

Preparation Lab4

Advanced CPU architecture and hardware accelerators lab

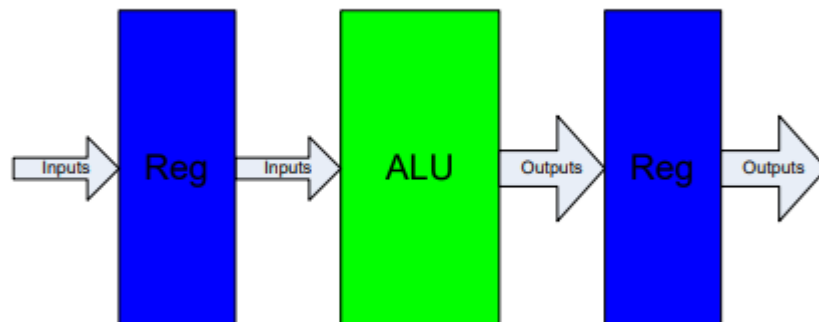
Prepared by: Yarin Oziel, Itay Kandil

הקדמה

במעבדה זו למדנו לבצע סינתזה עבור מודלים שפיתחנו, בנוסף למציאת critical path , frequency limitation וביצוע אופטימיזציה חומרית בהתאם. כדי לבצע סינתזה השתמשנו בתוכנה quartus של אינטל על גבי FPGA V Cyclone של כרטיס standard 10DE.



Performance test case - ALU

בתור הקדמה נדרשנו לבצע בדיקת ביצועים עבור alu : בשונה ממעבדה 1, ומכיוון שבמעבדה זו המודל נצרר על גבי רכיב ממשי (כרטיס fpga שלנו) יש להתחשב במגבלות פיזיקליות של המודל כמו תדר השעון המקסימלי בו הוא מסוגל לפעול באופן תקין . כדי למצוא אותו תחמנו את המודל בין שני רגיסטרים באופן הבא



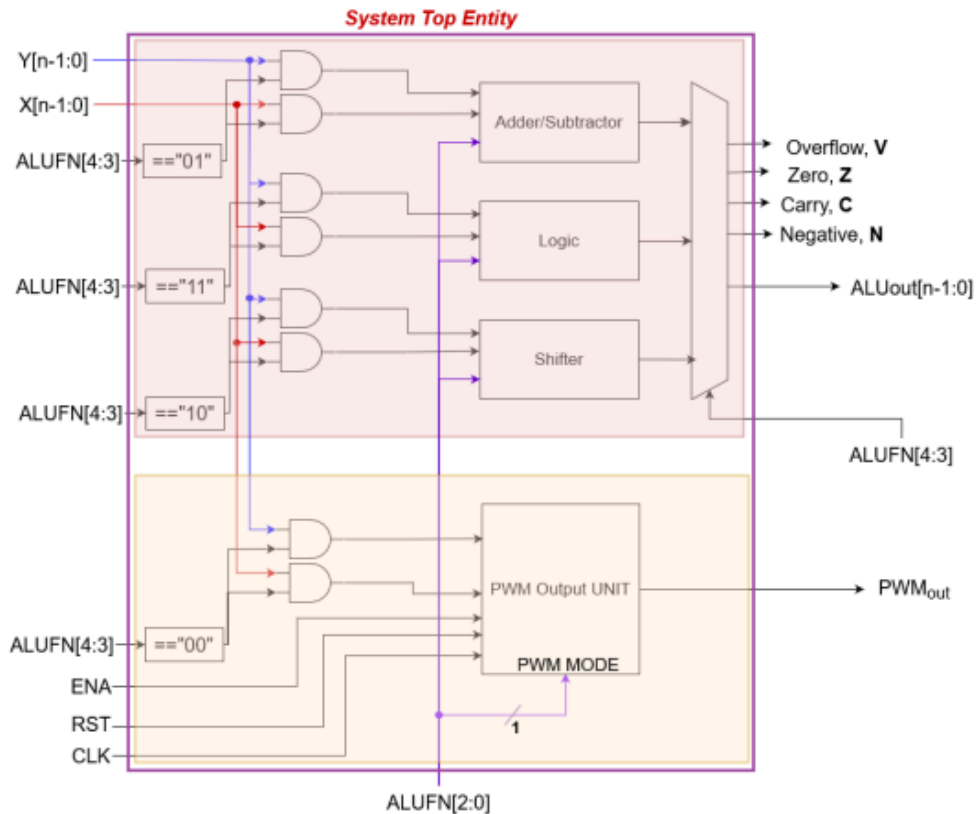
- הקוד בו השתמשנו למידול נמצא בקבצי ההגשה

להלן התוצאות :

Slow 1100mV 85C Model Fmax Summary				Slow 1100mV 0C Model Fmax Summary			
 <<Filter>>				 <<Filter>>			
	Fmax	Restricted Fmax			Fmax	Restricted Fmax	
1	155.28 MHz	155.28 MHz	clk	1	155.45 MHz	155.45 MHz	clk

System architecture

בחלק זה נציג את המערכת, את אותות הכניסה והמוצא, את תתי המודולים שלה, ואת אופן פעולתם



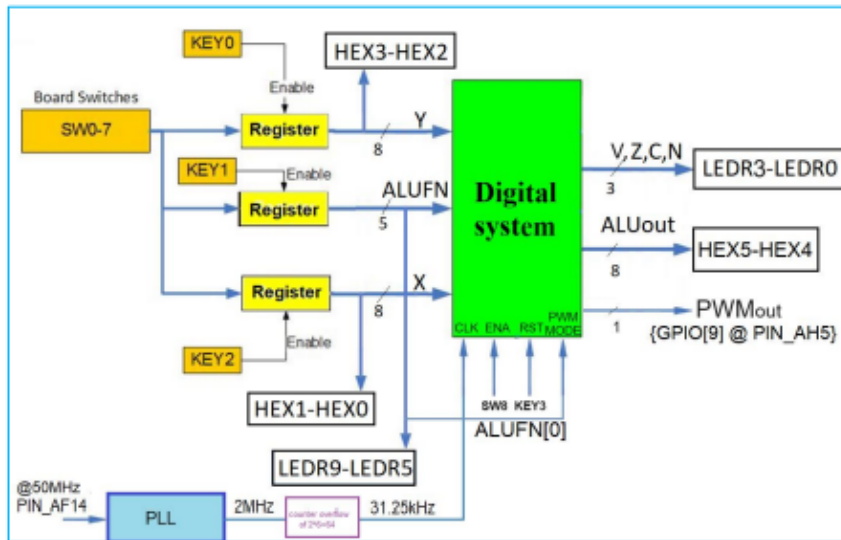
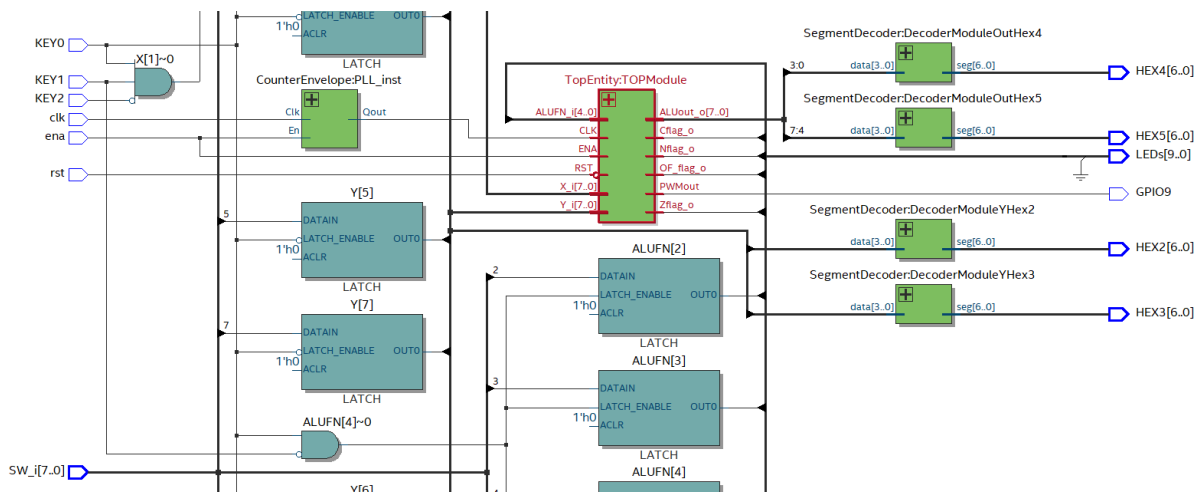
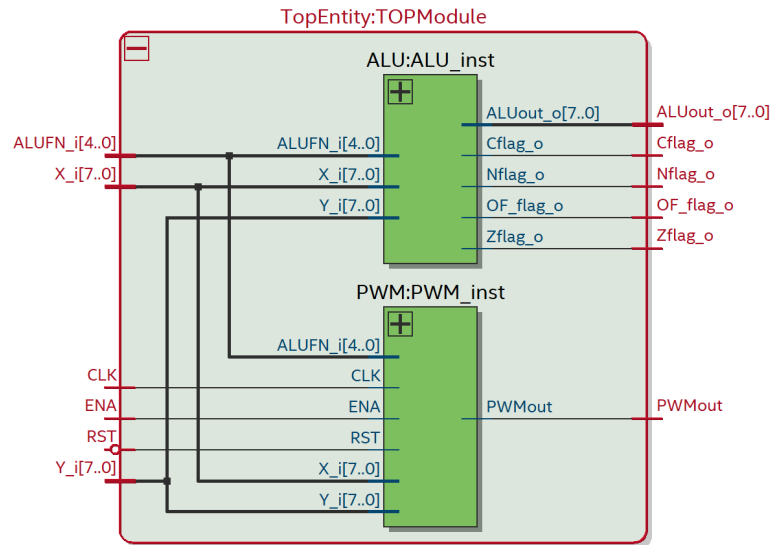


Figure 3: Digital system with I/O interface

המערכת נעטפת על ידי שכבת io_interface, המקבלת את אותות הכניסה מן המשתמש מעבירה אותם אל שכבת top - שבתורה מחווטת אותם אל תתי המודולים. בסיום החישובים הלוגיים שכבת io_interface מקבלת את אותות המוצא ומציגה אותם על רכיבי החומרה השונים של כרטיס fpga.





אותות הכניסה :

X , y, alu func :

המשתמש ימתג את SW0-SW7 כדי שיציגו את המספר הרצוי בתצוגה בינארית ולאחר מכן ישתמש בkey0-key2 כדי ללכוד את את הקלט ברגיסטר המתאים (x,y,alu_func) ערכי הרגיסטרים המכילים את x,y,alu_func יוצגו על גבי led5-9, hex2-3, hex0-1 בהתאמה. ערכי דגלי המוצא v,z,c,n ומוצא המערכת עצמו יוצגו על גבי led0-3 hex 4-5 בהתאמה

Clock :

כרטיס ה FPGA מכיל גביש המסוגל לייצר שעון בתדר 50 megahertz . אנחנו נדרשנו לייצר שעון בתדר 31.25 kilohertz לכן ביצענו את הפעולות הבאות : את מוצא השעון חיברנו את מודול pll (מיוצר template של סביבת העבודה) המסוגל לחלק את תדר השעון פי 25- כלומר 2 megahertz . את מוצא ה pll חיברנו אל מונה 6 ביט הפולט "1" לוגי כאשר הוא ב overflow כלומר חלוקת תדר פי 64 . מוצא המונה יהווה שעון בעל תדר 31.25 kilohertz

Rst, ena :

כפתור key 3 מהווה ריסט אסינכרוני למערכת
סוויטצ 8 מהווה מאפשר ליחידת ה pwm

Alu module :

היחידה האריתמטית לוגית מבוססת על המודול שפיתחנו עבור מעבדה אחת. היחידה מקבלת כקלט של אופרנדים - x,y בנוסף לקלט פונקציונלי - המהווה את הקידוד עבור הפעולה הרצויה .
היחידה פולטת את alu_out - תוצאת החישוב , בנוסף ל 4 דגלים : zero , carry flag , negative flag , overflow flag
היחידה כוללת 3 תתי מודולים :

יחידת חיבור / חיסור מבוססת ripple adder בשיטת המשלים ל 2
יחידה לוגית המסוגלת לבצע פעולות לוגיות בסיסיות כגון or , and , not
יחידת שיפטר מבוססת ארכיטקטורת barrel shifter

בחלק הבא נציג טסטים בסיסיים עבור כל אחד מתתי המודולים בנוסף ל rtl view וה critical path :

מודול adder_sub

סימולציה

נבדוק את תקינות המודול לפי כמה מקרי קצה, הכנסנו בדיקה בכל מקרה כך שבמידה ולא תהיה תואמת לפונקציה הבולאנית המתארת את המעגל נקבל שגיאה בסימולציה.
בדקנו את מקרי הקצה: חיבור, חיבור עם carry, חיסור ו negative.

Waveform 80ns (20ns per test, Hexadecimal Radix):

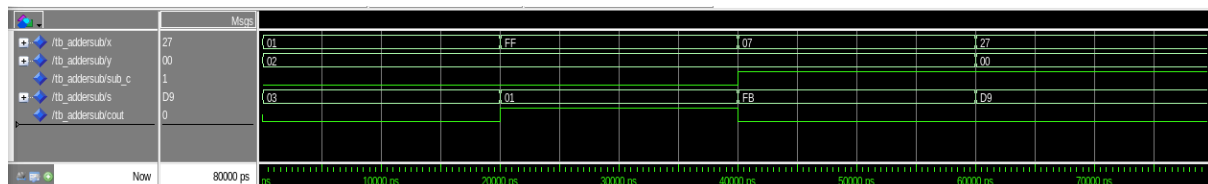


Table:

ps- delta		/tb_addersub/x	/tb_addersub/y	/tb_addersub/sub_c	/tb_addersub/s	/tb_addersub/cout
0	+3	00000001	00000010	0	00000011	0
20000	+10	11111111	00000010	0	00000001	1
40000	+9	00000111	00000010	1	11111011	0
60000	+3	00100111	00000000	1	11011001	0

קיבלנו תוצאות תקינות וללא שגיאות.

מודול Logic

מודול זה מתוכנן באופן דומה ל־MUX כך שבהינתן סיגנל ALUFN בביטים 0,1,2 הוא בורר מבין 7 מצבים ומבצע פעולות על 1 או 2 אופרנדים.

המצבים התואמים ל ALUFN[2:0] עבור כניסות X ו- Y:

- Not(Y): 000
- Y OR X: 001
- Y AND X: 010
- Y XOR X: 011
- Y NOR X: 100
- Y NAND X: 101
- Y XNOR X: 111

סימולציה

נבדוק את תקינות המודול לפי כמה מקרי קצה, הכנסנו בדיקה בכל מקרה כך שבמידה ולא תהיה תואמת לפונקציה הבולאנית המתארת את המעגל נקבל שגיאה בסימולציה.
בדקנו את מקרי הקצה: .NOT, OR, AND,XOR,NOR,NAND,XNOR.

Waveform 160ns (20ns per test, Hexadecimal Radix):

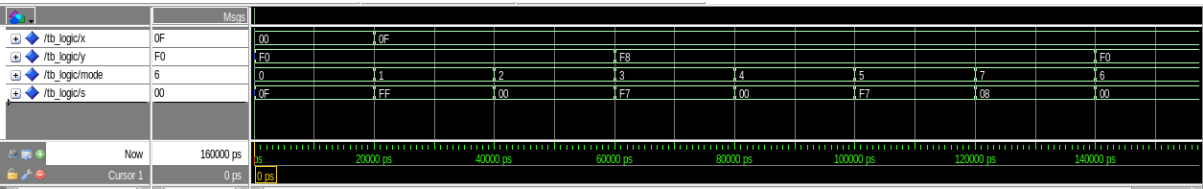


Table:

ps	delta	/tb_logic/x	/tb_logic/y	/tb_logic/s	/tb_logic/mode
0	+2	00000000	11110000	000	00001111
20000	+2	00001111	11110000	001	11111111
40000	+2	00001111	11110000	010	00000000
60000	+2	00001111	11111000	011	11110111
80000	+2	00001111	11111000	100	00000000
100000	+2	00001111	11111000	101	11110111
120000	+2	00001111	11111000	111	00001000
140000	+2	00001111	11110000	110	00000000

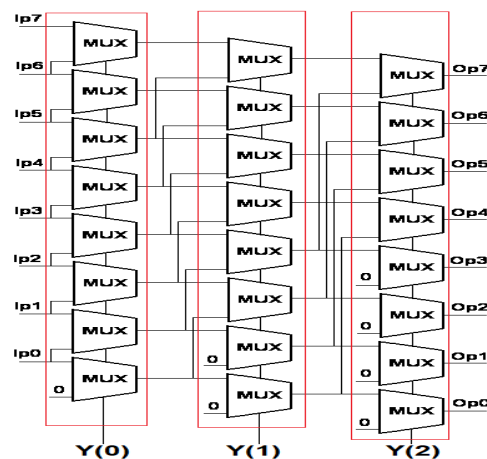
קיבלנו תוצאות תקינות וללא שגיאות.

מודול Shifter

מודול זה מבצע הזזות לכניסה Y לפי X פעמים מבוסס על n -bit barrel-shifter. סיגנל בקרה ALUFN קובע בביטים 0,1,2 את סוג ההזזה (שמאלה: 000, ימינה: 001).

מימוש המודול מבוצע ע"י מטריצה דו-מימדית $((n+1) \cdot \log(n))$. תחילה אם ההזזה נקבע שמאלה נעביר את Y כפי שהוא לשלב הראשון במטריצה, אחרת נכניס את Y הפוך (INVERTED) כך שנוכל לממש הזזה ימינה בעזרת מנגנון הזזה שמאלה. לאחר מכן נבצע הזזה לפי 3 הביטים הראשונים של X ונכניס לכל שלב במטריצה. נשמור את ה carry ז"א הביט האחרון שנדחף מ Y ולבסוף נעביר את התוצאה למוצא המודול (במידה וההזזה ימינה נהפוך את הפלט כדי לקבל את התוצאה הרצויה).

המודול מחקה את הארכיטקטורה הבאה:



סימולציה

נבדוק את תקינות המודול לפי כמה מקרי קצה, הכנסנו בדיקה בכל מקרה כך שבמידה ולא תהיה תואמת לפונקציה הבולאנית המתארת את המעגל נקבל שגיאה בסימולציה. בדקנו את מקרי הקצה: הזזה שמאלה, הזזה שמאלה עם carry, הזזה ימינה, הזזה ימינה עם carry והזזות לא מוגדרות (הכנסת קלטים לא מוגדרים לDir).

Waveform 200ns (20ns per test, Hexadecimal Radix):

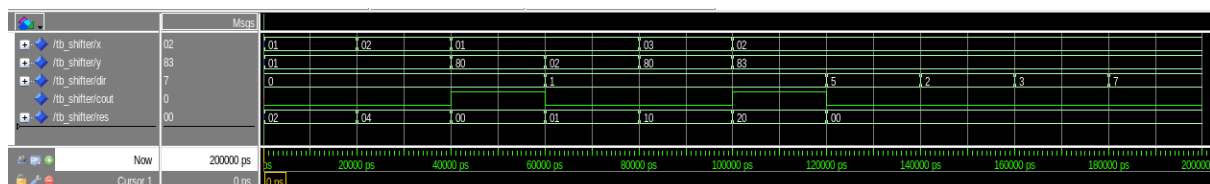


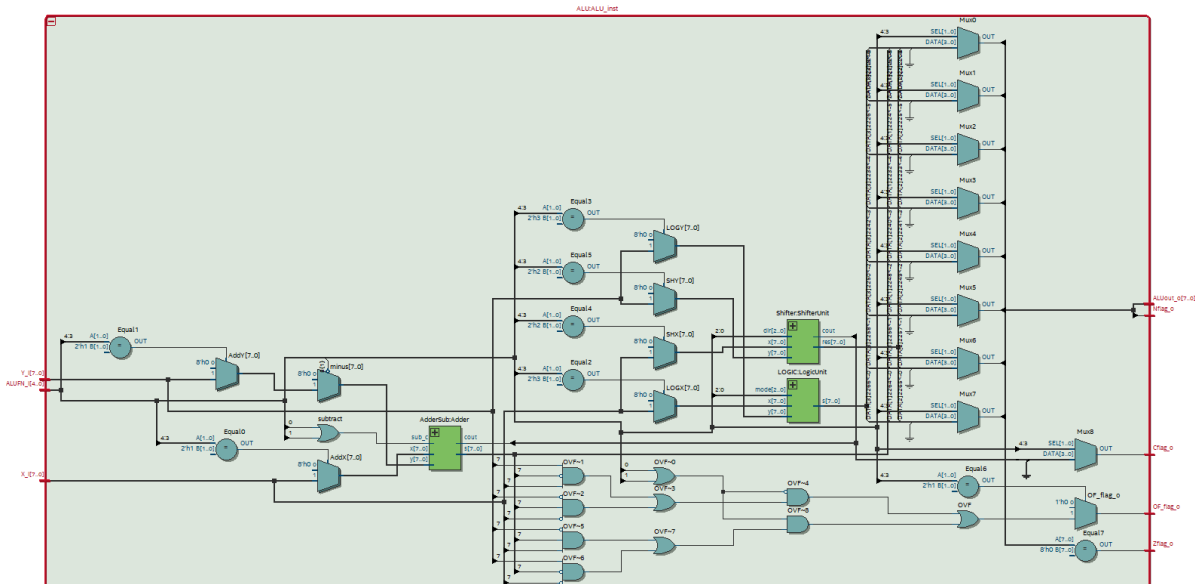
Table:

ps delta		/tb_shifter/x	/tb_shifter/res	/tb_shifter/y	/tb_shifter/dir	/tb_shifter/cout
0	+6	00000001	00000001	000	0	00000010
20000	+5	00000010	00000001	000	0	00000100
40000	+6	00000001	10000000	000	1	00000000
60000	+6	00000001	00000010	001	0	00000001
80000	+6	00000011	10000000	001	0	00010000
100000	+6	00000010	10000011	001	1	00100000
120000	+2	00000010	10000011	101	0	00000000
140000	+1	00000010	10000011	010	0	00000000
160000	+1	00000010	10000011	011	0	00000000
180000	+1	00000010	10000011	111	0	00000000

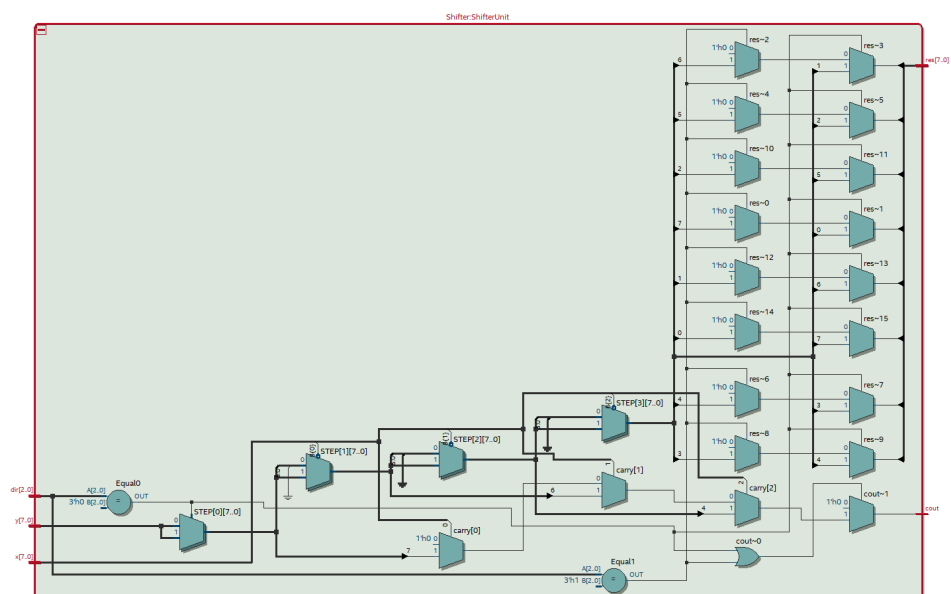
קיבלנו תוצאות תקינות וללא שגיאות.

Alu rtl view

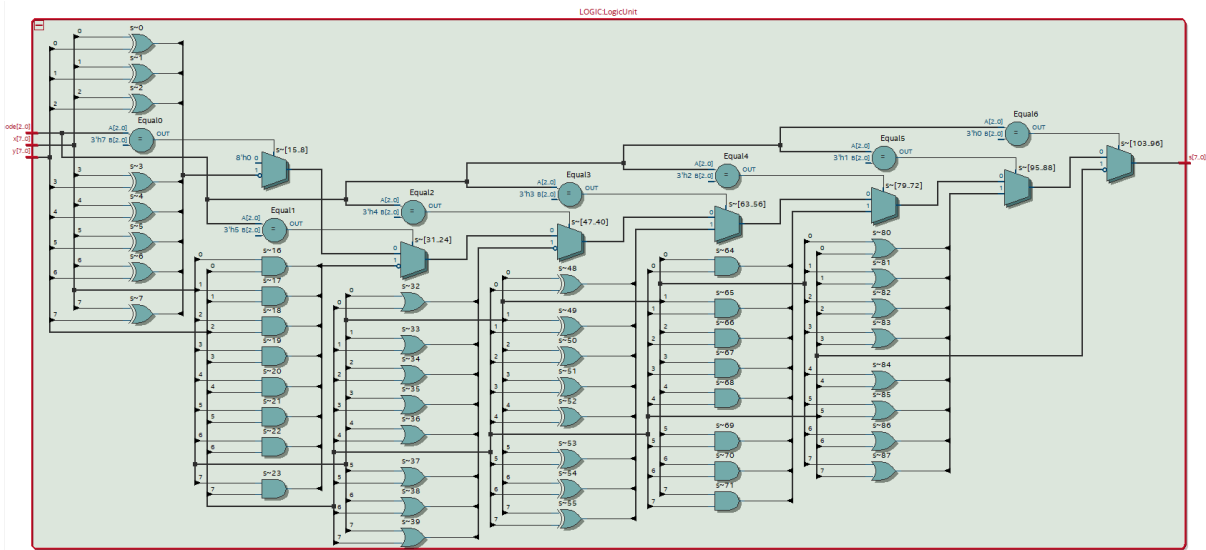
Entire module:



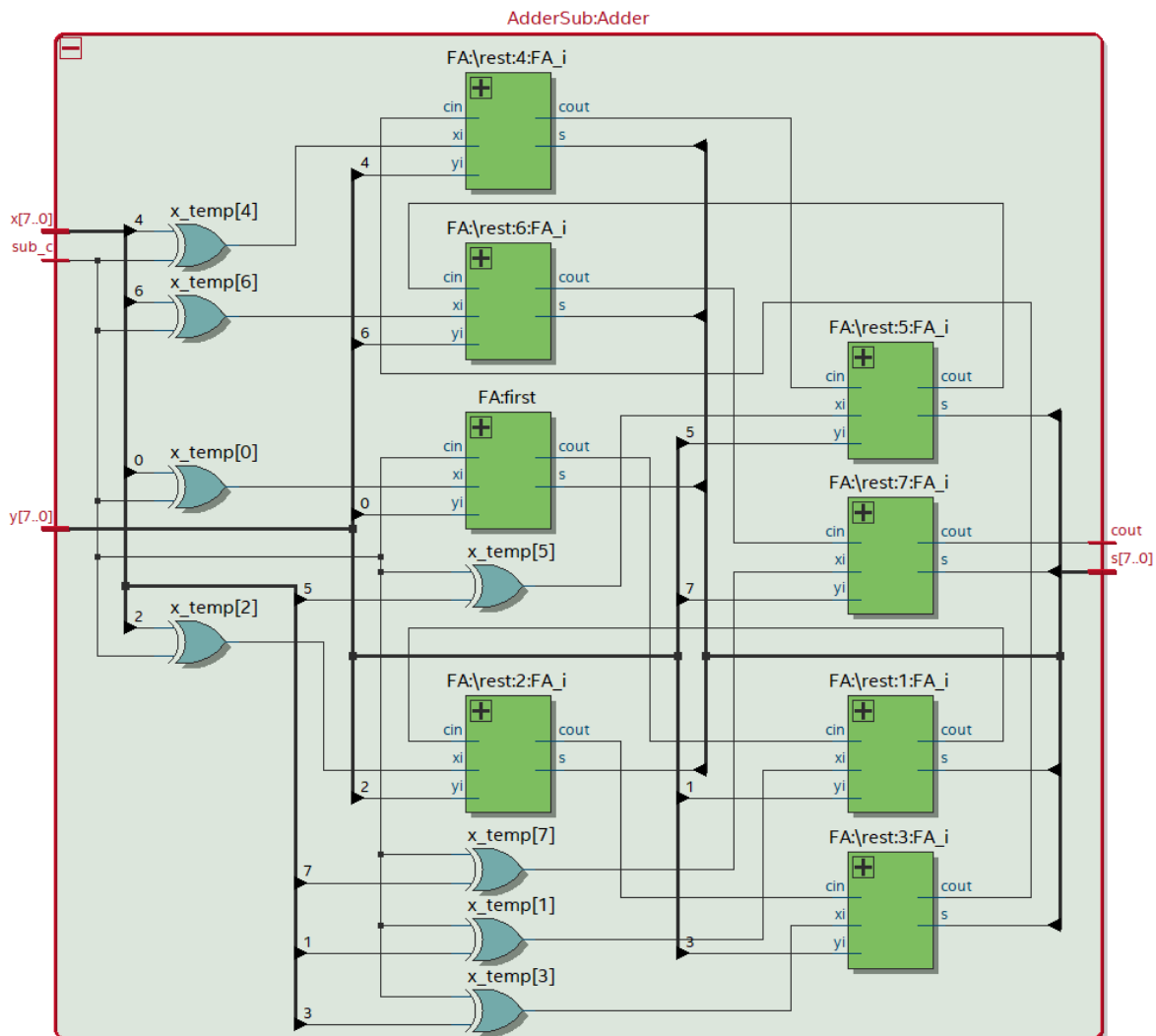
Shifter submodule:



Logic submodule:




Adder submodule:

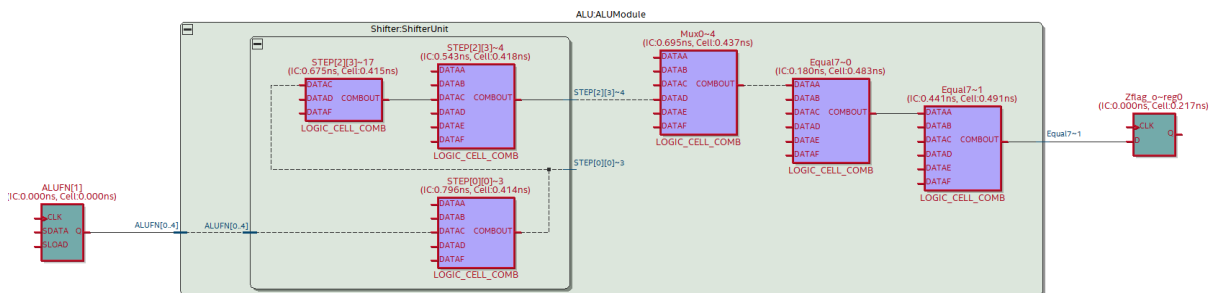
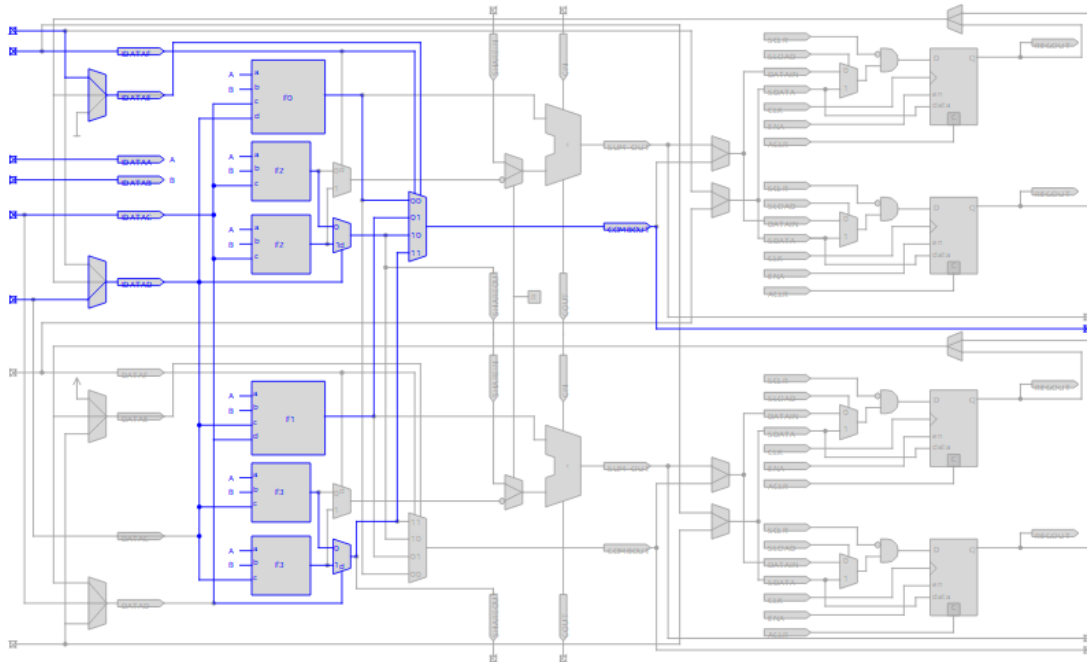


Logic usage

בחלק זה נציג את מפרט הרכיבים החומרתיים שהשתמשנו בהם לצורך
הסינטזה:

Analysis & Synthesis Resource Usage Summary		
 <<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	74
2		
3	▼ Combinational ALUT usage for logic	105
1	-- 7 input functions	5
2	-- 6 input functions	35
3	-- 5 input functions	36
4	-- 4 input functions	14
5	-- <=3 input functions	15
4		
5	Dedicated logic registers	33
6		
7	I/O pins	34
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	ALUFN[1]
12	Maximum fan-out	48
13	Total fan-out	634
14	Average fan-out	3.08

critical path view



מכיוון שיחידה מסוגלת לבצע מגוון רחב של חישובים אריטמטיים לוגיים נצפה לראות כמות גדולה של רכיבי חומרה. logic usagen מאמת טענה זו. בנוסף, מכיוון שבפעולת השיפט ישנם הרבה

רכיבי mux המחברים בצורה טורית, נצפה שהמסלול הארוך ביותר יעבור דרכה critical path view מאמת טענה זו.

Pwm unit

המודול הזה מקבל 2 אופרנדים x,y ו pwm mode המודול פולט אות pwm בצורה הבאה :

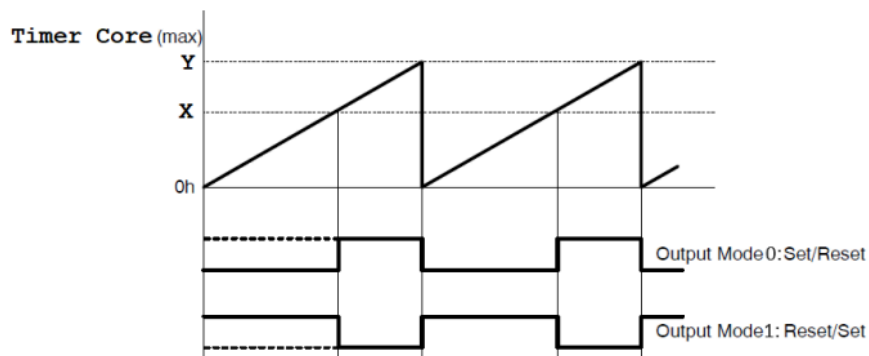


Figure 5: PWM output modes

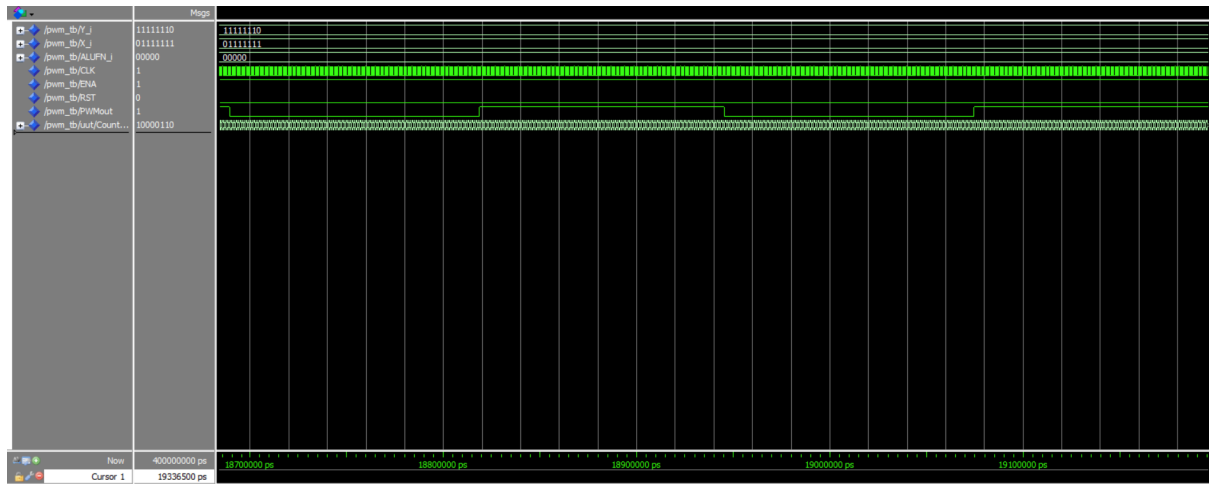
לדוגמא עבור mode 0 - כאשר עוברים "x" מחזורי שעון מוצא המערכת משתנה ל "1" לוגי וכאשר מגיע ל "y" מחזורי שעון המערכת משתנה ל "0" לוגי והספרה תחל מחדש אות המוצא מחווט אל GPIO 9 וניתן להציגו על המחשב.

סימלוציה

נבדוק את המקרה הבא : נזין את המערכת עם הערכים הבאים

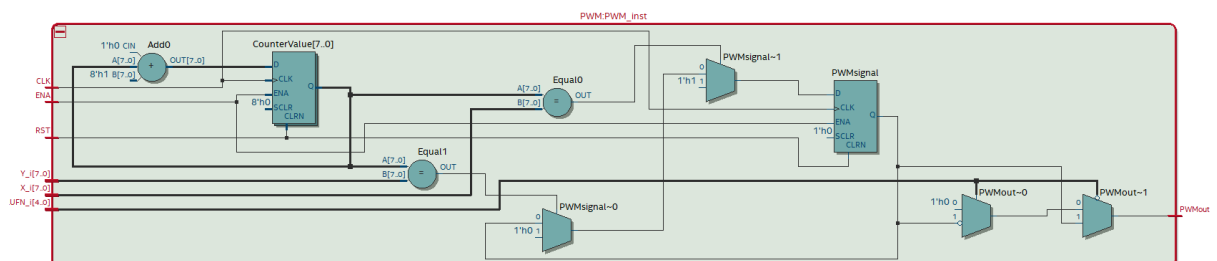
```
x="01111111"
y="11111110"
alufn="ooooo"
```

כלומר במקרה הבא אנו מצפים לקבל "1" לוגי כאשר ה counter נמצא בין x ל y ו duty cycle 50%




התוצאות תקינות כפי שציפינו .

Pwm rtl view



Logic usage

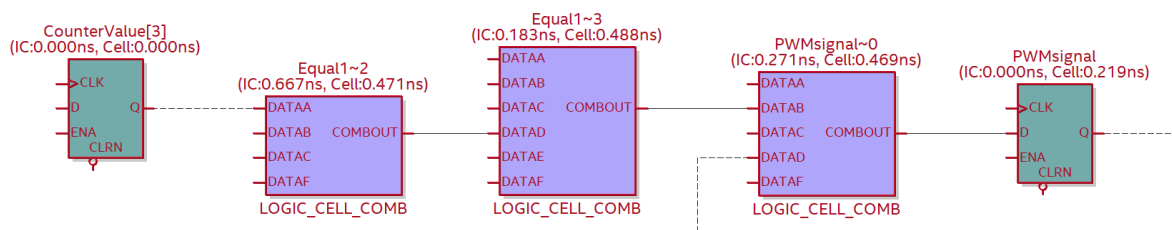
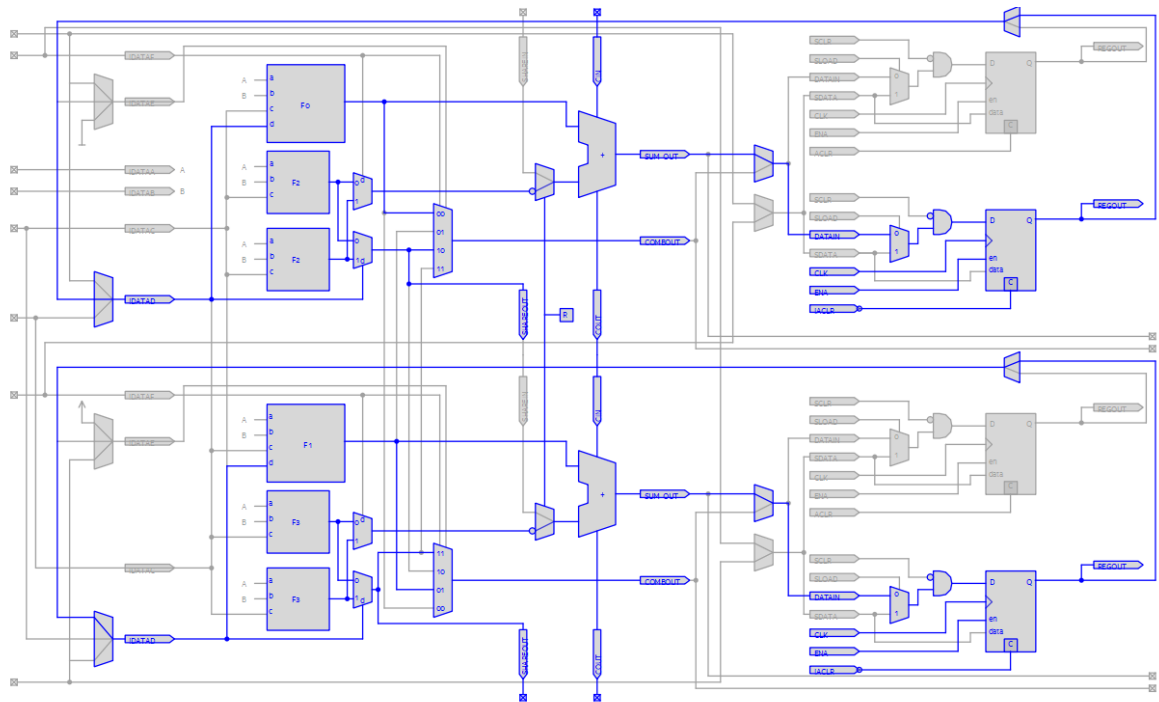
Analysis & Synthesis Resource Usage Summary		
 <<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	10
2		
3	▼ Combinational ALUT usage for logic	18
1	-- 7 input functions	0
2	-- 6 input functions	2
3	-- 5 input functions	1
4	-- 4 input functions	6
5	-- <=3 input functions	9
4		
5	Dedicated logic registers	9
6		
7	I/O pins	25
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	CLK~input
12	Maximum fan-out	9
13	Total fan-out	120
14	Average fan-out	1.56

Fmax

	Fmax	Restricted Fmax	
1	332.45 MHz	332.45 MHz	CLK

	Fmax	Restricted Fmax	
1	329.6 MHz	329.6 MHz	CLK

Critical path path



נשים לב שיחידה זו משתמשת בפחות חומרה מהיחידה האריטמטית לוגית, שכן יחידה זו מבצעת פעולה אחת. בנוסף, כפי שציפינו הcritical path הוא למעשה כל היחידה - אותות הכניסה אינן משנות הלכה למעשה את אופן הפעולה של המודל ולכן "זרימת המידע" היא תמיד לאורך כל היחידה ומכילה את כל תתי הבלוקים שלה.

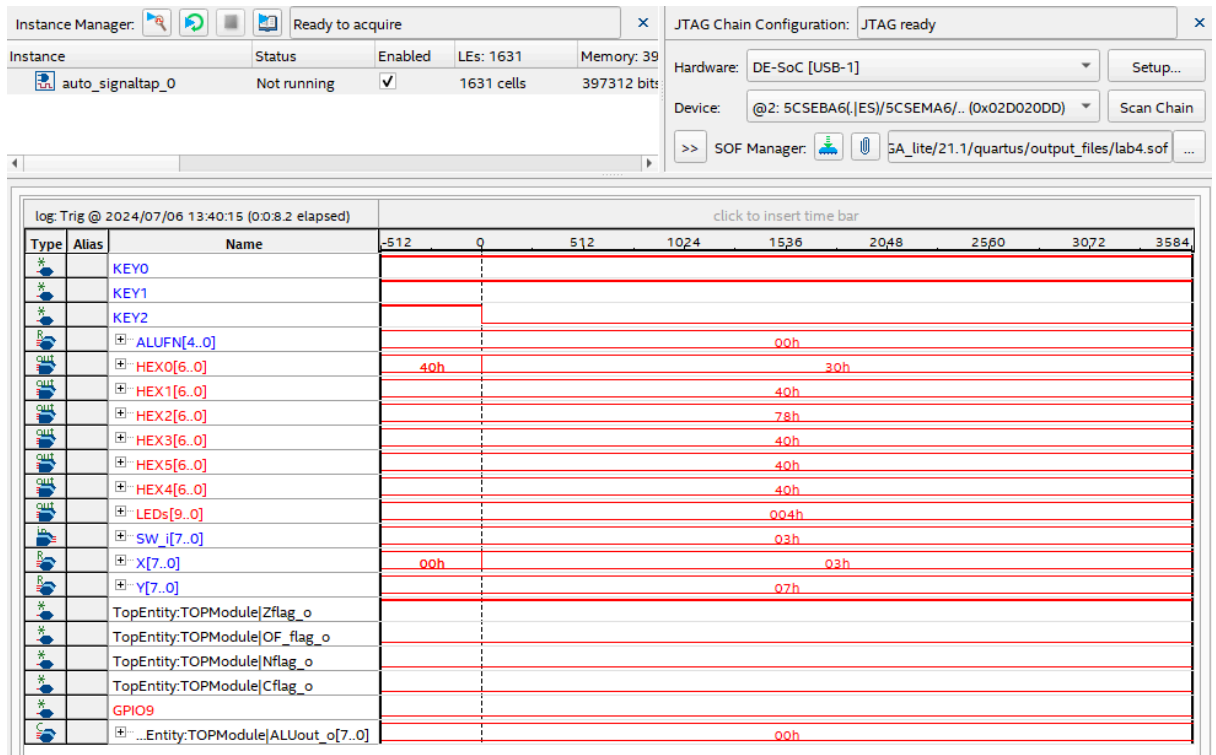
Signal tap

כדי לבצע וורפיקציה חומרית ב "זמן אמת" נשתמש בפונקציית ה-tap signal של ה-quartus. פונקצייה זו מאפשרת להגדיר רשמית רגישויות, כאשר אחד מן הסיגנלים שהוגדרו משתנה - המערכת תלכוד את ערכי הסיגנלים השונים ובכך נוכל לאמת את נכונות המערכת.

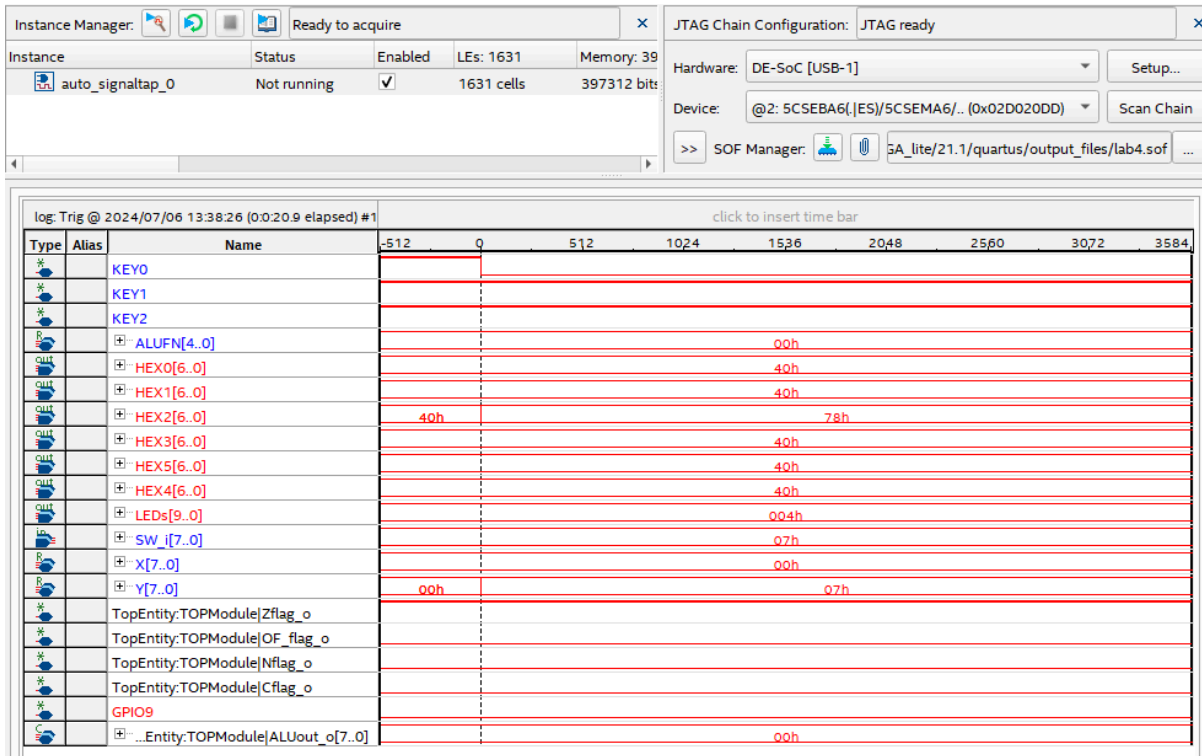
התבקשנו לבצע signal tap עבור 2 אופרציות שונות - אחת עבור פעולה אריתמטית והשנייה עבור פעולת שיפט. ברשימת הרגישויות הגדרנו את key 0, key 1, key 2 ולכן כאשר נלחץ על אחד מן הכפתורים הללו (הפונקציה הוגדרה לפעול ב falling edge עבור הכפתורים) כדי ללכוד את ערכי האופרנדים או האופרציה הדרושה ברגיסטרים - פונקציית הסיגנל טפ תצא לפעולה ונקבל פירוט של ערכי הסיגנלים השונים במערכת בעת הלחיצה .

להלן התוצאות :

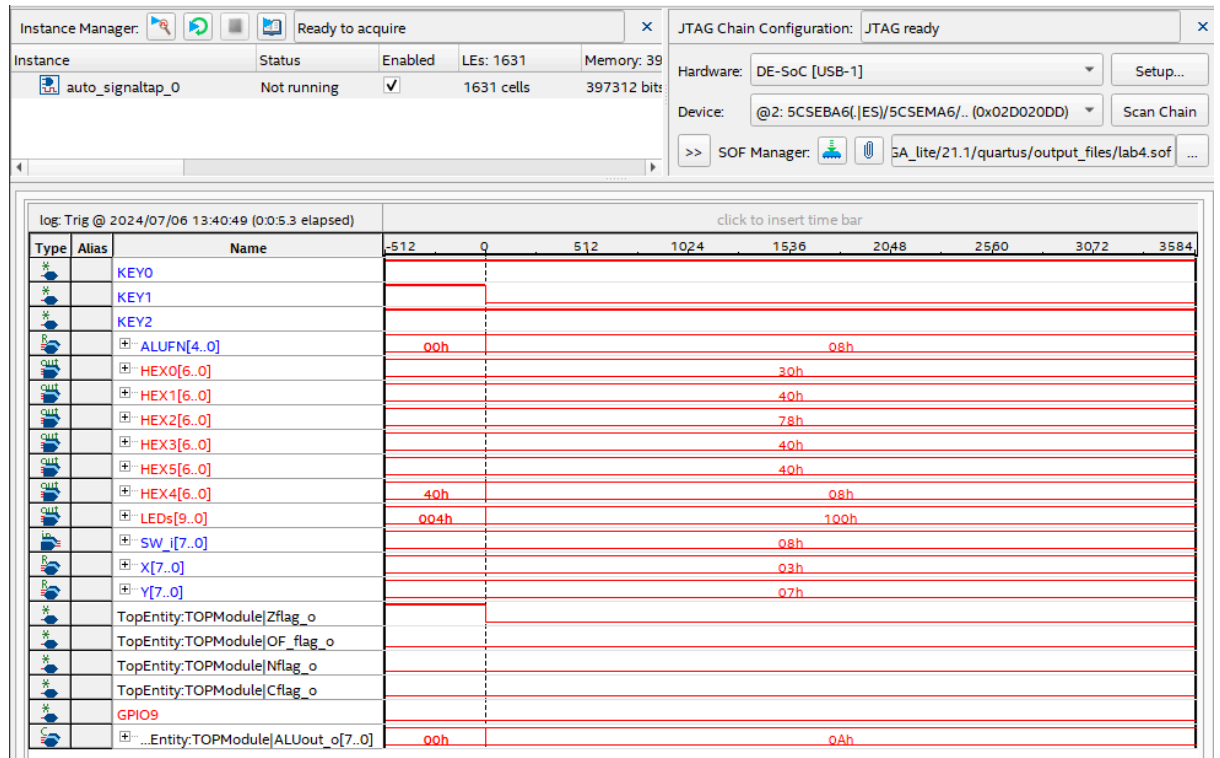
Choosing x : key 2 is being pressed therefore x changes from 0 to 3



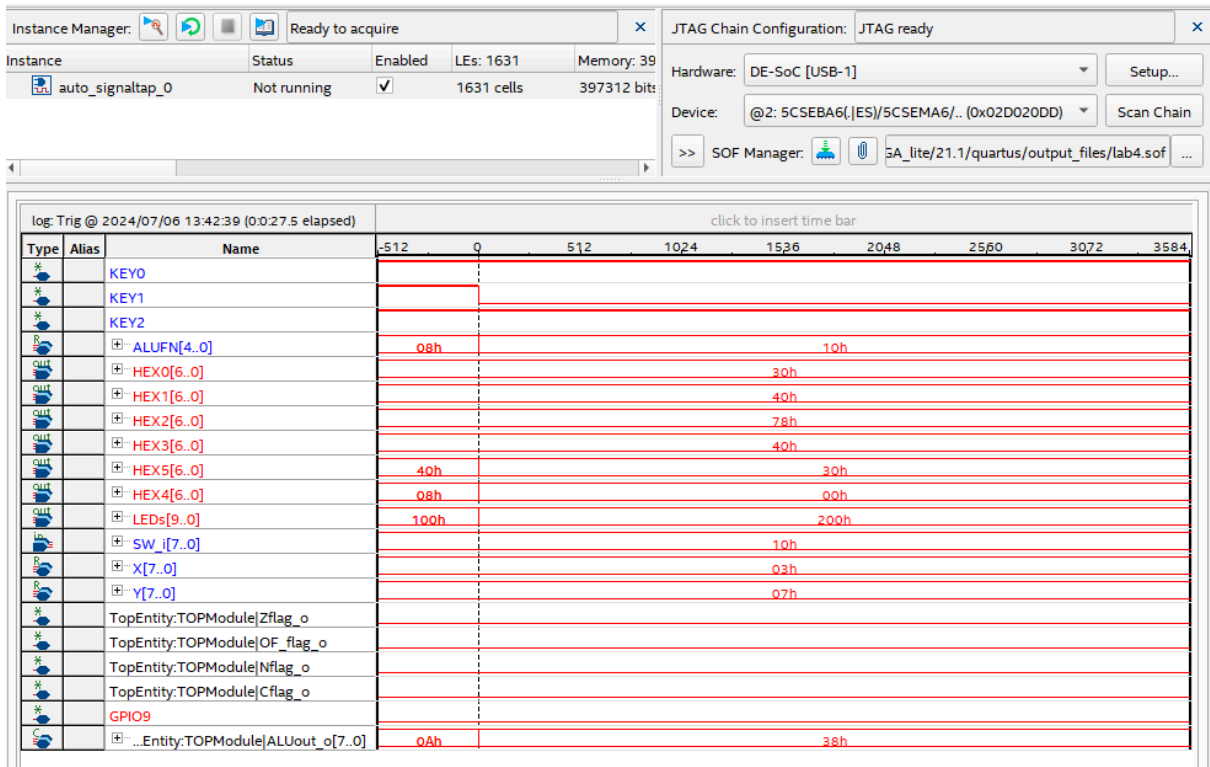
Choosing y : key 0 is being pressed therefore y changes from 0 to 7

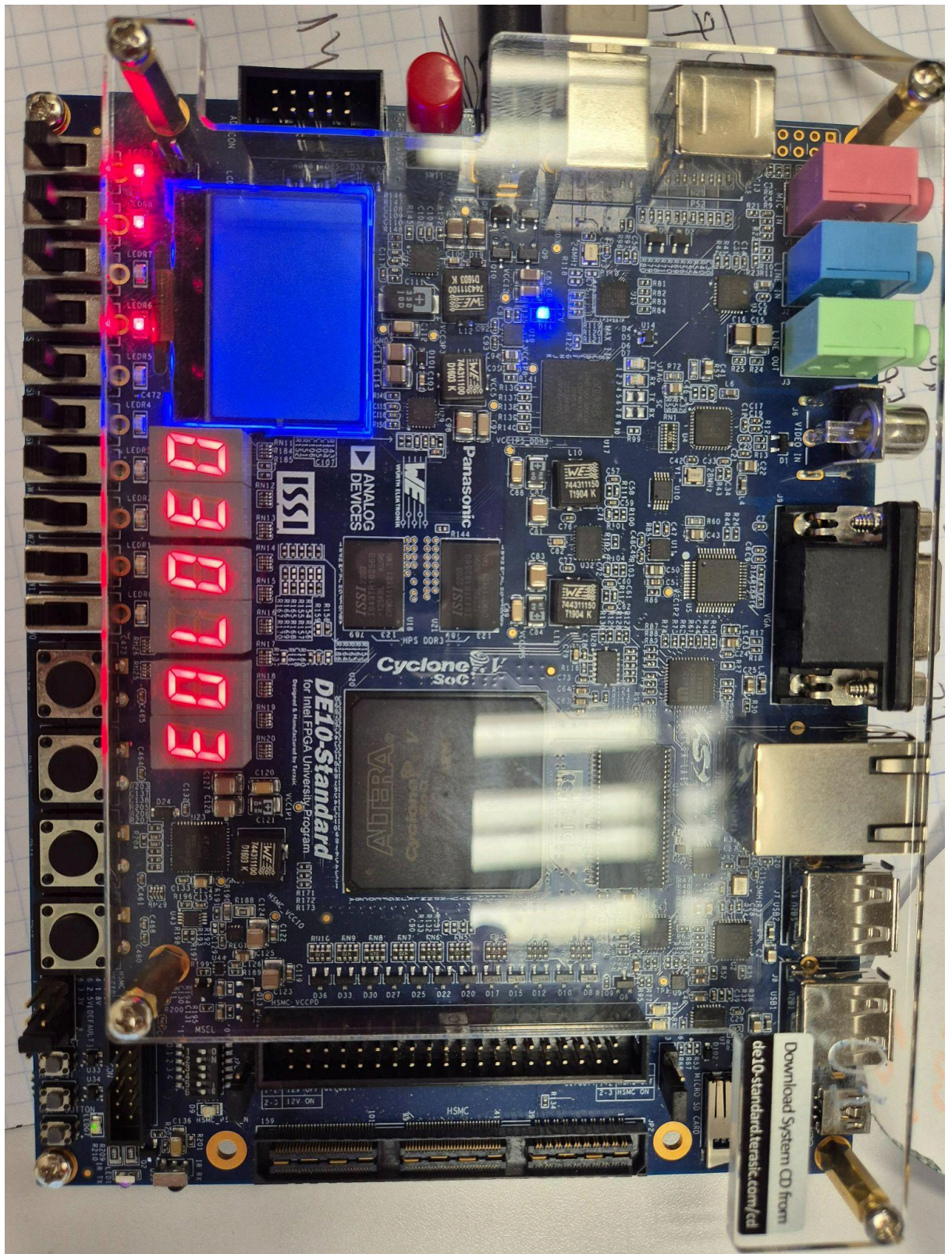


Choosing add function : key 1 is being pressed therefore alufn[4,0] is changed from 0 to 8 and hex 4 and leds change accordingly to display result

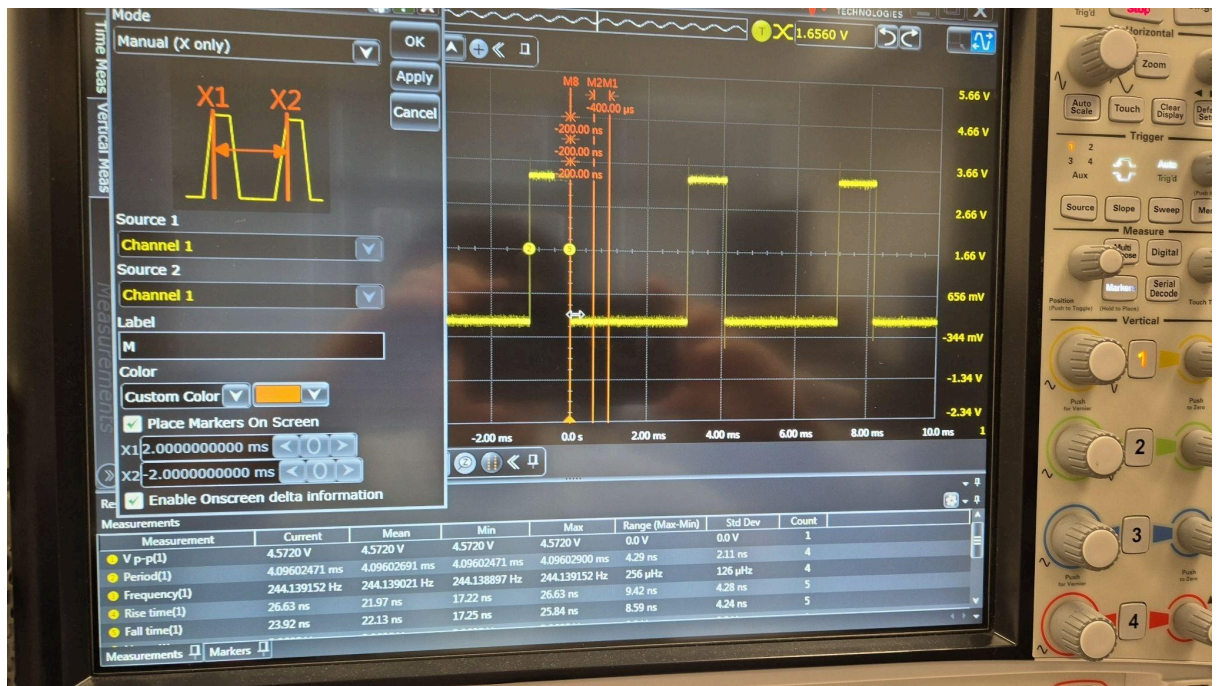


Choosing shift function : key 1 is being pressed therefore alufn[4,0] change from 8 to 10 and hex [4,5] and leds change accordingly to display result





And operation between 3 and 7



244 hz pwm pulse