

Preparation Report LAB4

ADVANCED CPU ARCHITECTURE AND HARDWARE

ACCELERATORS LAB

361.1.4693

Aram Khater 314813452

Ahseen Alazazma 324038132

מטרת המעבדה

במעבדה זו למדנו כיצד להשתמש ביכולותיה של תוכנת Quartus, ובפרט לבצע סינתזה עבור מודלים שפיתחנו במעבדה 1 עם תוספת של חלק סינכרוני שמייצר אות PWM בשלושה מצבים. את הסינתזה ביצענו על גבי רכיב FPGA מסוג Cyclone II, באמצעות כרטיס DE10-Standard.

סימולצית ModelSim

נראה סימולציה של החלק החדש שפיתחנו, נותנים ערכים קבועים ל $Y = \text{"FFFF"}$, $X = \text{"7FFF"}$ ו $ALUFN = \text{"00010"}$ כדי לראות את PWM יציב.

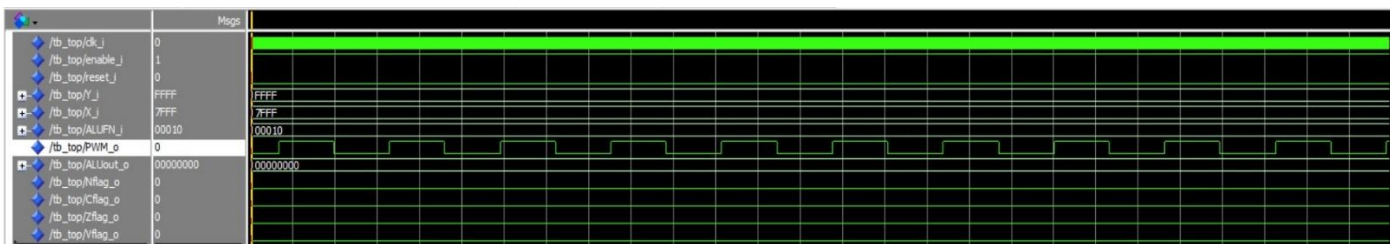


Figure1: סימולציה עבור אות PWM במצב Toggle

מציאת תדר מקסימלי

על מנת למצוא את התדר המקסימלי של המערכת, יש להוסיף רגיסטרים סינכרוניים בכניסה וביציאה של ה-ALU. לשם כך, יצרנו קובץ VHDL חדש אשר עוטף את מערכת ה-ALU-ברגיסטרים, כאשר כולם מוזנים מאותו אות שעון. את תהליך הקימפול והסינתזה ביצענו ללא השמה לפינים, כולל אות השעון, בהתאם להנחיות. לאחר השלמת הקימפול והסינתזה, תוכנת Quartus מספקת את ערך התדר המקסימלי שבו יכולה המערכת לפעול, והתוצאה שהתקבלה היא: עבור החלק הקומבינטורי קיבלנו:

Slow 1100mV 0C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	132.42 MHz	132.42 MHz	clk	

Slow 1100mV 85C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	129.48 MHz	129.48 MHz	clk	

עבור החלק הסנכרוני קיבלנו:

Slow 1100mV 85C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	310.66 MHz	310.66 MHz	clk	

Slow 1100mV 0C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	305.53 MHz	305.53 MHz	clk	

שיפור תדר השעון באמצעות PLL

בנוסף, כפי שלמדנו, ניתן להעלות את תדר השעון של המערכת בעזרת רכיב חומרה מסוג PLL הקיים בכרטיס הפיתוח שבו אנו משתמשים. רכיב זה מאפשר להכפיל או לשנות את תדר השעון הנכנס בהתאם לצרכי המערכת.

פירוט המערכת

בסעיף זה נציג סקירה כללית של המערכת שפיתחנו, וכן נפרט על תתי-המודולים המרכיבים אותה. עבור כל תת-מודול נסביר בקצרה את אופן פעולתו, נציג את תרשימי ה-RTL שלו לאחר ביצוע הסינתזה, נפרט את הלוגיקה שבה הוא משתמש, ונאתר את הנתבי הקריטי לפעולתו. את הנתבי הקריטי נציג באמצעות כלים שמספקת תוכנת Quartus.

מערכת ה-ALU

המערכת נועדה לממש מספר מודולים שונים לביצוע פעולות לוגיות ואריתמטיות. כל מודול פועל באופן עצמאי ונפרד מהאחרים. הבחירה באיזה מודול להשתמש מתבצעת באמצעות אות בחירה הניתן על ידי המשתמש, אשר מפעיל את המודול הרלוונטי בהתאם לערך שנבחר.

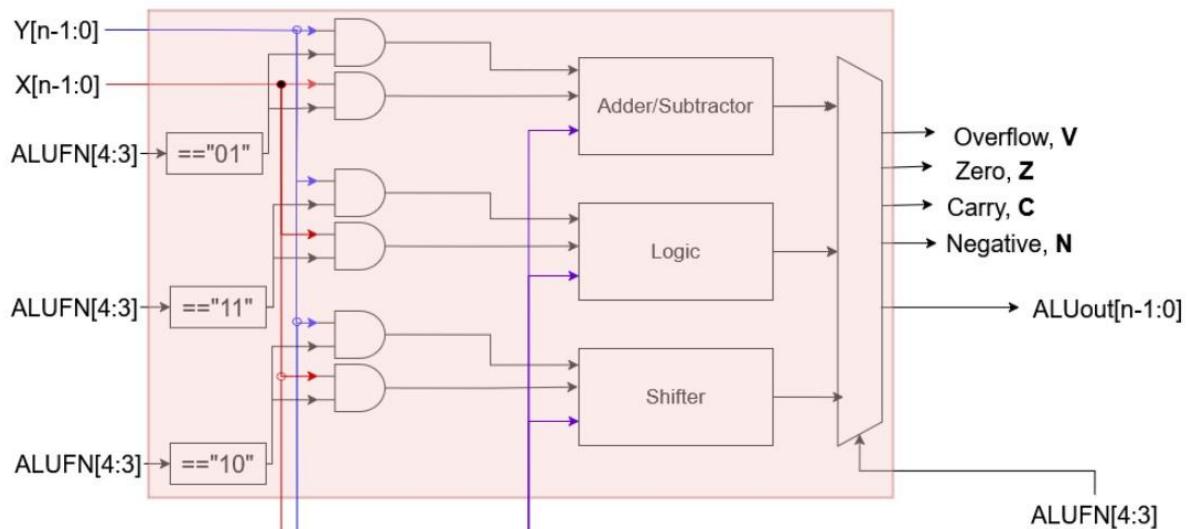


Figure 2: מודל המערכת

מערכת ה-ALU – מבנה וכניסות/יציאות

בסקירה הבאה נסביר על כל אחד מתתי-המודולים בנפרד. מערכת ה-ALU מקבלת שלוש כניסות עיקריות:

- אות כניסה X

- אות כניסה Y

- קו בקרה, ALUFN שבו:

- הביטים 3-4 משמשים לבחירת המודול:

- 01 מודול AdderSub

- 10 מודול Shifter

- 11 מודול Logic

- הביטים 0,1,2 משמשים כבקרת פעולה פנימית בתוך כל מודול, ומאפשרים לו לפעול במספר אופנים

שונים בהתאם לערכם.

יציאות המערכת כוללות ארבעה רכיבים:

- שלושה דגלים:

- N (Negative) מסמן אם התוצאה שלילית

- Z (Zero) מסמן אם התוצאה היא אפס

- C (Carry) מסמן אם התרחשה נשיאה (Carry)

- V(Overflow) מסמן אם התרחשה גלישה אריתמטית (Overflow) בפעולות חיבור/חיסור

- מוצא – ALUout מייצג את התוצאה של המודול שנבחר לפעולה, בהתאם לערך שבקו הבקרה ALUFN.

שרטוט הRTL

להלן entity של המודול –

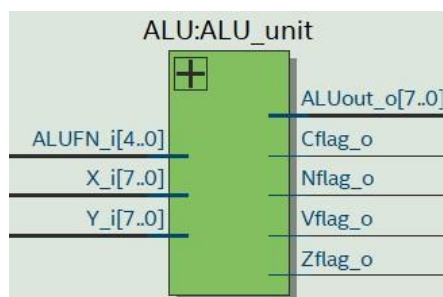


Figure 3: ALU Entity

שרטוט הRTL של מודול זה –

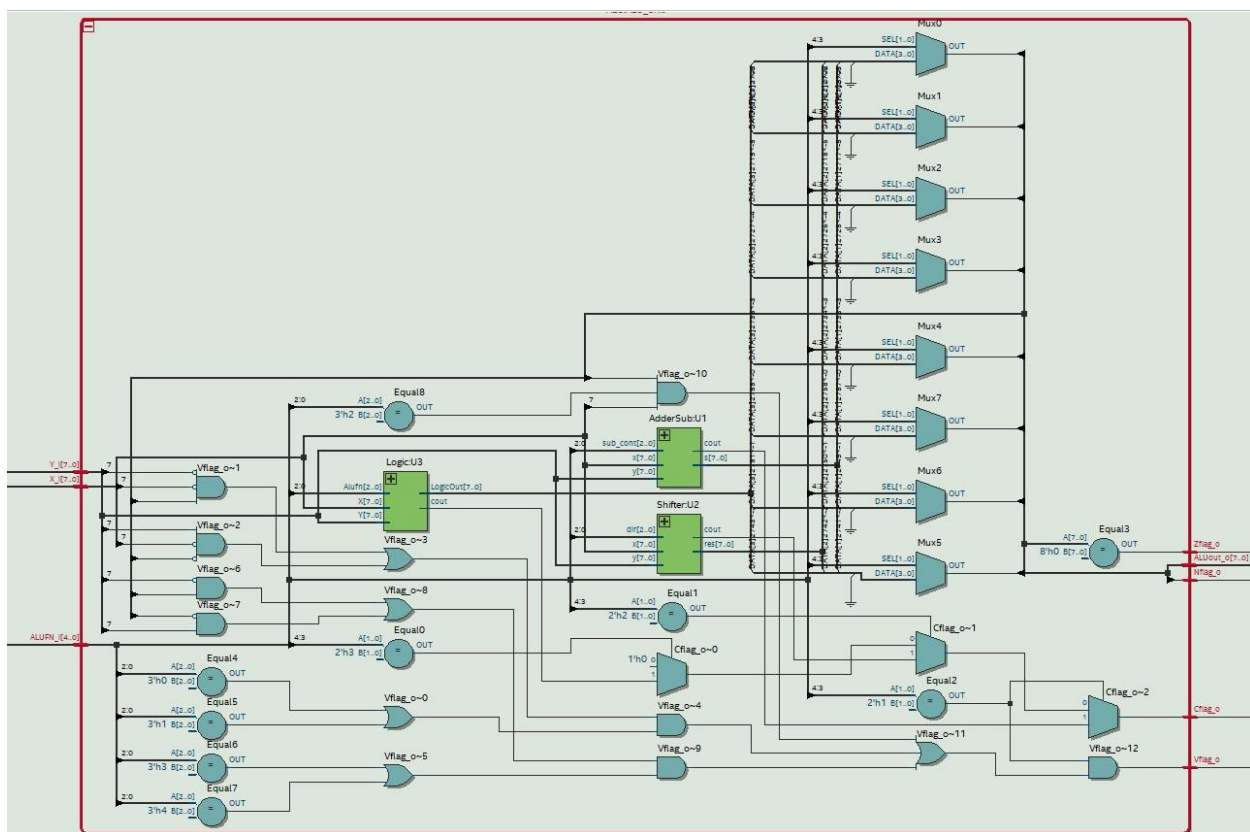


Figure 4: ALU RTL

שימוש בלוגיקה

כידוע, אנו מפתחים קוד לוגי המבוסס על רכיבים לוגיים הניתנים לקונפיגורציה בהתאם לקוד שנכתב – תהליך שמבוצע כחלק מתהליך הסינתזה. בניתוח זה נציג את הלוגיקה שנעשה בה שימוש עבור כל אחד מהמודולים, בהתאם לדרישות, ונפרט כיצד ממומשת כל פעולה ברמת השערים או המבנה הפנימי של המודול.

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	66
2		
3	Combinational ALUT usage for logic	94
1	-- 7 input functions	1
2	-- 6 input functions	34
3	-- 5 input functions	26
4	-- 4 input functions	16
5	-- <=3 input functions	17
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_i[0]
12	Maximum fan-out	38
13	Total fan-out	567
14	Average fan-out	2.91

Figure 5: System Logic Usage

נתיב קריטי

הנתיב הקריטי הוא מרכיב מרכזי להבנת זרימת המידע במערכת, והוא קובע את זמן ההתייצבות המינימלי הדרוש לפני שניתן לעבד קלט חדש. מאחר שלכל רכיב לוגי קיימת השהיה – גם כאשר הם פועלים במקביל – יש להביא בחשבון את זמן ההשהיה הכולל במערכת. לכן, איתור הנתיב הקריטי מאפשר לנו להעריך את ביצועי המערכת ולקבוע את התדר המקסימלי האפשרי לפעולה.

להלן מציאת הנתיב הקריטי עבור מודול זה, כפי שנעשה באמצעות כלי הניתוח של תוכנת Quartus :

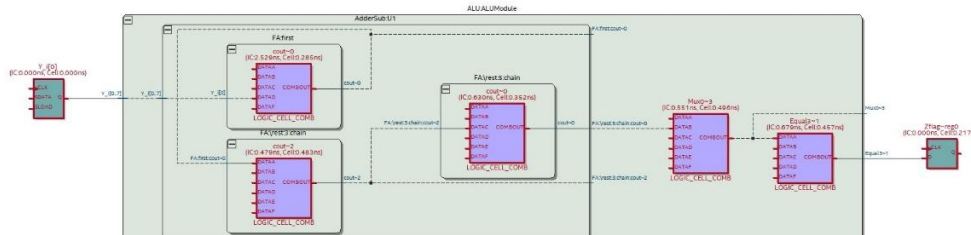


Figure 6: ALU Critical Path

מודול AdderSub

סקירת פעולת המודול

מודול זה מבצע, בהתאם לקו הבקרה ALUFN, אחת מתוך מספר פעולות אריתמטיות על שני סיגנלים באורך זהה – X ו-Y. הפעולות כוללות:

- חיבור (Addition) של X ו-Y באמצעות שרשרת חוסרי חיבור (Full Adders – FA)
- חיסור (Subtraction) באמצעות הוספת המשלים של X
- היפוך ביטים – (NEG) הפעלת היפוך ביט-ביט
- הוספה של 1 ל-Y (Increment) ביצוע $Y + 1$
- הפחתה של 1 מ-Y (Decrement) ביצוע $Y - 1$
- החלפת סדר בתים ב-Y (Swap) היפוך סדר הבתים של Y

הבחירה בפעולה המתבצעת נעשית לפי הביטים **ALUFN[2:0]**, כאשר כל ערך תואם לפעולה שונה.

שרטוט הRTL

להלן entity של המודול –

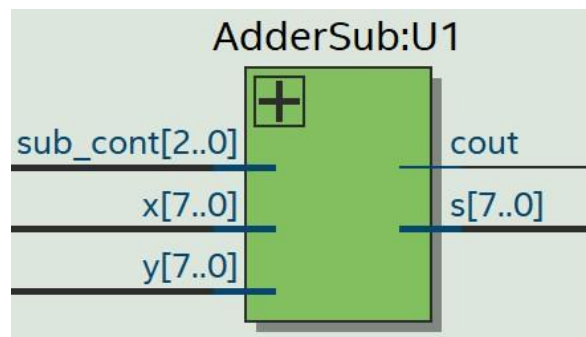


Figure 7: AdderSub Entity

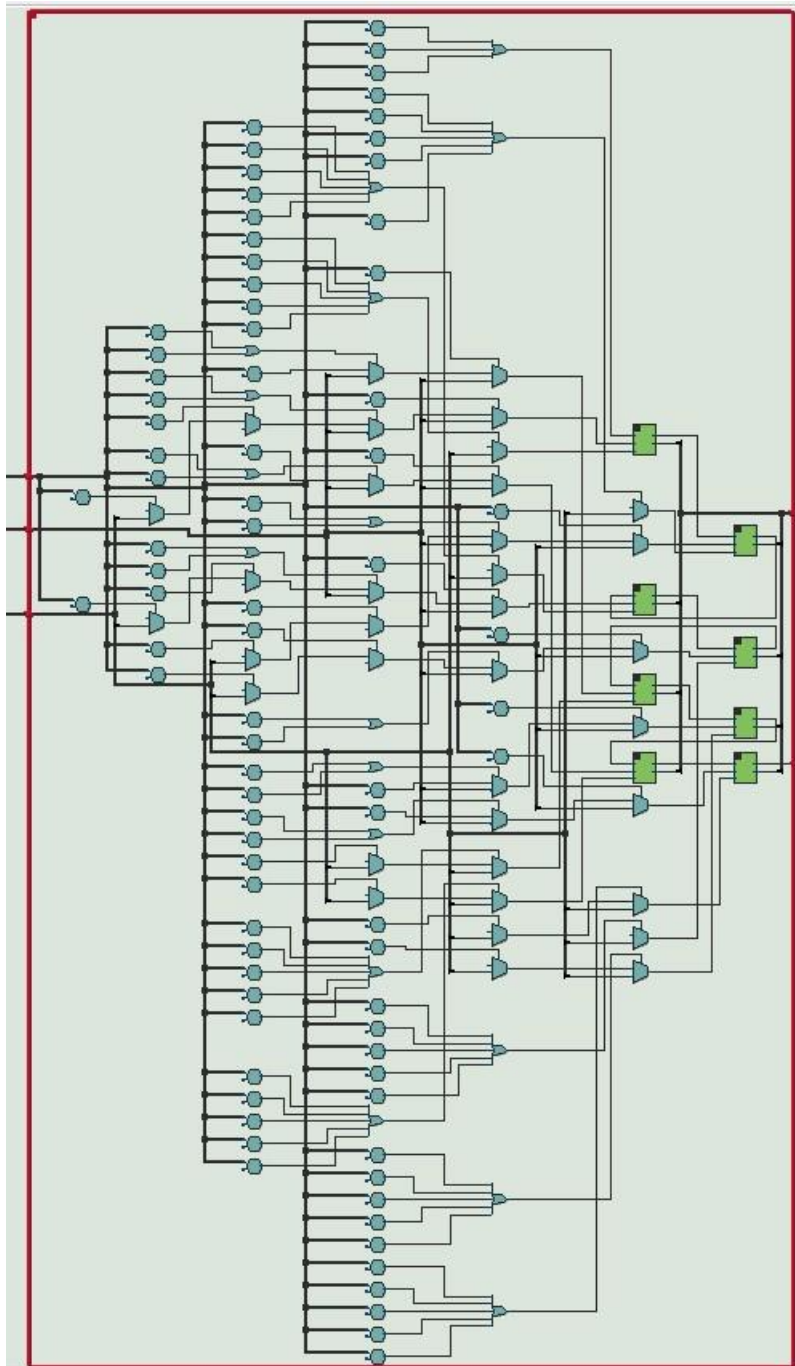


Figure 8: AdderSub RTL

מודול – Shifter

סקירת פעולת המודול

מודול זה מבצע פעולת הזזה (Shift) על בסיס מבנה **Barrel Shifter**, בהתאם לערכי קו הבקרה **ALUFN**. כאשר הביטים **ALUFN[2:0]** הם:

- – '000' מתבצעת הזזה שמאלה
- – '001' מתבצעת הזזה ימינה

למימוש המודול, עברנו על כל אחת מ- k השכבות כאשר $k = \log_2 n$ ובכל שכבה מתבצעת הזזה מותנית. עבור כל שכבה:

- אם הביט המתאים בשכבת הבקרה (ממוצא ה-X) הוא 0 – לא מתבצעת הזזה, והמוצא של השכבה זהה לכניסה
 - אחרת נבצע הזזה בהתאם לשכבה
- בנוסף, לצורך חישוב ה-Carry (סיבית שנדחפה החוצה במהלך ההזזה), יצרנו סיגנל ייעודי אשר שומר את ערכי ה-Carry בכל שלב, וממנו אנו מחזירים את הסיבית המתאימה (לרוב האחרונה) כמוצא carry של המודול.

שרטוט הRTL

להלן entity של המודול –

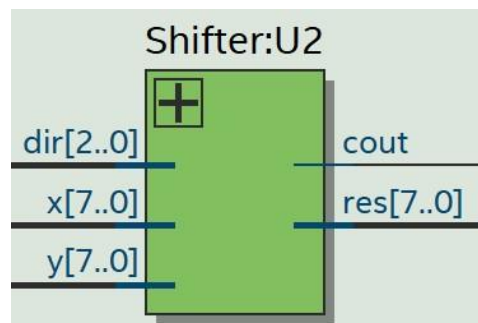


Figure 9: Shifter Entity

שרטוט הRTL של מודול זה –

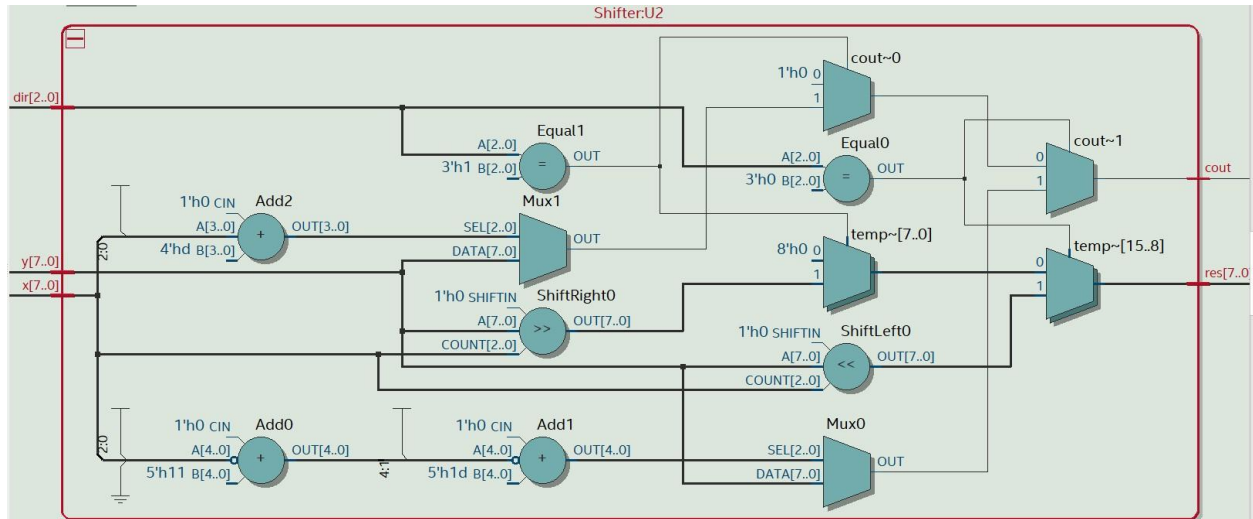


Figure 10: Shifter RTL

מודול – Logic

סקירת פעולת המודול

מודול זה מבצע פעולות לוגיות שונות בין הסיגנלים X , Y , ו- $ALUFN$. בהתאם לערכי קו הבקרה $ALUFN$.

הבחירה בפעולה המתאימה נעשית לפי ערכי הביטים $ALUFN[2:0]$, בהתאם להגדרות שניתנו ב-ISA-המצורפת.

מבנה המודול מאפשר הרחבה עתידית לפעולות לוגיות נוספות במידת הצורך, תוך שמירה על פשטות המימוש והיעילות בזמן סינתזה.

שרטוט הRTL

להלן entity של המודול –

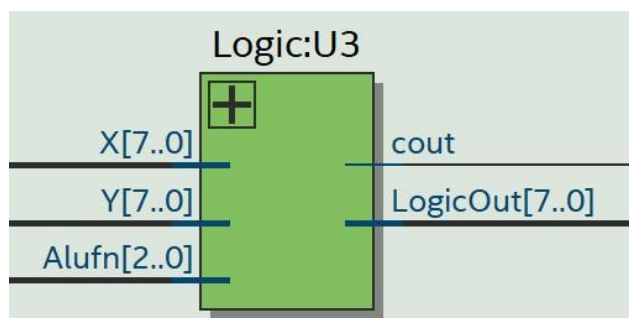


Figure 11: Logic Entity

שרטוט הRTL של מודול זה –

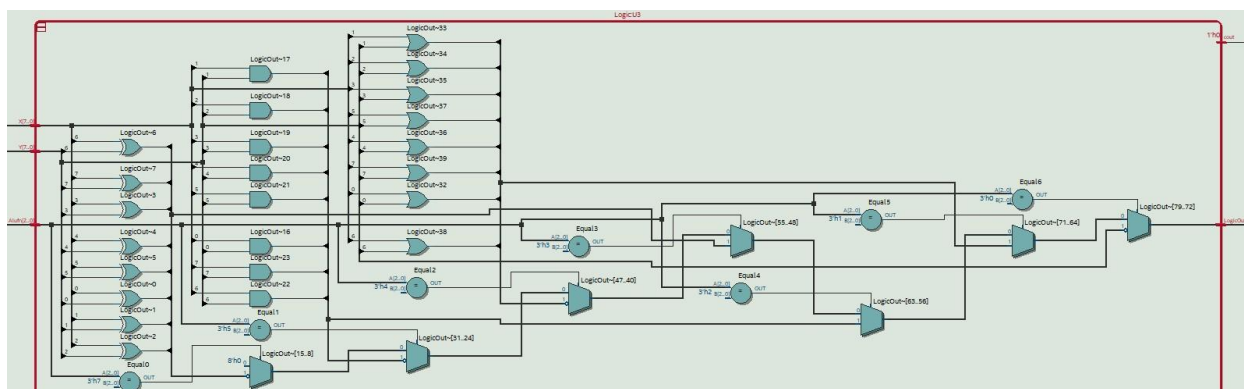


Figure 12: Logic RTL

מערכת ה PWM

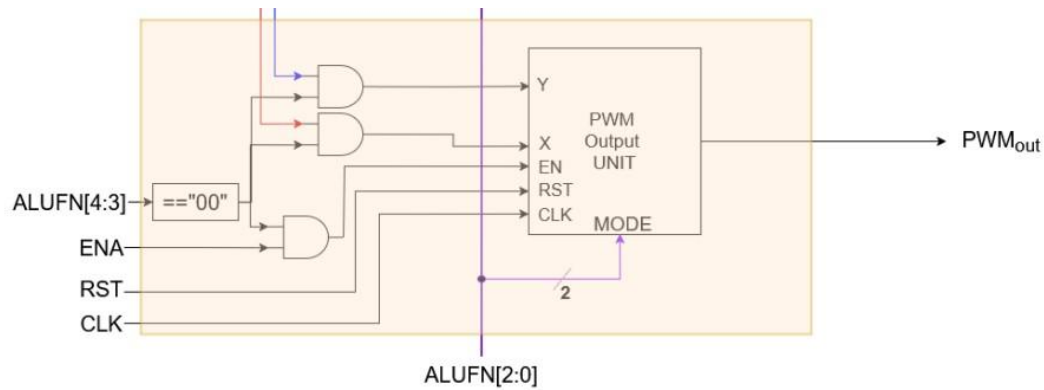


Figure 13: PWM Module

המערכת תקבל 6 כניסות ותייצר אות PWM בשלושה מצבים:

- אות כניסה X.
- אות כניסה Y.
- שעון CLK.
- קו ENA שעוצר או מפעיל את המערכת.
- קו RST שמאפס את המערכת
- קו בקרה ALUFN כאשר ביטים 0,1 קובעים את המודול הנבחר כך ש-
 - "00" נבחר PWM במצב Set/Reset
 - "01" נבחר PWM במצב Reset/Set
 - "10" נבחר PWM במצב Toggle

שרטוט הRTL

להלן הentity של המודול –

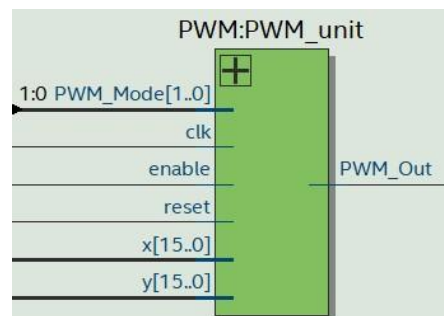


Figure 14: PWM Entity

שרטוט ה-RTL של מודול זה –

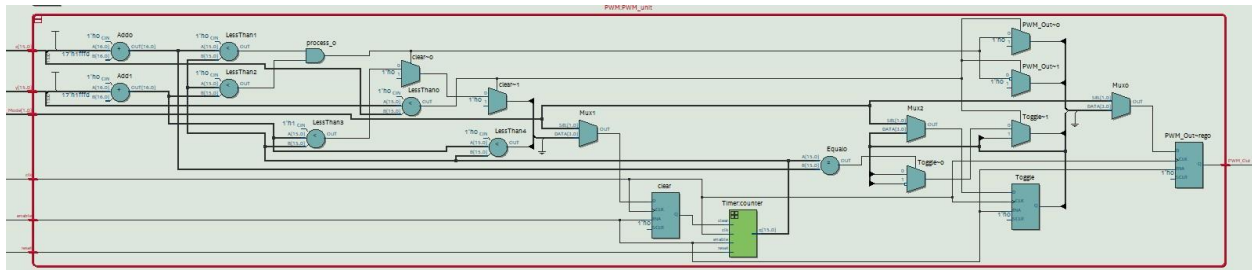


Figure 15: PWM RTL

בשימוש בלוגיקה

כידוע אנחנו מפתחים קוד לוגי אשר משתמש באלמנטים לוגיים שניתן לקנפג אותם בהתאם לקוד, דבר

שנעשה כחלק מההליך הסינתזה. בנייתו זה נציג את הלוגיקה עבור כל מודול כנדרש –


Analysis & Synthesis Resource Usage Summary		
 <<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	38
2		
3	▼ Combinational ALUT usage for logic	56
1	-- 7 input functions	0
2	-- 6 input functions	19
3	-- 5 input functions	1
4	-- 4 input functions	7
5	-- <=3 input functions	29
4		
5	Dedicated logic registers	11
6		
7	I/O pins	22
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	clk~input
12	Maximum fan-out	11
13	Total fan-out	275
14	Average fan-out	2.48

Figure 16: System Logic Usage

נתיב קריטי

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

להלן מציאת הנתיב הקריטי במודול זה –

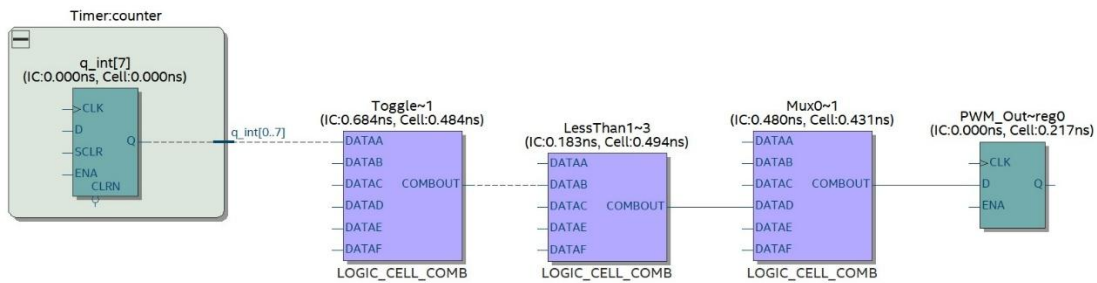


Figure 17: PWM Unit Critical Path

מודול – Timer

מודול זה מבצע מנייה כלפי מעלה עד שיגיע לערך של Y ומתחיל שוב מנייה מאפס.

שרטוט הRTL

להלן הentity של המודול –

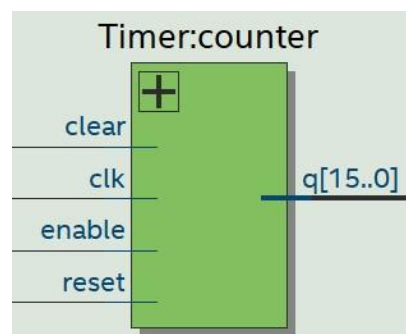


Figure 18: Timer Entity

שרטוט הRTL של מודול זה –

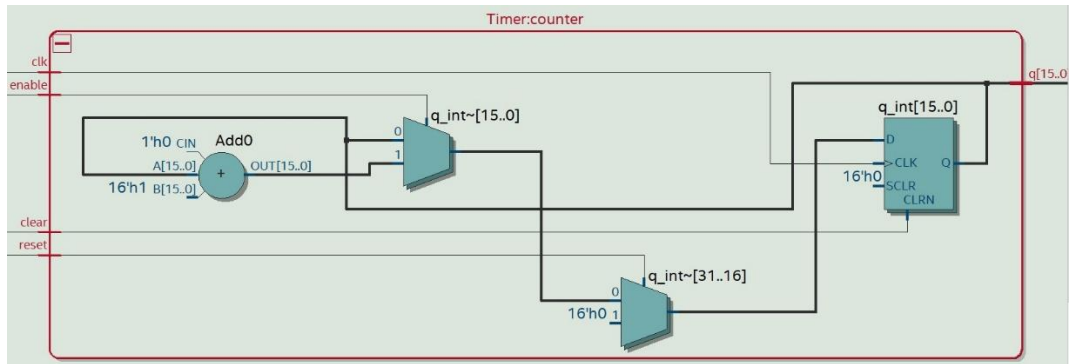


Figure 19: Timer RTL

בשימוש בלוגיקה עבור המערכת כולה-

Analysis & Synthesis Resource Usage Summary		
<<Filter>>		
	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	193
2		
3	▼ Combinational ALUT usage for logic	318
1	-- 7 input functions	1
2	-- 6 input functions	66
3	-- 5 input functions	32
4	-- 4 input functions	80
5	-- <=3 input functions	139
4		
5	Dedicated logic registers	19
6		
7	I/O pins	68
8		
9	Total DSP Blocks	0
10		
11	▼ Total PLLs	1
1	-- PLLs	1
12		
13	Maximum fan-out node	ALUFN_r[1]
14	Maximum fan-out	44
15	Total fan-out	1442
16	Average fan-out	3.04

Figure 20: System Logic Usage

הנתיב הקריטי עבור המערכת כולה הוא –

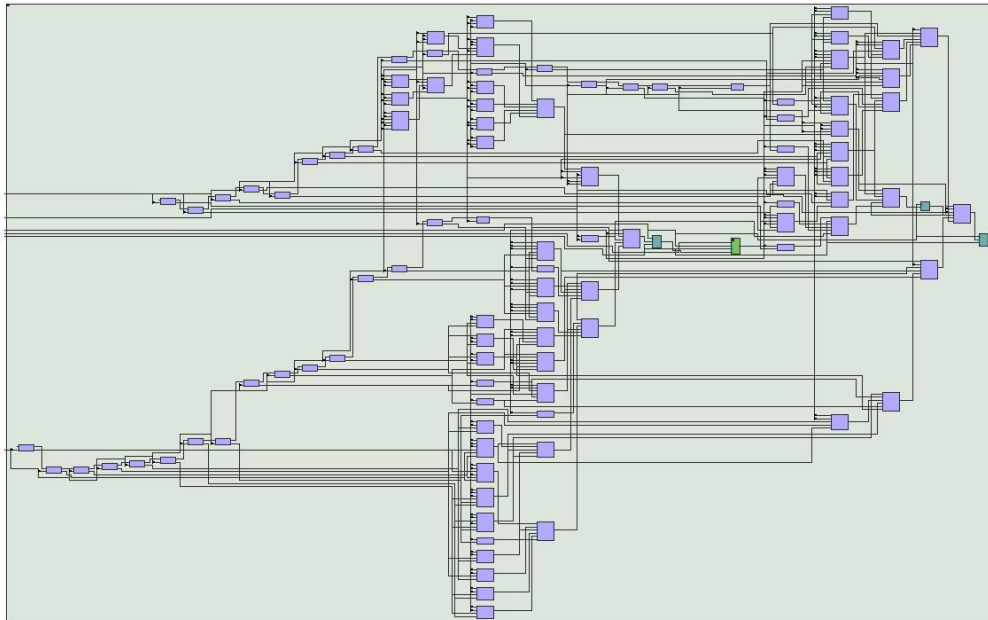


Figure 21: Top Critical Path

Signal Tap

לצורך ביצוע וורייפיקציה של החומרה שפיתחנו, השתמשנו בכלי **Signal Tap** של תוכנת **Quartus**. הכלי מאפשר לתפוס את מצבי הסיגנלים ברכיב בזמן אמת, ולנתח את התנהגות המערכת במהלך הפעולה.

במסגרת הניסוי, הגדרנו את הסיגנלים אותם נרצה ללכוד – ובפרט את סיגנלי **Keys**, נלכוד אותם בעת עליית המתח (Rising Edge), כלומר כאשר נשחרר את המפתח. במהלך הריצה, אנו מדפיסים ומנתחים את ערכי הכניסות והמוצאים של המערכת בזמן אמת, לצורך בדיקת תקינות וביצועים.

פעולת Shifter –



Figure 22: SHL Operation

בפעולה זו, נרצה לבצע הזזה שמאלה לא ציקלית על ידי ה- $\text{Opcode} = 10000$ של הווקטור Y כמות הפעמים של $X[0..2]$. כאן ההזזות היא 2 והערך של Y הוא 00000001 ולכן קיבלנו ב ALUout את הערך 04h שזה 00000100.

פעולת חיבור –



Figure 23: ADD Operation

בפעולה זו נרצה לבצע חיבור בין X ו Y ע"י ה- $\text{Opcode} = 01000$. ניתן לראות שהערכים של Y ו X הם 316 ולכן קיבלנו ב ALUout את הערך 09h.