***הפקולטה להנדסת חשמל ואלקטרוניקה***

**דוח ביניים לפרויקט גמר – תואר ראשון**

***מימוש מיקרופרוססור על FPGA הכולל פריפריות DSP***

***Implementation of Microprocessor Includes DSP Peripherals***

***מאת:***

***אריאל אוחיון #206912354***

***ליאור ידגרוב #315852848***

***מנחה: שמואל מעודה***

**תוכן עניינים**

**[תקציר](#_Toc123490854)** [4](#_Toc123490854)

[**רשימת איורים** 5](#_Toc123490855)

[**רשימת קיצורים** 6](#_Toc123490856)

[**פרק א' – מבוא לפרויקט** 7](#_Toc123490857)

[**רקע** 7](#_Toc123490858)

[**תיאור הבעיה** 8](#_Toc123490859)

[**פרק ב' – ניתוח תיאורטי** 9](#_Toc123490860)

[**סוגים של ארכיטקטורות מעבדים** 9](#_Toc123490861)

[**ארכיטקטורת MIPS** 11](#_Toc123490862)

[ **תכונות ומימוש Pipeline** 12](#_Toc123490863)

[ **בעיות Hazards במימוש Pipeline** 14](#_Toc123490864)

[**הקדמה לעיבוד אותות ושימושים שלהם** 15](#_Toc123490865)

[**תכונות של מערכות** 15](#_Toc123490866)

[ **פעולות מתמטיות בניתוח של אותות ומערכות** 15](#_Toc123490867)

[ **מערכות LTI** 15](#_Toc123490868)

[**דגימת אותות אנאלוגיים** 15](#_Toc123490869)

[ **ממירים ADC/DAC** 15](#_Toc123490870)

[**מסננים** 15](#_Toc123490871)

[ **מסננים אנאלוגיים (HPF,LPF)** 15](#_Toc123490872)

[ **מסננים דיגיטליים (FIR,IIR)** 15](#_Toc123490873)

[**מימוש אלגוריתמים לעיבוד תמונה** 15](#_Toc123490874)

[**מימוש אלגוריתמים לעיבוד קול** 15](#_Toc123490875)

[**תיאור הכלים והעזרים בסביבת הפיתוח** 16](#_Toc123490876)

[**רכיב FPGA** 16](#_Toc123490877)

[ מבנה הרכיב FPGA 16](#_Toc123490878)

[ שלבי פיתוח 17](#_Toc123490879)

[**שפות תיאור חומרה HDL** 17](#_Toc123490880)

[**תוכנות Quartus וModelsim** 18](#_Toc123490881)

[**הכרטיסים שבחרנו לעבוד איתם** 18](#_Toc123490882)

[**נספחים** 20](#_Toc123490883)

# **תקציר**

הפרויקט עוסק בתכנון חומרה של מיקרופרוססור בארכיטקטורת MIPS 5 Stage Pipeline.  
מימוש החומרה בפרויקט יבוצע באמצעות צריבה לרכיב FPGA.  
תחילה נציג רקע כללי על ארכיטקטורות שונות של מיקרופרוססורים ובעיקר נרחיב על ארכיטקטורת MIPS.  
נתאר רקע תיאורטי בנושאי העיבוד אותות ועל האופן שבו ניתן לשלב בין יחידות לעיבוד אות לרכיב FPGA.  
נדבר על כלי העבודה של סביבת הפיתוח שיעזרו לנו בשלבי הפרויקט כגון: תוכנה לניהול גרסאות, כרטיסים אלקטרוניים, תוכנות וכלים לצרכי Debug וסימולציות.  
נתאר את אופן העבודה עם רכיבי FPGA וההתמודדות עם בעיות.

בהמשך נתאר את המבנה הכללי של רכיב ה- FPGA שבחרנו לעבוד אתו בפרויקט ושלבי הפיתוח שביצענו.

מטרת הפרויקט היא לאפשר ממשקי DSP למיקרופרוססור על מנת לפתור בעיות בתחום העיבוד אותות באופן יעיל ופשוט.

**Abstract**

The project based on hardware design of MIPS 5 Stage Pipeline Microprocessor.

The hardware will be implemented on FPGA component.  
At first we will show some background on microprocessor architecture and specially the structure of the MIPS architecture.

We will describe theoretical info of DSP and how to interface the FPGA with the DSP components.

Also we will talk about the development environment tools that will help us to implement the project: version management software, Electronic Cards, software and debug tools and simulations.

We will describe how to work with FPGA components and deal with problems.

Then, we will present the specific FPGA component that we chose for this project, describe the development stages.

Our goal is to allow a DSP interfaces for microprocessor in order to analyze and solve signal processing problems.

# **רשימת איורים**

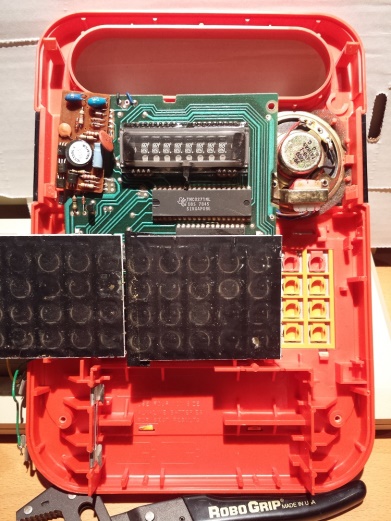
# **רשימת קיצורים**

SNR – Signal to Noise Ratio יחס אות לרעש

# **פרק א' – מבוא לפרויקט**

## **רקע**

בשנת 1976, פותחה מערכת עיבוד אותות הראשונה על ידי חברת Texas Instruments הנקראת Spell & Speak. מערכת זו פותחה כמשחק ילדים אך שימשה כמהפכה טכנולוגית בעולם עיבוד האותות.  
המערכת מימשה משחק איות של מילים ואותיות באנגלית, על ידי ממשק משתמש של מקלדת ורמקול המחוברים לכרטיס אלקטרוני שמכיל בקר (TMS5100).  
הבקר בכרטיס הכיל סט פקודות שאפשרו ביצוע עיבוד אותות שמע (פיתוח אות שמע מהרמוניות של אותות סינוסואידליים) .



איור 1 - המכשיר Speak & Spell

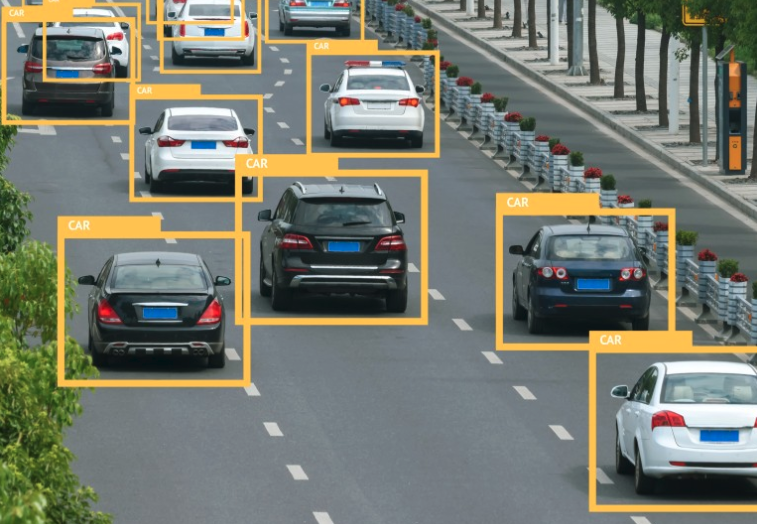
איור 2- הכרטיס האלקטרוני במכשיר

מאז, עולם האלקטרוניקה נחשף יותר ויותר למעבדים המאפשרים:

1. פעולות מתמטיות מורכבות.
2. קריאה וכתיבה של נתונים אנאלוגיים על ידי ממירים לתחום האנאלוגי.
3. מימוש מסננים במעבדים.

המטרה העיקרית היא לייעל את החומרה והתוכנה של המעבדים, כדי לעבד את הנתונים המתקבלים מהאותות האנאלוגיים הנקלטים מהסביבה שמחייבים פענוח מהיר (לדוגמא: אותות שמע, אותות וידאו, אותות תקשורת).

פיתוח מערכות בתחום עיבוד האותות מהווה אחד מהגורמים העיקריים בהתקדמות הטכנולוגית.  
כיום, בעולם המחשוב ניתן לבצע עיבוד תמונה לזיהוי עצמים, עיבוד אותות קול (סינתזה – יצירה של קולות חדשים, וניתוח ספקטרלי של אותות שעברו דגימה), קידוד ופענוח של אותות תקשורת המועברים בכבל ובצורה אלחוטית, ועוד.



איור 3- דוגמא לשימוש בעיבוד תמונה

המטרה שלנו בפרויקט היא לייצר ארכיטקטורה לחומרה של מעבד אשר יכלול בתוכו סט פקודות שיאפשרו מימושים בסיסיים ליישומים בתחום עיבוד אותות.

## **תיאור הבעיה**

הטכנולוגיה הנוכחית דורשת ממהנדסים בתחום החשמל והאלקטרוניקה לחשוב על פתרונות במגוון תחומי החיים לצורך שיפור איכות החיים.  
הנושא המרכזי סביבו מתמקדים מהנדסים ואנשי פיתוח רבים הוא כיצד ניתן לייצר מערכות המשלבות חומרה ותוכנה על מנת לפתור בעיות בחיי היום יום וכיצד ניתן לקדם תחומים כמו תקשורת, רפואה, תחבורה ועוד לכיוון של אוטומציה והתייעלות באמצעות כלי הפיתוח.  
הרכיבים האלקטרוניים המסוגלים לפתור בעיות מסוג זה על ידי עיבוד נתונים מהיר הם רכיבים מתוכנתים.  
תחום הרכיבים המתוכנתים מחולק למיקרובקרים ורכיבי FPGA.

היתרון העיקרי של רכיבי FPGA על פני המיקרובקרים הוא שניתן להגדיר חומרה לרכיב FPGA ולצרוב אותה עליו, הדבר מאפשר לעבוד עם סט שעונים מהיר. במיקרובקר החומרה קיימת כבר ברכיב ומוגדרת לתדר שעון מקסימלי (שהוא נמוך יותר מתדר השעון שניתן לעבוד עליו ברכיב FPGA).  
לכן, בחרנו לתכנן את המערכת שלנו בפרויקט על רכיב FPGA.

# **פרק ב' – ניתוח תיאורטי**

## **סוגים של ארכיטקטורות מעבדים**

ניתן לסווג את הארכיטקטורות השונות לפי פרמטרים שונים התלויים במימוש כמו: סט ההוראות של המיקרופרוססור, אופן הקידוד של ההוראות, מבנה הזיכרון.

בסיכום זה נסווג את הארכיטקטורות לפי מבנה הזיכרון.

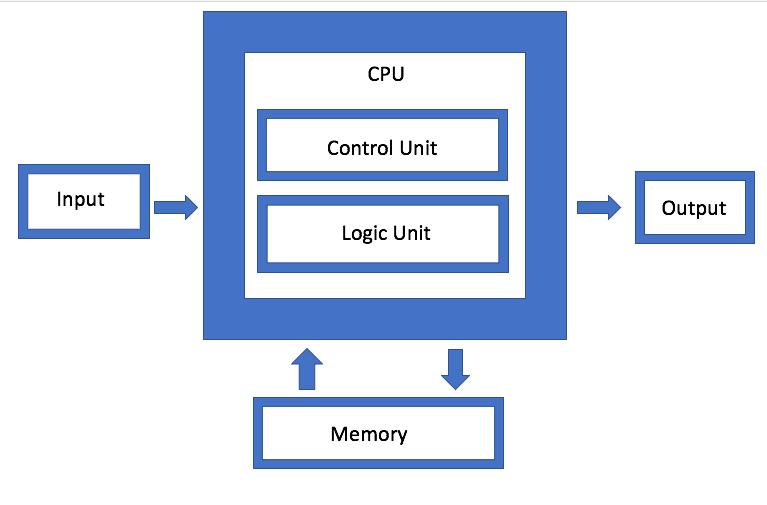
**ארכיטקטורת Von Neumann:**

הארכיטקטורה בנויה באופן כזה שהזיכרון הראשי שהמעבד עובד אתו כולל פקודות שהתוכנית מריצה (Instruction Memory) ואת הנתונים שהמעבד קורא\כותב לזיכרון במהלך ריצת התוכנה (Data Memory).

בעיה נפוצה של שימוש בשיטת תכנון לפי ארכיטקטורה כזו היא "צוואר בקבוק הפון נוימן".

בעיה זו מתארת התנגשות בין קריאה\כתיבה לזיכרון הנתונים במהלך התוכנית, לבין שליפת הפקודה הבאה לביצוע מהזיכרון.  
אלו שתי פעולות שהמעבד מבצע אותן עבור כל הוראה שכתובה בתוכנית ויפורטו בפרק של תכונות ומימוש Pipeline.

בעיה נוספת חמורה עוד יותר בארכיטקטורה זו, היא שהמעבד מסוגל לרשום נתוני מידע שהוא משתמש בהם במהלך התוכנית על כתובות בזיכרון ששמורות עבור הפקודות שהמעבד צריך לבצע.



איור 4 – מבנה ארכיטקטורת Von Neumann

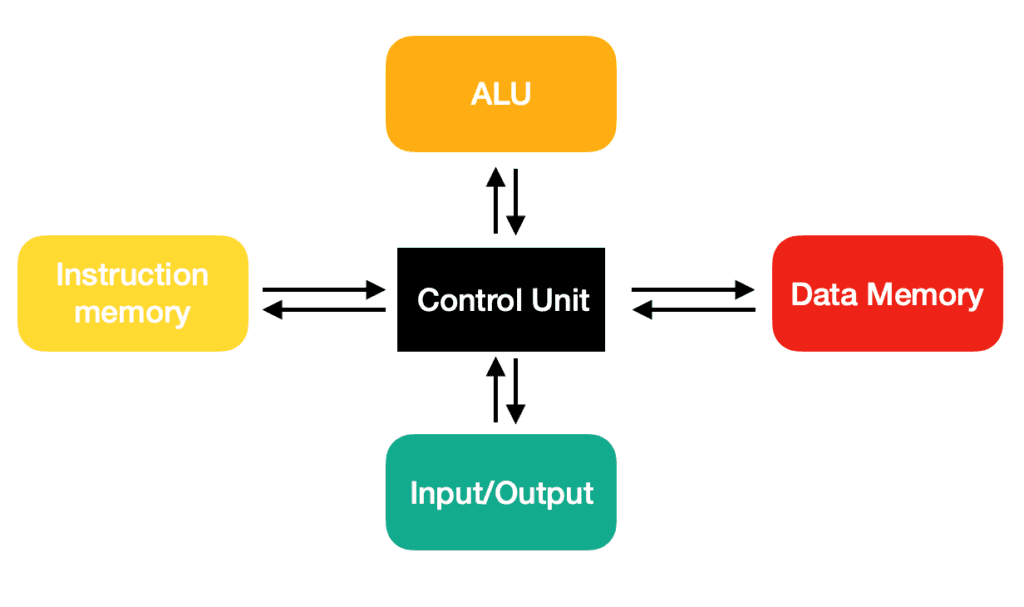
**ארכיטקטורת Harvard:**

הארכיטקטורה בנויה בצורה בה קיימים שני רכיבי זיכרון

1. מיועד לשמירת ההוראות לביצוע (Instruction Memory).
2. משמש לצורך שמירת הנתונים במהלך התוכנית (Data Memory).

ארכיטקטורה זו פותרת את הבעיות שנוצרו משימוש בזיכרון בודד גם להוראות וגם לנתונים אך דורשת יותר משאבים לצורך תכנון שני מודולים של זיכרון בארכיטקטורה.

שיטת עבודה בארכיטקטורה זו מאפשרת כבר את העבודה הבסיסית עם מיקרופרוססורים שמיושם בהם Pipeline.



איור 5 - ארכיטקטורת Harvard

## **ארכיטקטורת MIPS**

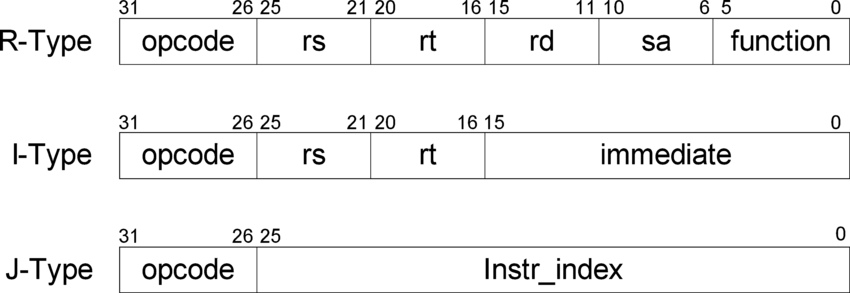
ארכיטקטורת MIPS (Microprocessor without Interlocked Pipeline Stages) היא ארכיטקטורה המממשת סט פקודות למעבדים מסוג RISC.  
הנתונים במעבד מורכבים מ-32 סיביות והוא מומש בשני סוגים של ארכיטקטורות שכל אחת מהן עובדת בצורה שונה:

1. Single Cycle Architecture – ארכיטקטורה מסוג זה הייתה מקבלת פקודה ומבצעת רק אותה עד שהייתה מסתיימת ורק אז שולפת את הפקודה הבאה מזיכרון התוכנית.
2. Pipelined Architecture – ארכיטקטורה מסוג זה בנויה באופן שבו ביצוע פקודה מפורק ל-5 שלבים (נקראת גם 5 Stage Pipeline) בזמן שחלק מהחומרה בארכיטקטורה מבצע שלב מסוים בביצוע הפקודה, חלקים אחרים בחומרה מבצעים שלבים אחרים עבור הפקודות הבאות שהמעבד צריך לבצע.

ארכיטקטורה זו יעילה יותר מה-Single Cycle ולכן בפרויקט החלטנו לממש אותה.

סט הפקודות המוגדר עבור ארכיטקטורת MIPS מורכב משלושה סוגים של הוראות:

1. R-Type Instruction: הוראה שמתבצעת על שני אוגרים (Registers) ומכניסה את התוצאה לאוגר נוסף.
2. I-Type Instruction: הוראה שמבצעת פעולה על אוגר וערך מידי שקיים כבר בקידוד של הפקודה ומכניסה את התוצאה לאוגר נוסף.
3. J-Type Instruction: הוראה שמבצעת בדיקה לתנאי מסוים, אם התנאי מתקיים התוכנית מבצעת קפיצה להוראה אחרת בתוכנית שלא לפי הסדר.



איור 6 – Instruction Type Format

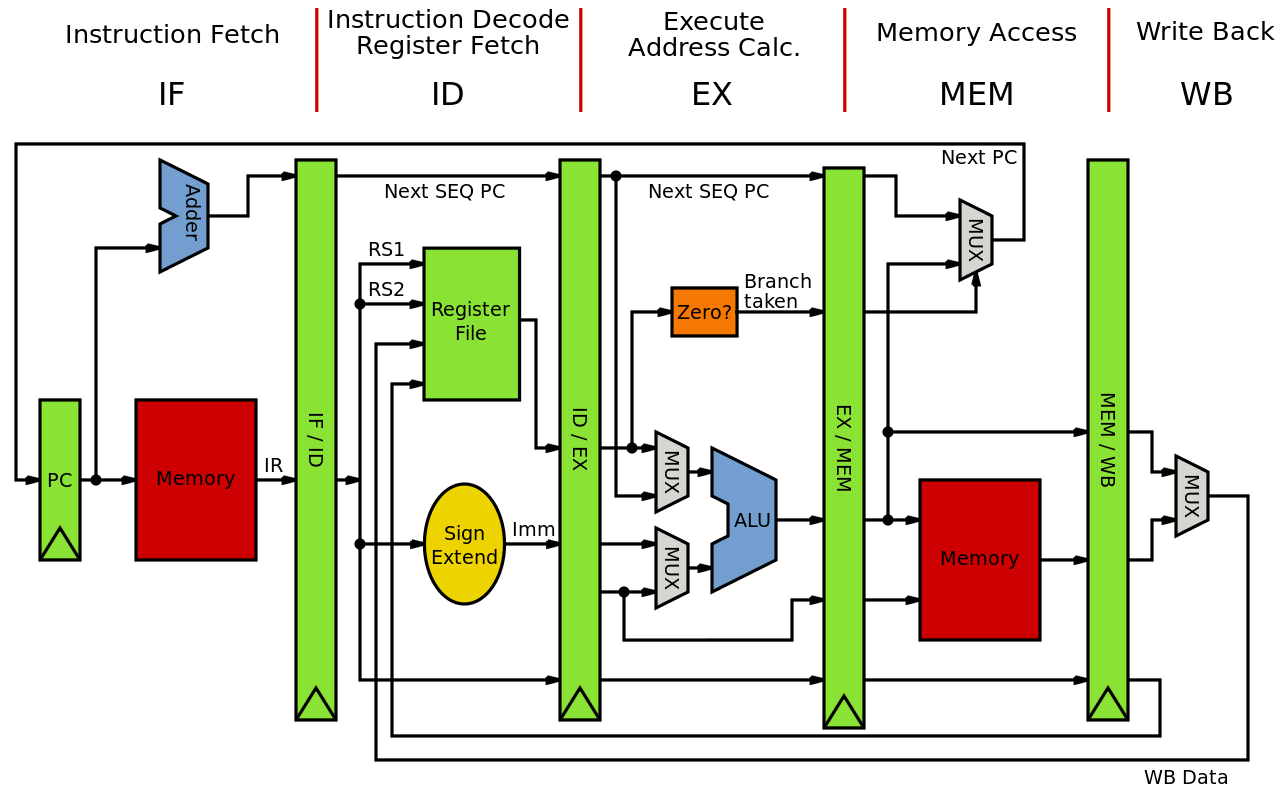
סט הפקודות המלא נמצא בנספחים. **(אפשר להוסיף הסבר מפורט יותר על קידוד ההוראות בMIPS)**

### **תכונות ומימוש Pipeline**

כפי שכבר צוין בפרויקט נרצה לממש ארכיטקטורת 5 Stage Pipeline בה כל הוראה בתוכנית עוברת 5 שלבים.  
לכל שלב יש חומרה משלו שמבצע את אותו השלב בביצוע הפקודה, בין השלבים יש חוצץ שאוגר את הנתונים של השלב הקודם ומעביר אותו לשלב הבא.

על ידי החוצצים בין השלבים ניתן לסנכרן את הפעולות שמתבצעות בין כל שלב בביצוע הפקודה.

ניתן לראות באיור מטה את ארכיטקטורת MIPS - 5 Stage Pipeline



איור 7 – MIPS 5 Stage Pipeline Architecture

שלבי ביצוע של ההוראות:

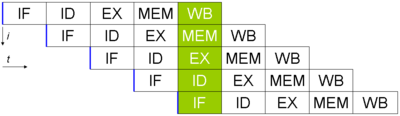
1. שלב Instruction Fetch (שליפת הוראה) – בשלב זה המיקרופרוססור מוציא הוראה מזיכרון התוכנית לאוגר IF/ID.

החומרה הדרושה לביצוע פעולה זו היא אוגר שיצביע על הכתובת בזיכרון בו נמצאת הפקודה שרוצים לבצע (PC), זיכרון התוכנית (Instruction Memory) ומחבר שמוצאו יחובר לאוגר PC כדי להעלות את ערכו בערך המתאים לכתובת בה נמצאת ההוראה הבאה בזיכרון התוכנית.

1. שלב Instruction Decode (פענוח הוראה) – בשלב זה המיקרופרוססור מחלק את סיביות ההוראה לשתי צורות עיקריות של פענוח: האחת, שידע לאיזה אוגרים ההוראה מופנית, והשנייה שיבדוק אם קיימים בהוראה ערכים מיידים שצריך לבצע איתם פעולה.  
   החומרה הדרושה לביצוע שלב זה היא האוגרים של המעבד.
2. שלב Execute (ביצוע) – בשלב זה החומרה הדרושה היא ALU, יודעת לעשות את הפעולות המתמטיות (אריתמטיות ולוגיות) שמוגדרות לה בסט ההוראות, ובנוסף זו יחידה שיודעת לבצע את הבדיקות הדרושות לצורך קפיצות בתוכנית (בדיקת תנאים).
3. שלב Memory (זיכרון נתונים) – בשלב זה המעבד כותב לזיכרון הנתונים (Data Memory) את התוצאה שיצאה מהשלב הקודם (מהיחידה של ה-ALU).  
   שלב זה אופציונלי שהרי לא תמיד כל נתון אנו כותבים בזיכרון הנתונים.
4. שלב Write Back (כתיבה בחזרה - משוב) – בשלב זה המעבד מבצע השמה של התוצאה שהתקבלה מהשלב השלישי ב-Pipeline באחד מהאוגרים של המעבד.

גם שלב זה אופציונלי שהרי לא תמיד כל נתון אנו כותבים בחזרה לאוגרי המעבד.

החוצצים בארכיטקטורה מסוג Pipeline מאפשרים לנו לבצע את הפקודות בתוכנית באופן יעיל כך שכאשר הוראה נכנסת לשלב השני (Instruction Decode) ההוראה הבאה אחריה נכנסת לשלב הראשון (Instruction Fetch), וכשאותה הוראה עוברת לשלב השלישי (Execute) ההוראה הבאה אחריה נכנסת לשלב השני וההוראה שתבוא אחריה תיכנס לשלב הראשון.



איור 8 – סדר ביצוע הוראות בארכיטקטורת Pipeline

לכן, שיטה זו נחשבת לשיטה יעילה בתכנון מיקרופרוססורים אך היא כוללת כמה בעיות שעלינו להתגבר עליהם.

### **בעיות Hazards במימוש Pipeline**

שגיאה מסוג Hazard קורית כאשר הוראה שנמצאת באחד משלבי הביצוע תלויה בתוצאה של הוראה שקדמה לה שעדיין לא הסתיימה.

דוגמה:

addi $t1,$t2,5

addi $t3,$t1,1

אם ההוראה הראשונה נמצאת בשלב MEM (כתיבה לזיכרון הנתונים) אז ההוראה שאחריה נמצאת בשלב Ex (ביצוע הפקודה על ידי ה-ALU) והתוצאה לא יכולה להיכנס לאוגר $t3 עד שלא יתבצע השלב WB (כתיבה לאוגר $t1) בהוראה הראשונה.

בעיות מסוג זה עלולות לבצע פעולות לא רצויות והשמת ערכים שגויים שהמתכנת לא התכוון אליהם.

אחת הדרכים לפתרון בעיה זו היא להוסיף לארכיטקטורה חומרה נוספת (Hazard Detection) אשר בודקת את התלות בין הוראות ועושה את ההשהייה המתאימה עד לסיום הפקודה.

קיים עוד סוג של בעיית Hazard מסוג Control Hazard, בעיה זו מתרחשת כאשר קיימת בתוכנית הוראה שבודקת תנאי לקפיצה בתוכנית, התנאי נבדק בשלב השלישי בPipeline - Ex (בALU) ומכיוון שהבדיקה רק בשלב השלישי, בינתיים נכנסת ההוראה הבאה אחריה לפי הסדר בשלבים הקודמים לה ב-Pipeline ולא מתבצע קפיצה בתוכנית כפי שתוכנן על ידי המתכנת.

גם בבעיה מסוג זו יהיה עלינו לתכנן השהייה בהתאם כדי שהקפיצה בתוכנית תתבצע מבלי לפגוע בנתונים שהמתכנת עובד איתם.

## **הקדמה לעיבוד אותות ושימושים שלהם**

## **תכונות של מערכות**

### **פעולות מתמטיות בניתוח של אותות ומערכות**

### **מערכות LTI**

## **דגימת אותות אנאלוגיים**

### **ממירים ADC/DAC**

## **מסננים**

### **מסננים אנאלוגיים (HPF,LPF)**

### **מסננים דיגיטליים (FIR,IIR)**

## **מימוש אלגוריתמים לעיבוד תמונה**

## **מימוש אלגוריתמים לעיבוד קול**

# **תיאור הכלים והעזרים בסביבת הפיתוח**

## **רכיב FPGA**

רכיב FPGA הוא רכיב אשר ניתן לתאר המבנה החומרתי שלו על ידי שפות תיאור חומרה (HDL).

לצורך תכנון חומרה על רכיב FPGA נדרש ידע והבנה במעגלים ספרתיים וידע בשפות תיאור חומרה ברמה שבה המתכנן יודע לקנפג את החומרה בצורה יעילה באמצעות כתיבת קוד על סמך מה שהוא נדרש לבצע.

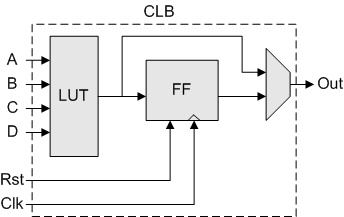
בעצם ניתן לפתח חומרה ספרתית (דיגיטלית) על גבי הרכיב FPGA באמצעות כתיבת קוד.

קיים סוג נוסף של רכיב מתוכנת הנקרא CPLD, רכיב זה בעל משאבים איטיים יותר והוא יותר מוגבל מבחינת היכולות שלו לעומת הרכיב FPGA.

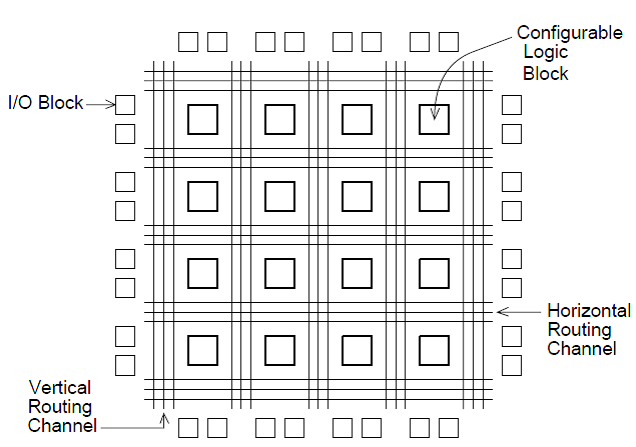
בפרויקט שלנו החלטנו לממש את החומרה על רכיב FPGA.

### מבנה הרכיב FPGA

רכיב FPGA מורכב מיחידות המכונות אלמנטים לוגיים (LE – Logic Elements) נקרא גם CLB.

כל יחידה לוגית כזו מורכבת ממעגל צירופי הנקרא LUT שיכול להיות ממומש כשער לוגי, דלגלג (Flip Flop), ומרבב במוצא הרכיב.

כל אלמנט לוגי כזה מהווה אבן הבניין למעגל אלקטרוני דיגיטלי שניתן באמצעותו לתכנן חומרה עבור מערכות מורכבות.

ברכיב FPGA יש כמות גדולה של אלמנטים לוגיים שמחווטים ביניהם כך שניתנת היכולת למפתח ליצור מעגל ספרתי מורכב. **(אפשר להרחיב על PLL, זיכרונות פנימיים, וDSP)**

### שלבי פיתוח

פיתוח חומרה על גבי רכיב FPGA דורש להעביר את החומרה 3 תהליכים עד לקבלת ההחלטה שהחומרה עובדת כראוי.

1. שלב ראשון – סינתזה

סינתזה זה התהליך הראשוני שבו מנסים להוריד את הרעיון של הרכיב שרוצים לתכנן לקוד שמתאר את החומרה.

לעיתים הפעולה שנרצה שהחומרה תבצע תהיה פעולה פשוטה ואז נוכל לבטא את החומרה באמצעות קוד בשיטת Dataflow Modeling, ולעיתים קרובות יותר כאשר נרצה לתאר חומרה מורכבת נרצה לתאר אותה בשיטת Behavioral Modeling.

בסוף התהליך של תיאור החומרה באמצעות HDL מתקבל שרטוט סכמתי שמתאר את החומרה (הסכימה מכונה RTL).  
בדרך כלל נעבור על המסלול נתונים של הסכימה הלוגית ונוודא שהמערכת שתכננו עומדת בדרישות שלנו, בנוסף בדרך כלל ננסה להימנע מנועלים (Latch) הממומש במערכות שאנו מתכננים מכיוון שהם עלולים לגרום לאותות במערכת לצאת מסנכרון של השעון הכללי שמסופק למערכת.

בסוף שלב הסינתזה נוצר דו"ח שנותן לנו נתונים על המשאבים שהחומרה שכתבנו צורכת מהרכיב FPGA שאיתו אנו עובדים.

1. שלב שני – סימולציה

בשלב זה אנו נדרשים לכתוב Script (הנקרא Test Bench) שמריץ מספר מסוים של אפשרויות בכניסה של המערכת החומרתית ומתקבל תרשים זמנים (Waveform) שבו ניתן לוודא אם הרכיב מבצע את הפעולה הדרושה. הסימולציה רצה על תוכנה לפני צריבת החומרה על הרכיב FPGA.  
מטרת הסימולציה לגלות בעיות בתכנון הרכיב לפני הצריבה.  
קיימת שיטה יעילה שמשתמשים בה בתעשייה לצורך ביצוע סימולציה המכונה UVM.

1. שלב שלישי – הרצה ודיבוג תקלות

לאחר הרצה רצוי לבדוק את האותות במערכת לצורך זיהוי תקלות (בדרך כלל ניתן לבדוק את האותות במערכת על ידי Logic Analyzer אנו נשתמש בכלי SignalTap של Intel).

## **שפות תיאור חומרה HDL**

יש 3 שפות HDL נפוצות בתעשייה לכתיבת חומרה ברכיבי FPGA: VHDL, Verilog, SystemVerilog.

השפה הנכונה לכתוב בה את הסינתזה של המודול היא VHDL על אף שהיא יותר קשוחה משאר השפות HDL ולכן נתכנן איתה את המודולים בפרויקט.  
השפת HDL שאיתה נבצע את הסימולציות תהיה SystemVerilog שהיא מאוד דומה לVerilog מבחינת סגנון הכתיבה אך היא כוללת עוד תוספת שנועדו להקל בסביבת הסימולציה.

## **תוכנות Quartus וModelsim**

## **הכרטיסים שבחרנו לעבוד איתם**

**ביבליוגרפיה**

1. Edenhofer, O., Pichs-Madruga, R., Sokona, Y., Seyboth, K., et al.: (2012) Annex 2. In *IPCC Special Report on Renewable Energy Sources and Climate Change Mitigation*. Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.
2. Nordhaus, W. D. (2017) Revisiting the social cost of carbon. *Proceedings of the National Academy of Sciences, 114(7),* 1518-1523. National Academy of Sciences. Washington, D.C., USA.
3. Rubinstein, F., Neils, D., & Colak, N. (2001) Daylighting, Dimming, and the Electricity Crisis in California. *Presented at the Illuminating Engineering Society of North America National Conference.*
4. Saadat, H. (1999) *Power system analysis*. Boston, MA, USA: McGraw Hill.
5. Sonmez, Y. (2013) Estimation of fuel cost curve parameters for thermal power plants using the ABC algorithm. *Turkish Journal of Electrical Engineering & Computer Sciences, 21,* 1827-1841.
6. United Nations. (2015) *Adoption of the Paris agreement.* Framework Convention on Climate Change. Paris, France.

# **נספחים**

1. <https://en.wikipedia.org/wiki/Digital_signal_processor>
2. <https://www.chipverify.com/>