

# **LAB 4**

## **ADVANCED CPU ARCHITECTURE AND HARDWARE ACCELERATORS LAB**

**Ido Ron: 322384330**

**Binyamin Oumanski: 322688946**

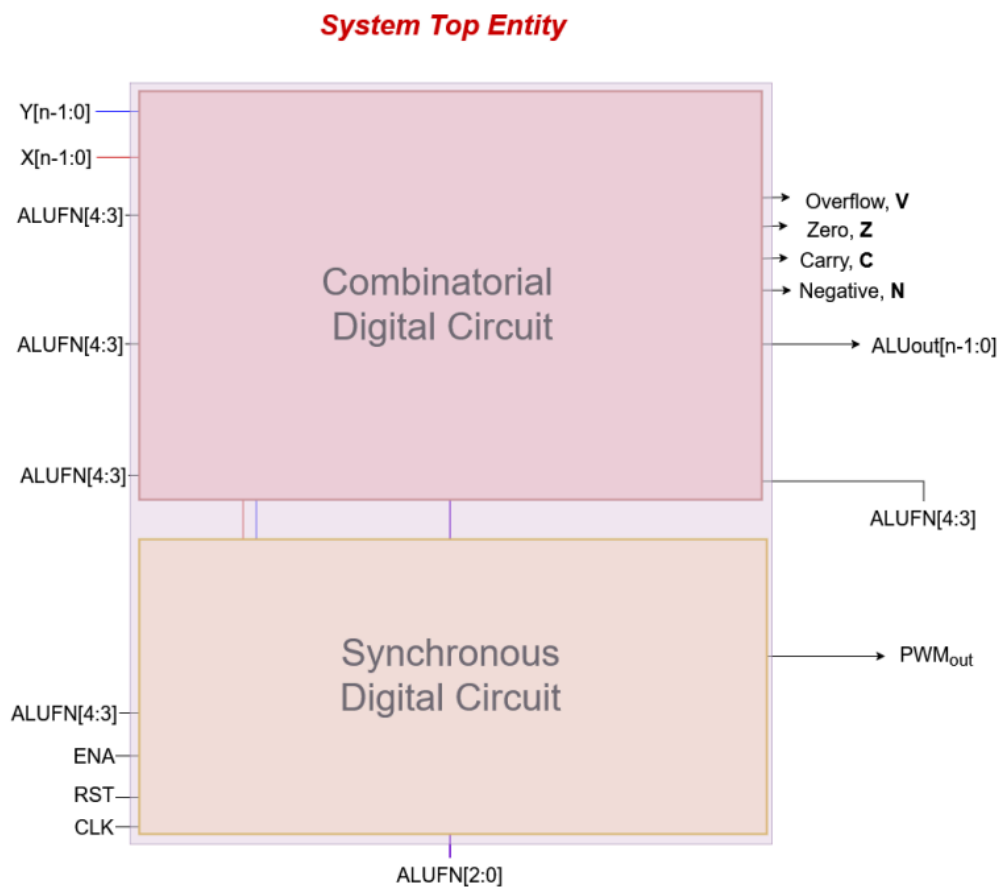
## מטרת המעבדה

במעבדה זו נלמד לעבוד בתוכנת Quartus כך שנבצע סינתזה של קוד VHDL ונצורב אותו על FPGA מדגם DE2 - 115.

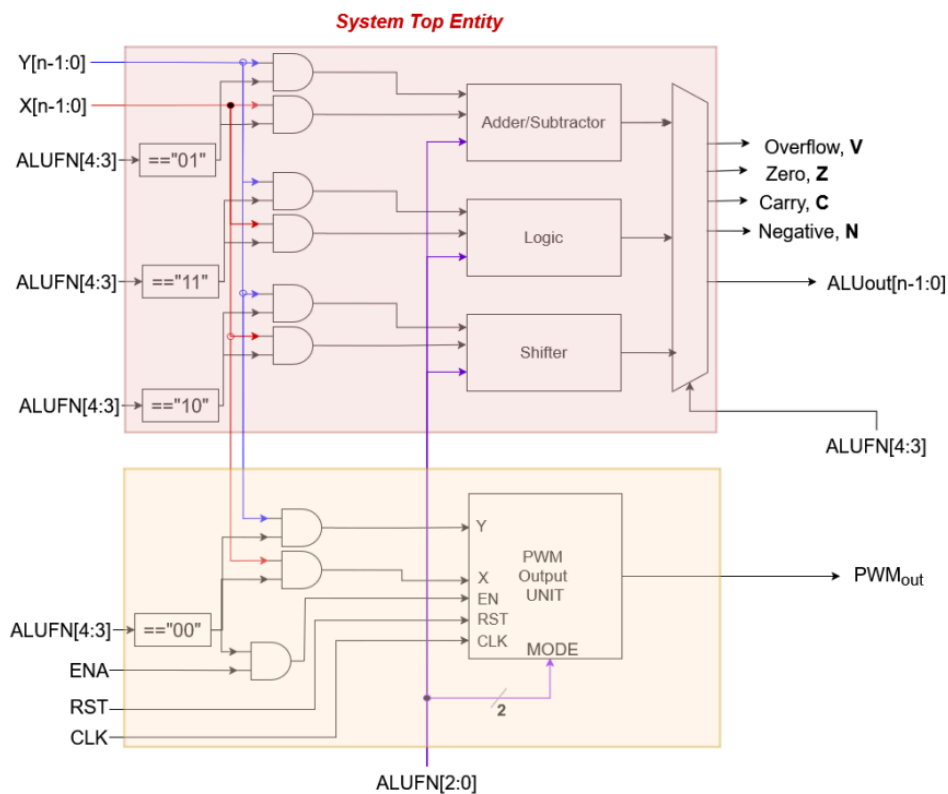
## פירוט המערכת

### Digital System

הרכיב הדיגיטלי מורכב משני תת רכיבים: CDC, SDC. הוא יקבל ככניסות וקטורי X ו-Y שעליהם יבוצעו הפעולות, ALUFN שמגדיר את הפעולה הנדרשת, שעון, RST לאיפוס המערכת, ו-ENA שמאפשר את ריצת גל ה-PWN. המוצאים יהיו FLAGS תוצאת ה-ALU והגל PWNOUT.



רכיב ה- CDC (הורוד באיור) כולל בתוכו את תתי הרכיבים שבנינו במעבדה 1: ADDERSUB, LOGIC ו- SHIFTER שזה בעצם ה-ALU. השינוי היחיד הוא שהוספנו את פעולת SWAP בתוך רכיב ה-ADDERSUB.



רכיב ה-SDC (הצהוב באיור הקודם) בעצם מוציא גל PWM שתלוי במונה שמתאפס בהגעת ערכו לערך Y ובעל 3 מצבים:

מצב 0: המוצא יהיה 1 כאשר המונה בין X ו-Y, יהיה 0 אחרת.

מצב 1: המוצא יהיה 0 כאשר המונה בין X ו-Y, יהיה 1 אחרת.

מצב 2: בהגעת המונה לערך X ערך המוצא יתהפך (מ-0 ל-1 ולהפך).

Figure 4: Digital System subparts architecture

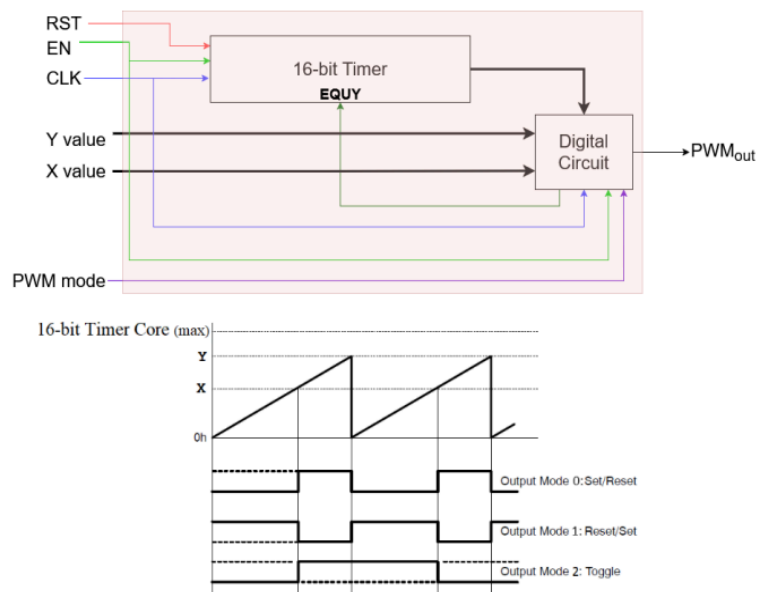


Figure 5: PWM output unit architecture

## TOP

רכיב ה-TOP כולל בתוכו את ה-DIGITAL SYSTEM וגם את ניהול הכניסות והמוצאים. המטרה היא ליצור קשר בין ה-INPUT של המשתמש מה-FPGA וזה נעשה ע"י הכנסת ערכים ל-ALUFN, X, Y בעזרת מיתוג ה-SWITCHES וקביעת האם הכניסה הינה 8 הביטים התחתונים או העליונים בעזרת SW9. המוצאים יתממשקו בסופו של דבר אל מסכי ה-HEX ונורות ה-LED.

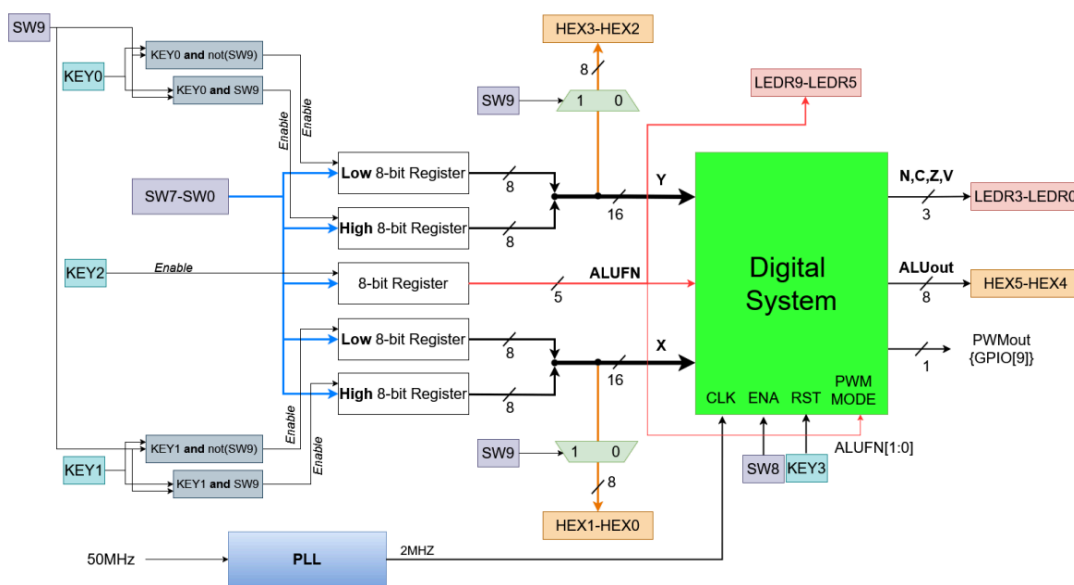


Figure 6: Digital system with I/O interface

בנוסף נשתמש במאיץ PLL לצורך הגברת מהירות השעון שיכול לספק לנו הגביש המובנה ב-FPGA.

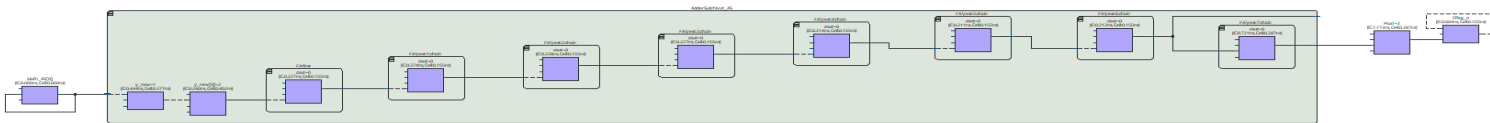
## נתיב קריטי:

הנתיב הקריטי מגדיר את הזמן הארוך ביותר שדרוש לאות לעבור בין שני איזורים במחזור שעון אחד ולכן קובע את תדירות השעון המקסימלית. אם הנתיב הקריטי ארוך מדי, ייתכנו הפרות של תנאי ה-Setup/Hold ברישומים, והתפקוד ישתבש.

כדי לייעל נהוג לקצר אותו באמצעות הוספת רישומים ביניים (pipelining) או שימוש ברכיבים חומרתיים ייעודיים. חשוב לשמור על נתיב קריטי קצר כדי ליצור מערכת מהירה, אמינה ויעילה מבחינת צריכת משאבים וחשמל.

אצלנו בעבודה כדי למצוא את הנתיב הקריטי סמנו 2 רגיסטרים שיעטפו את המערכת, וכך למצוא את הנתיב הארוך ביותר ביניהם.

להלן הנתיב הקריטי בALU :



## מציאת תדר מקסימלי:

נדרשנו למצוא תדר מקסימלי של המערכת, לכן שמנו 2 רגיסטרים שיעטפו את הALU מהכניסה והמוצא.

לאחר מכן קימפלנו והQUARTUS הביא לנו את התדרים הבאים במעלות: 0 ו85 צלזיוס.

- התדר המקסימלי בכל טבלה הוא השני.

Slow 1200mV 85C Model Fmax Summary			
<<Filter>>			
	Fmax	Restricted Fmax	Clock Name
1	67.79 MHz	67.79 MHz	DigitalSystem:portmap_top ALUFN_i_s[3]
2	92.85 MHz	92.85 MHz	hivutPLL altpll_component auto_generated pll1 clk[0]

Slow 1200mV 0C Model Fmax Summary			
<<Filter>>			
	Fmax	Restricted Fmax	Clock Name
1	74.67 MHz	74.67 MHz	DigitalSystem:portmap_top ALUFN_i_s[3]
2	102.29 MHz	102.29 MHz	hivutPLL altpll_component auto_generated pll1 clk[0]

## signal tap

ביצענו וריפיקציה של ה-FPGA על ידי Quartus בשימוש של signal tap. המטרה של הפעולה היא לבדוק את הסיגנלים של המערכת בזמן אמת.

נדרשנו לבדוק את פעולה אחת אריתמטית ואחת של shift.

לכל אחת מהפעולות נבדוק את :

Node	
Type	Alias
	Name
	⊕ DigitalSystem:portmap top X i[15..0]
	⊕ DigitalSystem:portmap top Y i[15..0]
	⊕ ALUFN[4..0]
	⊕ DigitalSystem:portmap top ALUout o[7..0]

כלומר על המערכת נכניס את X ו Y באמצעות לחיצה על הswitches הנכונים.

נשים לב שבפעולות האלו מה שרלוונטי אילו רק 8 הביטים הlsb של X ו Y.

את X נטען על ידי לחיצה על ה switches ובנוסף ש 9 switch מכובה , לאחר מכן נלחץ על 0 key.

את X נטען על ידי לחיצה על ה switches ובנוסף ש 9 switch מכובה , לאחר מכן נלחץ על 1 key.

**פעולה ADD :**

log: Trig @ 2025/06/03 17:41:16 (0:0)		click to insert time bar				
Node		1	2	3	4	
Type	Alias	0	1	2	3	4
	⊕ ...ortmap top X i[15..0]	FFFFh	X		FE0Ch	
	⊕ ...ortmap top Y i[15..0]			FF03h		
	⊕ ALUFN[4..0]		1Fh	X	0Ch	08h
	⊕ ...p top ALUout o[7..0]		00h		X	0Eh

\* ההערה: הערך ההתחלתי של X לפני הטעינה של הערך היה הערך FFFF.



טענו ל X את הערך FF0C ולY את FF03 . \*הערה : מה שרלוונטי זה 8 ביטים הLSB כלומר 2 הספרות הימניות.

לאחר מכן כדי להשתמש בADD לחצנו על הswitches לקבל את הערך 8 ולחצנו על 3.key.

ואכן ניתן לראות שבתוצאה בייצוג 8 ביטים קיבלנו 0F ב hex כדרוש.

### פעולה shift left :

ביצענו את הפעולה shift left שהערך שלה הוא 16 .

מטרת הפעולה היא להזיז את Y ב-X פעמים שמאלה.

טענו ל X את הערך FF02 ולY את FF04 . \*הערה : מה שרלוונטי זה 8 ביטים ה-LSB כלומר 2 הספרות הימניות.

לאחר מכן כדי להשתמש בshift left לחצנו על הswitches לקבל את הערך 16 ולחצנו על 3.key.

ואכן ניתן לראות שבתוצאה בייצוג 8 ביטים קיבלנו 05 ב hex כדרוש.

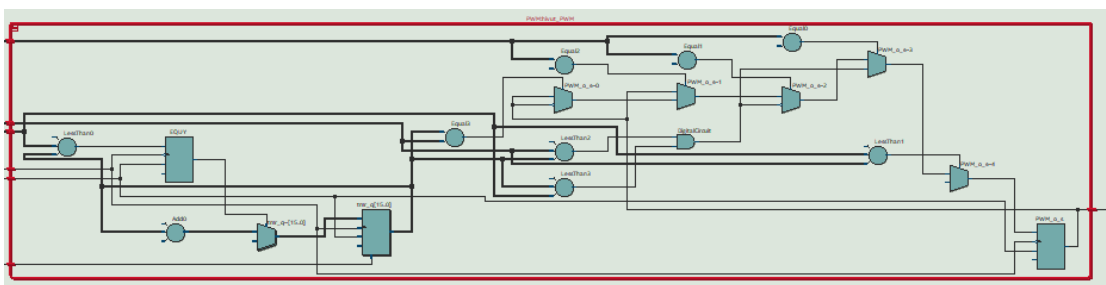
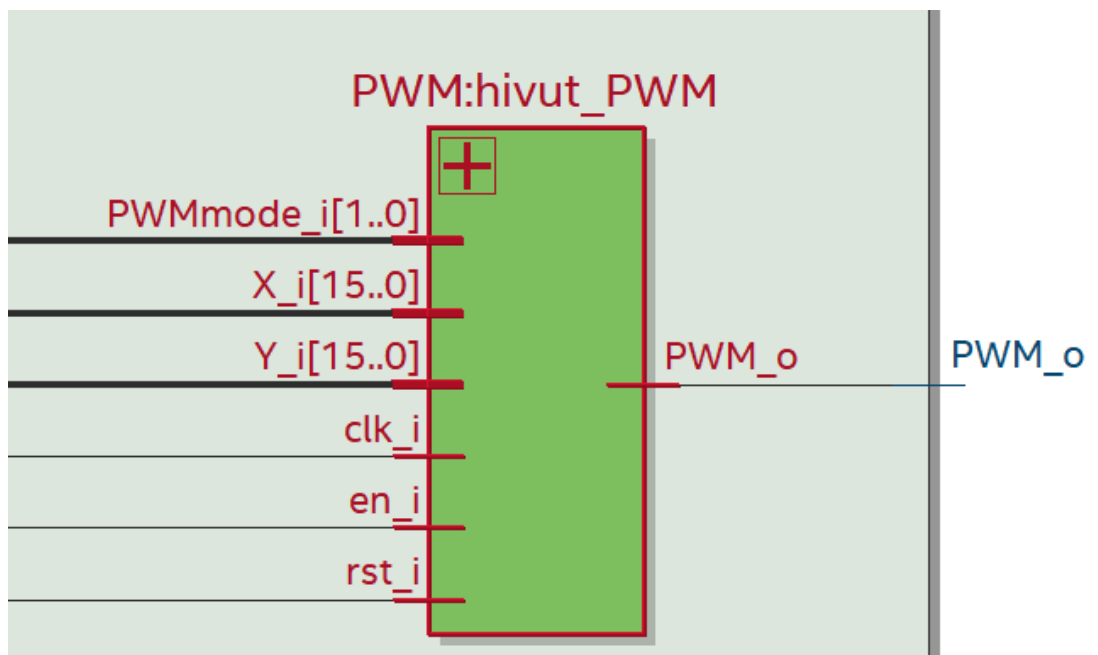
log: Trig @ 2025/06/03 17:42:26 (00)		click to insert time bar					
Node	1	2	3	4	5	6	
Name	0	1	2	3	4	5	6
...ortmap top[X][15.0]				FF02h			
...ortmap top[Y][15.0]		FF04h			FF04h		
ALUFN[4.0]			08h			10h	
...p top[ALUout_o][7.0]	0Fh		05h		06h		10h

## שרטוטי RTL

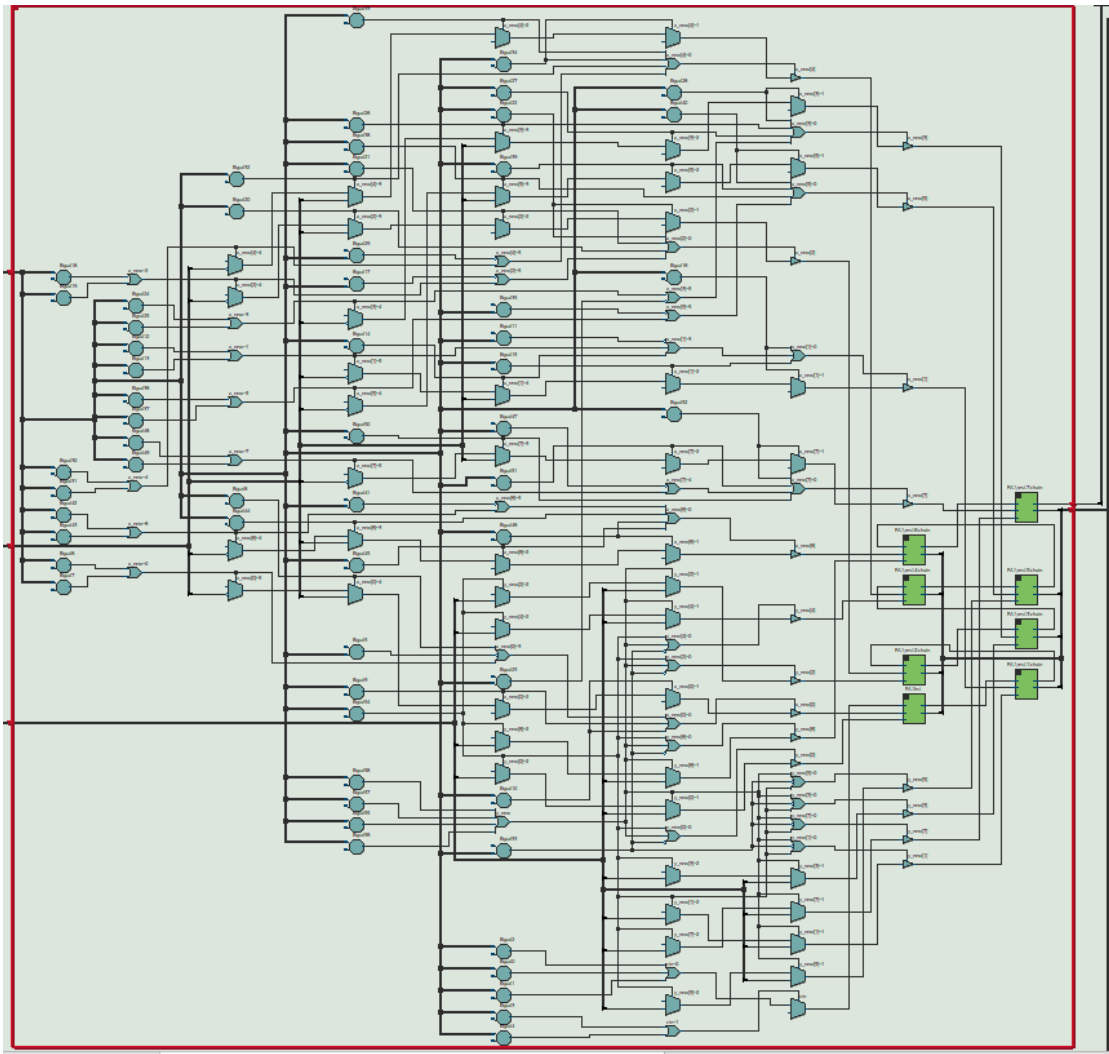
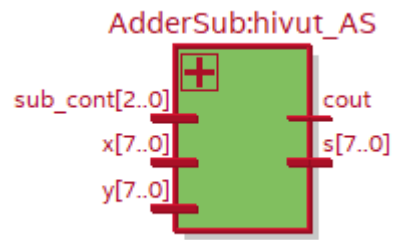
ב- Quartus צפינו במימוש החומרתי של הקוד שכתבנו בVHDL על ידי  
RTL VIEWER.

לכל מודול נראה את entity שלו ושרטוט ה- RTL :

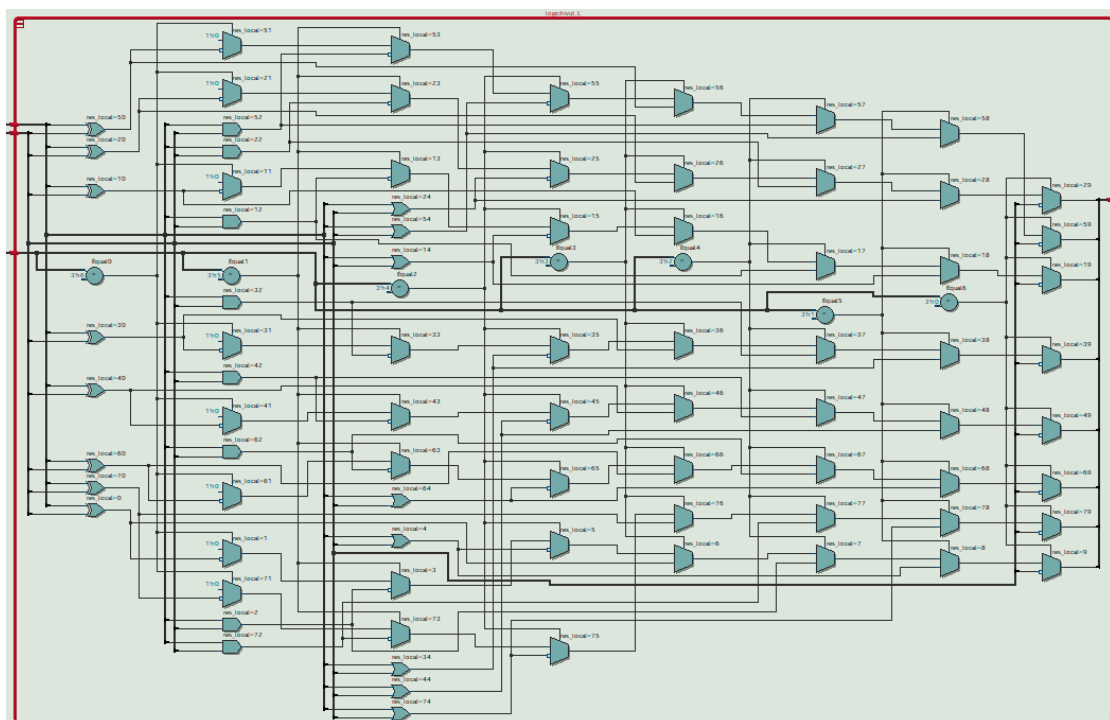
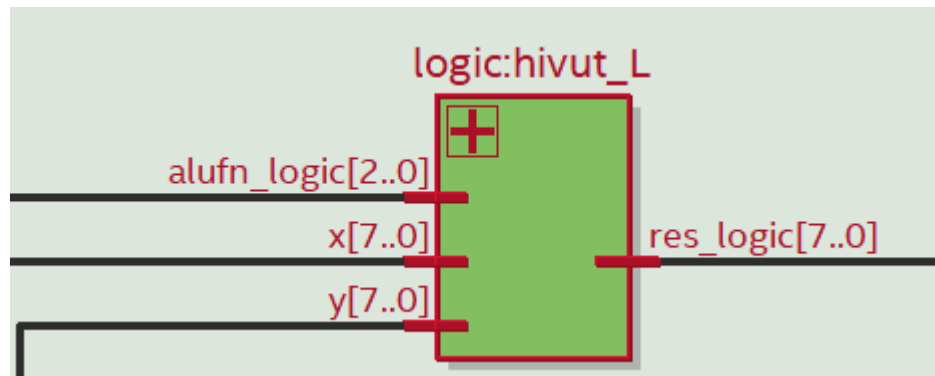
### PWM:



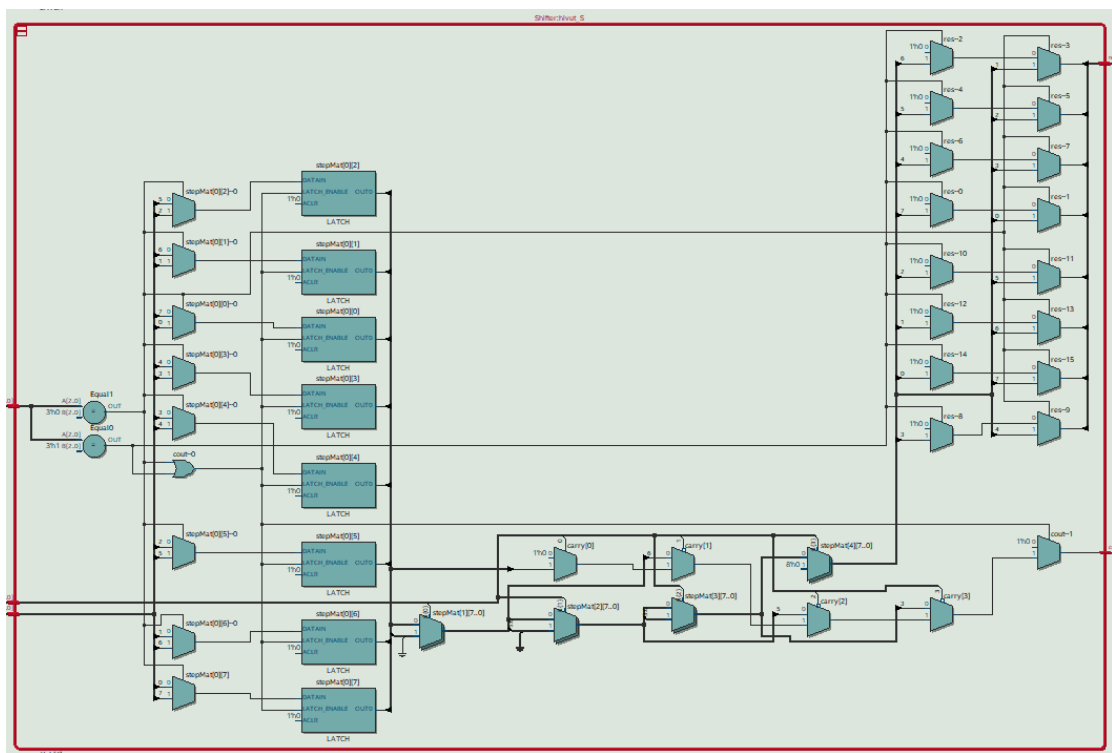
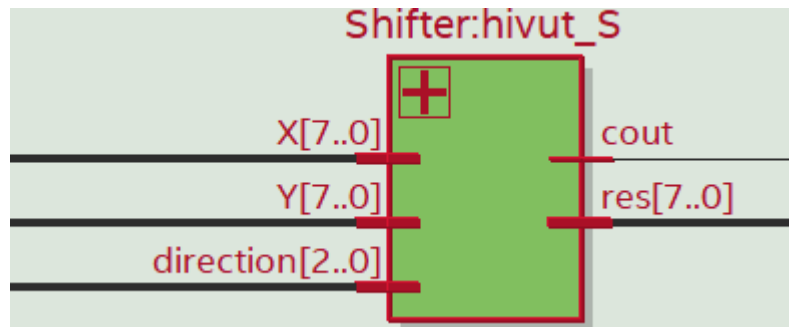
# ADDERSUB



# LOGIC



# SHIFTER



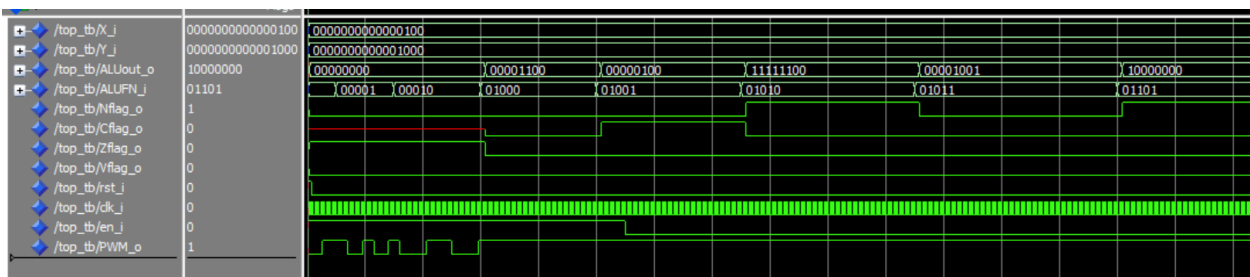
## סימולציית modelsim:

יצרנו קבצי testbench , כדי לבדוק תחילה את המערכת שיצרנו לפני המעבר לחומרה.

עשינו testbench לכל תת מודול כדי לבחון אותו.

טסט כללי של ה- SYSTEM TOP ENTITY:

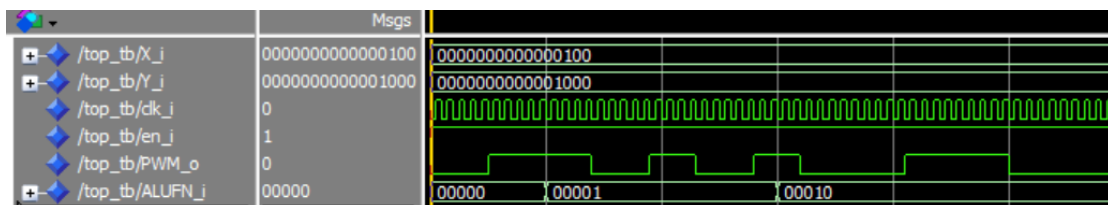
ניתן לראות במקביל תוצאות של גלים של ה- pwm ובנוסף תוצאות של ALU עם דגלים שיוצאים בהתאמה. כלומר שילוב של 2 המערכות .sdc | cdc



סימולציה של pwm:

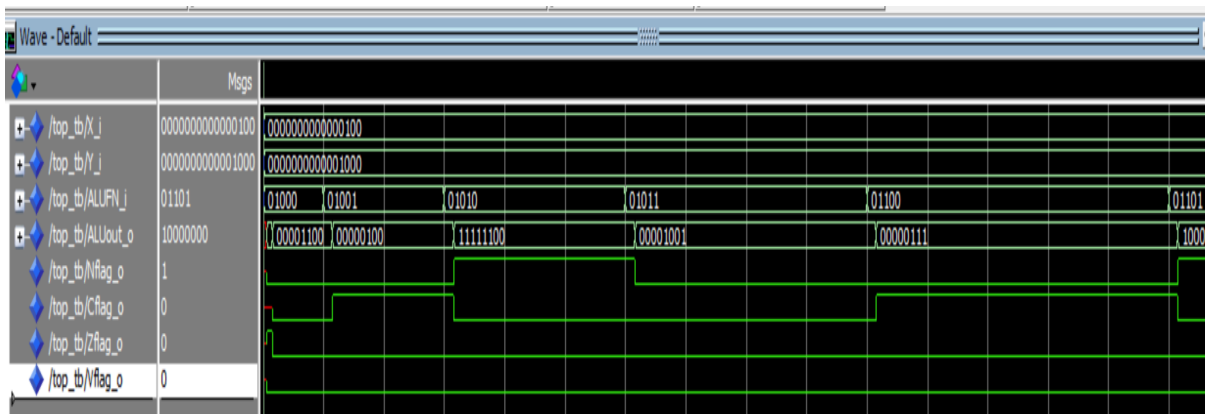
ניתן לראות את הגל המתאים בהתאם לALUFN, לפי הסדר:

mode 0, mode 1, toggle (mode 2)



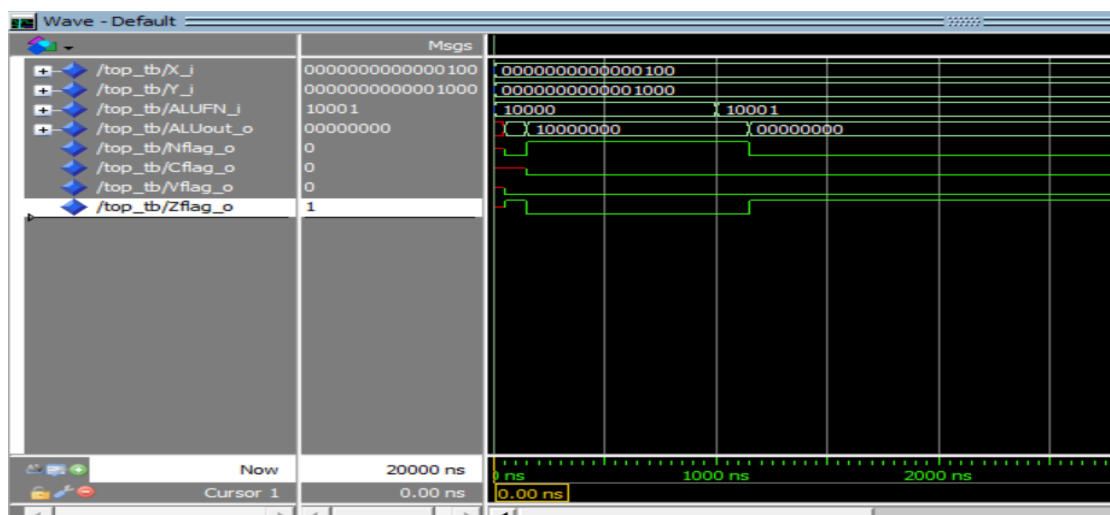
## סימולציה של addersub:

שמנו כקלט את המספרים 4 ו 8 , ועברנו על כל הפעולות האריתמטיות.



סימולציית shifter :

שמנו כקלט את המספרים 4 ו 8, ועברנו על כל הפעולות ההוצזה.



## סימולציית logic

שמנו כקלט את המספרים 4 ו 8 , ועברנו על כל הפעולות הלוגיות

