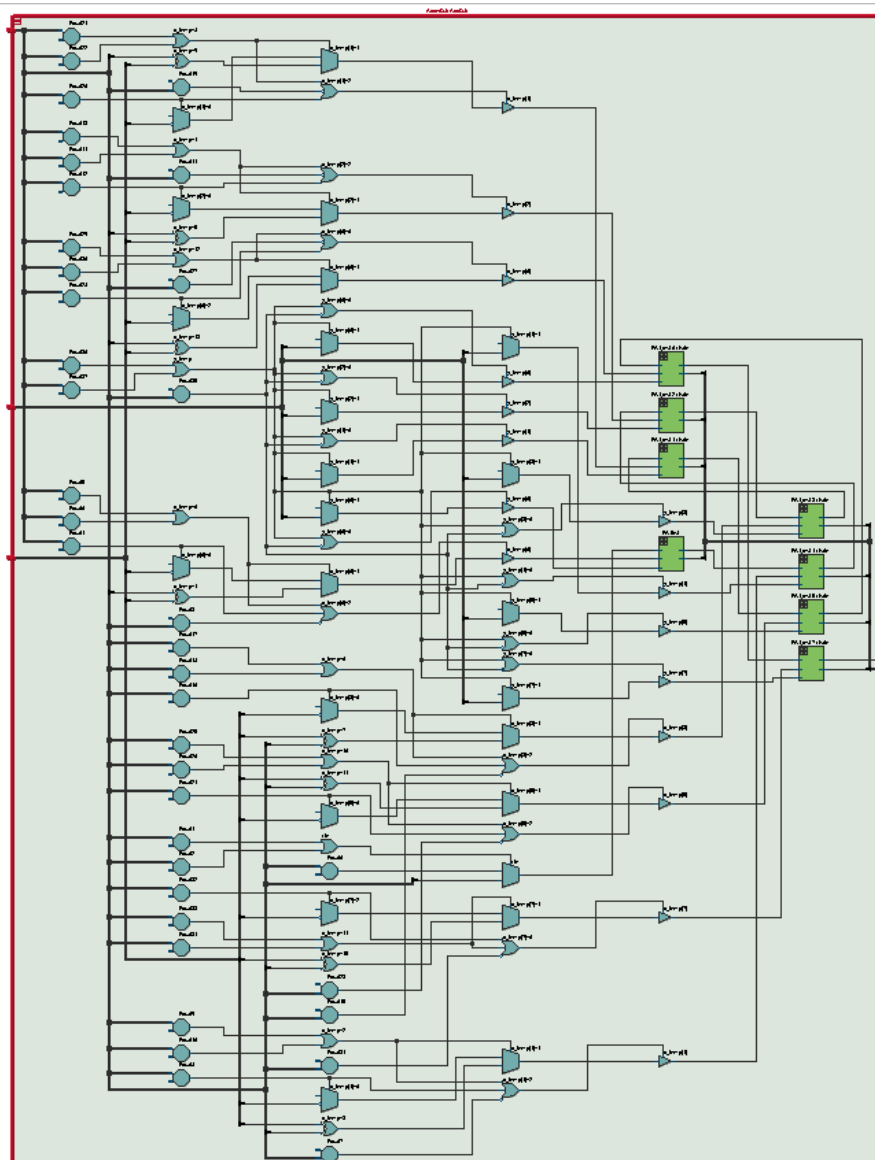
לאחר הגדרות אלו, אנו מבצעים שרשרת של n יחידות FA, כאשר לכל אחת מהן מוזנות הכניסות המתאימות מתוך הווקטורים X_sig ו Y_sig.

להלן תיאור ה- entity של המודול-



דיאגרמת בלוק של ה- AdderSub

להלן שרטוט ה- RTL של המודול -



RTL של ה- AdderSub

בשימוש בלוגיקה

כפי שידוע, אנו מפתחים קוד לוגי המשתמש באלמנטים לוגיים הניתנים לקונפיגורציה בהתאם לקוד שנכתב. תהליך זה מתבצע כחלק מתהליך הסינתזה. בנייתו הבא, נציג את הלוגיקה 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_cont[1]~input
12	Maximum fan-out	18
13	Total fan-out	161
14	Average fan-out	1.92

Figure 27 - System Logic Usage

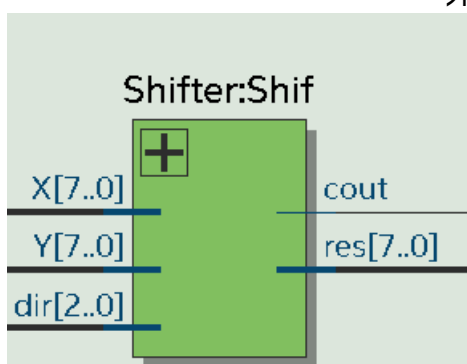
מודול Shifter

סקירת פעולת המודול

מודול זה מבצע הזזה מבוססת Barrel-Shifter בהתאם לכניסת ALUFN. כאשר הביטים 0, 1, 2 של ALUFN הם '000', מתבצעת הזזה שמאלה, ואילו כשהם '001', מתבצעת הזזה ימינה. המימוש נעשה באמצעות מעבר על k שכבות כאשר $K = \log_2(n)$ בכל שכבה מתבצעת הזזה בהתאם לערך הביט הרלוונטי ב- X : אם הביט הוא 0, לא מתבצעת הזזה והפלט זהה לכניסה; אחרת, מתבצעת הזזה בהתאם למיקום השכבה. חשוב לציין כי במקרה של הזזה ימינה, מתבצעת הפיכה של השורה הראשונה והאחרונה כדי להשיג את התוצאה הנכונה. בנוסף, לצורך מציאת ה- $carry$ היוצא מהמודול, יצרנו סיגנל ייעודי המכיל את ה- $carry$ בכל שלב, כאשר הערך האחרון שלו מוחזר כפלט.

שרטוט RTL

להלן תיאור ה-`entity` של המודול-



דיאגרמת בלוק של ה-`Shifter` Figure 28-

להלן שרטוט ה-RTL של המודול -

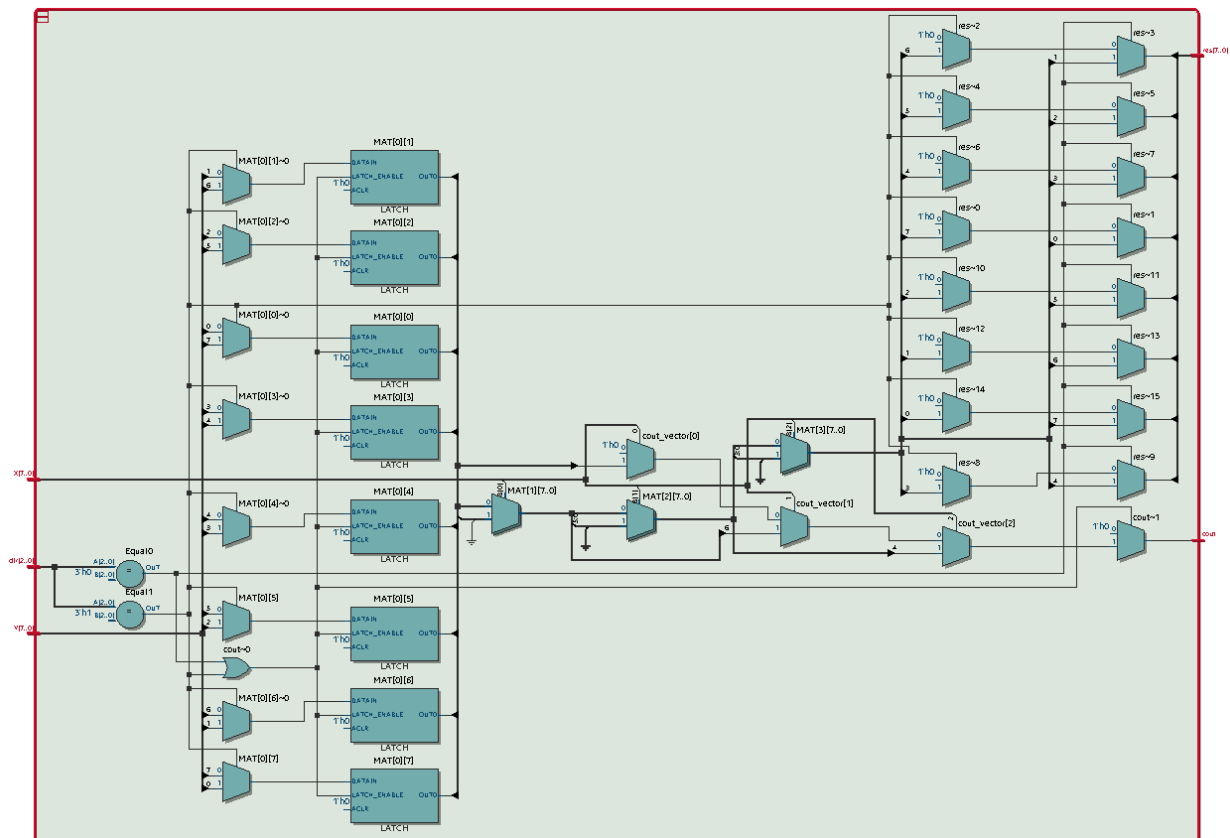


Figure 29-Shifters RTL

בשימוש בלוגיקה

כפי שידוע, אנו מפתחים קוד לוגי המשתמש באלמנטים לוגיים הניתנים לקונפיגורציה בהתאם לקוד שנכתב. תהליך זה מתבצע כחלק מתהליך הסינתזה. בניתוח הבא, נציג את הלוגיקה עבור כל מודול כנדרש-

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	27
2		
3	▼ Combinational ALUT usage for logic	37
1	-- 7 input functions	0
2	-- 6 input functions	16
3	-- 5 input functions	2
4	-- 4 input functions	2
5	-- ≤3 input functions	17
4		
5	Dedicated logic registers	0
6		
7	I/O pins	28
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	Equal0~0
12	Maximum fan-out	17
13	Total fan-out	201
14	Average fan-out	2.16

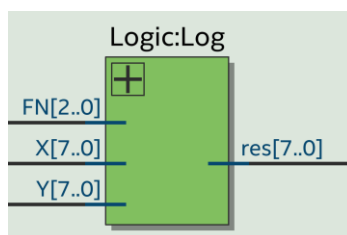
Figure 30 - System Logic Usage

סקירת פעולת המודול

מודול זה מיועד לביצוע פעולות לוגיות מגוונות על האותות X ו-Y, כאשר הפעולה הספציפית נקבעת על ידי כניסת ה-ALUFN. פעולות לוגיות אלו ניתנות לביצוע ישיר ב-VHDL, והן מסונתזות בקלות. לפיכך, יישמנו אותן באופן פשוט, כאשר הבחירה בפעולה המתאימה נעשית על פי שלושת הביטים התחתונים (0, 1, 2) של ALUFN, בהתאם ל-ISA המצורפת.

שרטוט ה-RTL

להלן תיאור ה-entity של המודול-



דיאגרמת בלוק של ה-Logic-31 Figure

להלן שרטוט ה-RTL של המודול -

בשימוש בלוגיקה

כפי שידוע, אנו מפתחים קוד לוגי המשתמש באלמנטים לוגיים הניתנים לקונפיגורציה בהתאם לקוד שנכתב. תהליך זה מתבצע כחלק מתהליך הסינתזה. בניתוח הבא, נציג את הלוגיקה 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[2]-input
12	Maximum fan-out	8
13	Total fan-out	75
14	Average fan-out	1.21

Figure 33 - System Logic Usage

חומרה

בשלב האופטימיזציה של הקוד שלנו, שילבנו את מודול ה-PLL הקיים בכרטיס כדי להקטין את תדר העבודה.
חשוב לזכור כי בקובץ ה-SDC הגדרנו את זמן המחזור להיות $T = 20\text{ns}$ ולכן תדר השעון המקורי הוא $f = 1/T = 1/20\text{ns} = 50\text{MHz}$. בהתאם לכך, תדר שעון הכניסה ל-PLL נקבע ל-50 MHz בעוד שקבענו את תדר המוצא, אשר מוזן לשעונים במערכת, ל-2.5 MHz.
מציאת השעון אנחנו נכנסים לקואנטר אשר עולה ב-Overflow בביט השישי ולכן עבור כניסה של 2.5 MHz נקבל מוצא של 31.25 KHz.

Signal Tap

אנו מתכננים לבצע וריפיקציה של החומרה באמצעות פונקציית ה-Signal Tap של Quartus. תהליך זה יאפשר לנו ללכוד בזמן אמת את מצב הסיגנלים של הרכיב. בהתאם לסיגנל שנבחר ללכידה, ברגע שהסיגנל ישתנה לערך הרצוי, נקבל את ערכי הסיגנלים שנבחר להציג על המסך.
נגדיר את הסיגנלים הבאים ללכידה: ה-Keys שהם במצב Pull Down, ולכן נלכוד אותם בירידת מתח. נציג את ערכי הכניסות והמוצאים של המערכת. בנוסף, נגדיר את תנאי הלכידה כ-Basic OR, כלומר, מספיק ששינוי יתרחש באחד מהסיגנלים כדי להפעיל את הלכידה, ולא נדרש שינוי בכל הסיגנלים.

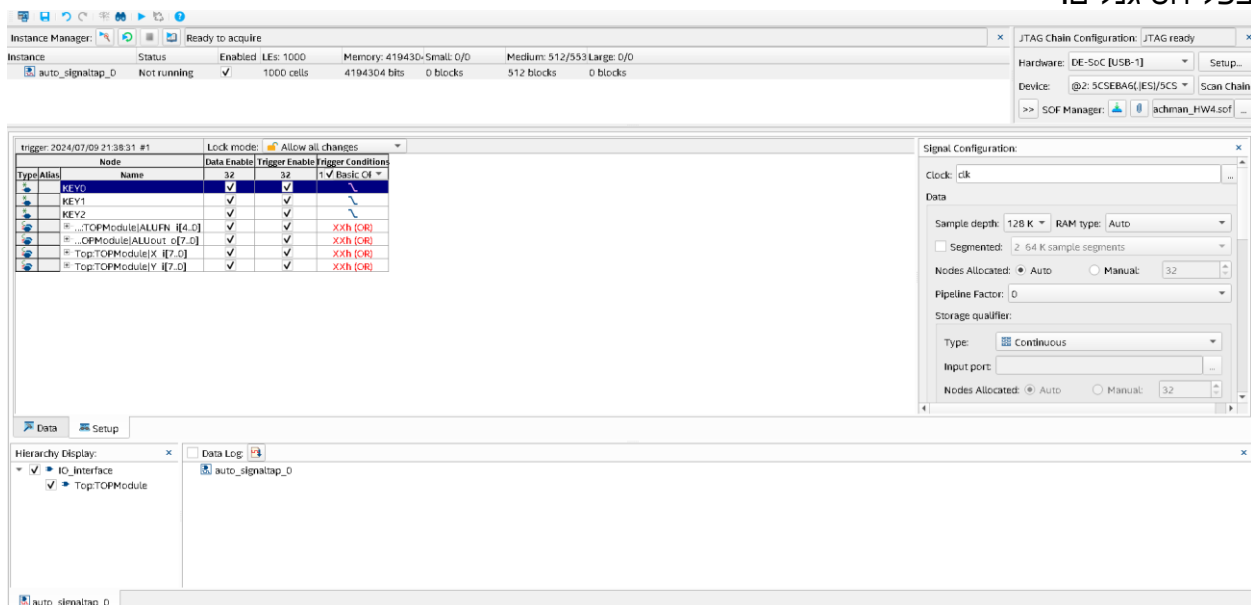


Figure 34 – Signal Tap Configuration

תחילה, נבחן את התוצאות עבור שינוי בערך של X:

log: Trig @ 2024/07/09 21:46:11 (D:0:11:2 elaps)																			click to insert time bar																		
Type/Alias		Name		-16384		-8192		0		8192		16384		24576		32768		40960		49152		57344		65536		73728		81920		90112		98304		106496		114688	
KEY0																																					
KEY1																																					
KEY2																																					
* Top:TOPModule ALU FN [4:0]																						00h															
* Top:TOPModule ALU out [7:0]																						00h															
* Top:TOPModule X [7:0]																						00h		00h													
* Top:TOPModule Y [7:0]																						00h															

Figure 34 – X change

כלומר, אנו מבחינים כי ברגע שלחצנו על Key2, ערך X השתנה מ-0 ל-4, בהתאם להגדרה שקבענו במתגים (SW_i).
באופן דומה, נבחן את התוצאות עבור Y :

log: Trig @ 2024/07/09 21:48:58 (D:0:6:4 elaps)																		click to insert time bar																	
Type/Alias	Name	16384	8192	0	8192	16384	24576	32768	40960	49152	57344	65536	73728	81920	90112	98304	106496	114688																	
KEY0																																			
KEY1																																			
KEY2																																			
	* Top:TOPModule ALU FN [4:0]										00h																								
	* Top:TOPModule ALU out [7:0]										00h																								
	* Top:TOPModule X [7:0]										04h																								
	* Top:TOPModule Y [7:0]		04h									08h																							

Figure 35 – X change

כלומר, אנו מבחינים כי במתגים מוגדר הערך 8, ולכן כאשר נלחץ על Key0, הערך של Y משתנה ל-8.
כעת נבחן שתי פעולות של ה: ALU-האחת היא פעולת חיבור, והשנייה היא פעולת הזזה שמאלה (Left Shift).
נתחיל בפעולת החיבור, המיוצגת על ידי הקוד 01000:

log: Trig @ 2024/07/09 21:50:26 (D:0:6:2 elaps)										click to insert time bar									
Type/Alias	Name	16384	8192	0	8192	16384	24576	32768	40960	49152	57344	65536	73728	81920	90112	98304	106496	114688	
	KEY0																		
	KEY1																		
	KEY2																		
	Top:TOPModule ALU FN [4:0]		00h									08h							
	Top:TOPModule ALU out [7:0]		00h									0Ch							
	Top:TOPModule X [7:0]											08h							
	Top:TOPModule Y [7:0]											00h							

Figure 36 – Adder Operation X+Y

ביצענו פעולת חיבור באמצעות ה- Opcode 01000 בין הערכים 4 ו-8. כצפוי, קיבלנו במוצא את הערך 12.

פעולת Shifter:

log Trig @ 2024/07/09 21:53:24 (0:01.9 elapsed)																			click to insert time bar																		
Type/Alias	Name	16384	8192	0	8192	16384	24576	32768	40960	49152	57344	65536	73728	81920	90112	98304	106496	114688																			
KEY0																																					
KEY1																																					
KEY2																																					
TOPModule ALUEN [4.0]	Obin																																				
OPModule ALUout_o[7.0]	Obin																																				
TopTOPModule X [7.0]																																					
TopTOPModule Y [7.0]																																					

Figure 37 – Shifter operation

בפעולה זו, אנו מבצעים הזזה ימינה (לא מעגלית) באמצעות ה- Opcode = 10001. ההזזה מתבצעת על הווקטור Y, כאשר מספר ההזזות נקבע על ידי $X[2..0]$. במקרה הנוכחי, מספר ההזזות הוא 2, והערך של Y הוא 00001000. לפיכך, אנו מצפים לקבל במוצא את הערך 00000010. ואכן, בסימולציה קיבלנו את התוצאה הצפויה זו.

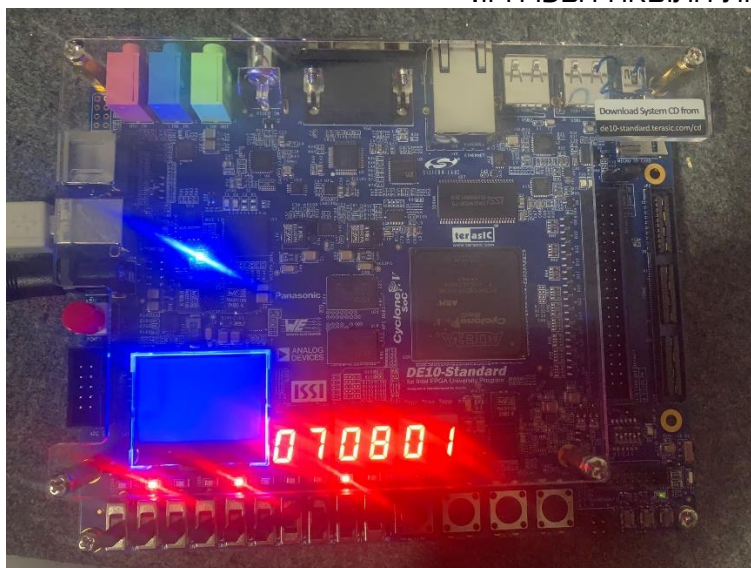


Figure 38 – X-Y Operation