



הפקולטה להנדסה ומדעי המחשב
החוג להנדסת חשמל ואלקטרוניקה

פרויקט גמר באלקטרוניקה

מימוש MD6 Hash של אלגוריתם FPGA FPGA Implementation of the MD6 Hash Algorithm

מגישים : אופק שרעבי, אביאל בירדוacker ט' טבת תשפ"ד
ירושלים
מנחה : מר אורן שטרו

תודות

ראשית, תודה לבורא עולם שיזכה אותנו להשלים את הפרויקט בהצלחה.

למנהל הפרויקט, מר אורי שטרו - על מתן הכוונה ותמיכה שלא יסולא בפז לאורך כל הפרויקט. אנחנו אסירים תודה על הסבלנות והמסירות הבלתי פוסקת שלך. העידוד שלך לאורך הפרויקט היה גורם מכריע בהשלמתו.

لد"ר דוד מרטין מרסלו, רכז הפרויקטדים – על שיתוף הפעולה לאורך כל הפרויקט ועל הבחינה וההתעניינות. אנחנו אסירים תודה.

לאלכס קלין על הקצת מחשב עצמאי והתקנת כל הדרייברים והתוכנות הנדרשות. תודה למרכז האקדמי לב, לראשי המחלקה בחוג אלקטרוניקה פרופ' שלמה אנגלברג ופרופ' בנימין מילגרום, לרכז החוג ד"ר יוסף גולבצ'יוב ובפרט למר עירן שלום וגברת ענת בן-חמו שננתנו בידינו את הכלים לעמוד במשימה בכבוד בפרויקט זה ובכלל בתואר כולם.

תקציר

אלגוריתם MD6 הוא משפחת פונקציות הגיבוב בתורת הצפנה. פונקציית גיבוב הידועה גם בשם פונקציה חד-כיוונית, מקבלת הודעה אקראית באורך אקראי ומיצרת הודעה מגובבת באורך קבוע. איזות פונקציית הגיבוב נבדדת, בין היתר, ע"פ רמת ה"עירבול" שלה. הוא אומר, נתוני הקלט יעברו פרוצדורה כזו שהייתה כמוה שפהות קשור בין הקלט לפולט ו疏散 כל הפלטים יתפזרו באופן שווה ככל האפשר על פני טווח הפלט. ניתן למצוא מגוון יישומים המבוססים על עקרונות הגיבוב, כגון הפסקת "טבעת אצבע" או "חתימה" באורך קבוע עבור מידע דיגיטלי באורך משתנה, או למשל הזנת סיסמה לצורך פתיחת נעילה. אותה סיסמה מומרת לקוד אשר משווה בתוך המחשב לצורך זיהוי הסיסמה שהזונה בתחילת. בכל הדוגמאות הללו אין שום מטרה לפענה בהזורה את המידע. אלגוריתם MD6 (פותח על ידי פרופסור Ron Rivest Ron Rivest מ-MIT, מומחה בעל עולמי בתחום הצפנה, וצוות מומחי הצפנה בראשותיו) הוגש בתחרות בינלאומית להגדלת אלגוריתם גיבוב מדור שלישי (SHA-3) ואת חלק מהמטרה לחזק את אלגוריתמי הגיבוב נגד פריצות מתחככות אשר עלולות הגיעו. במסגרת הפרויקט, האלגוריתם ממומש בחומרה ובתוכנה, לצורך השוואה והמחשת הייעולות של המימוש בחומרה.

אלגוריתם MD6 יש כמה מצבים שבהם הוא משתמש במספר משתנה של פונקציות דחיסה לצורך גיבוב המידע. אי לכך, השימוש בחומרה יכול להיות מאוד יעיל בשל העובדה שהיא יכולה לבצע דחיסה במקביל, או בשיביל להתאים בצורה ספציפית את אופן הפעולה של האלגוריתם המתאים לנו, וכמו כן מהירות החישוב בחומרה לעומת התוכנה.

במסגרת הפרויקט מומש אלגוריתם MD6 בשפת החומרה – Verilog. הפרויקט כלל את כתיבת האלגוריתם לפי מסמכיו המפתחים של האלגוריתם וכן בדיקה וIMPLEMENTATION על מערכת פיתוח הכללת רכיב FPGA. מומשה פונקציית דחיסה ראשית של MD6-3 ב�ורא Basys3 של חברת Digilent Inc. המבוסס על Xilinx Artix-7 FPGA. קצב עיבוד המימוש בחומרה יותר מהיר לעומת התוכנה.

ישום יעיל של פונקציית הדחיסה המורכבת של MD6 על פלטפורמת החומרה של לוח Basys3 והוספת GUI, הפיק תוצאות של מהירות חישוב יותר טובות מאשר המימוש בתוכנה.



Abstract

The MD6 algorithm is a member of the cryptographic hashing function family. A hash function, alternatively referred to as a one-way function, processes a randomly sized message and generates a fixed-length hashed message. The effectiveness of the hashing function is evaluated, in part, based on its "scramble" level. This refers to the process by which the input data undergoes a procedure to minimize the correlation between the input and output, ensuring that all outputs are evenly distributed across the output range. Variety of application based on hashing principles could be found, such as producing a fixed-length "fingerprint" or "signature" for variable-length digital data, or for instance, when entering a password for unlocking purposes. The password is converted into a code that is compared by the computer to identify the initially entered password. In all these scenarios, there is no need to decrypt the code. Developed by Professor Ron Rivest from MIT, a renowned expert in encryption, and by encryption experts led by him. The MD6 algorithm was submitted as part of an international competition to define the third-generation hashing algorithm (SHA-3). This competition aimed to strengthen hashing algorithms against sophisticated hacking techniques. As part of the project, the algorithm is implemented in hardware and software, for the purpose of comparing and illustrating the efficiency of the implementation in hardware.

The MD6 algorithm employs various modes of operation that utilize a variable number of compression functions to compress information. This flexibility enables efficient hardware utilization, as multiple compression functions can be computed simultaneously in hardware or tailored to specific operational requirements, enhancing computational speed, comparing to a software implementation.

As part of the project, the MD6 algorithm was implemented in Verilog, a hardware description language. The implementation involved following the algorithm's developer documentation, conducting testing, and deploying it on a development system with an FPGA component. The primary objective was to efficiently implement the MD6 hash algorithm's main compression function on the Basys3 FPGA board, which is based on the Xilinx Artix-7 FPGA. The processing rate of the hardware implementation is higher compared to the software implementation.

Effective implementation of the complex MD6 compression function on the hardware platform of the Basys3 board and adding a GUI, produced better calculation speed results than the software implementation.

תוכן עניינים

| | |
|----|---|
| 7 | רשימת איורים |
| 9 | רשימת טבלאות |
| 10 | 1 מבוא |
| 11 | 2 רקע תיאורטי |
| 11 | 2.1 מבוא לкриיפטוגרפיה [1] |
| 12 | 2.2 פונקציית גיבוב – [2] HASH |
| 13 | 2.3 אלגוריתם MD6 [3] |
| 13 | 2.3.1 הקלט והפלט של ה-MD6 |
| 14 | 2.3.2 קבועים ב-MD6 |
| 17 | 2.3.3 מצבים הפעולה של ה-MD6 |
| 19 | 2.3.4 פונקציית הדחיסה |
| 23 | 3 פיתוח ושיטות |
| 23 | 3.1 ערכת הפיתוח ורכיב ה-[8] FPGA |
| 24 | 3.2 מטפט להעברת נתונים |
| 25 | 3.3 מימוש האלגוריתם בשפת חומרה |
| 27 | 3.3.1 בлок ה-Receiver |
| 29 | 3.3.2 MD6 Mode |
| 37 | 3.3.3 Transmitter |
| 38 | 3.3.4 Button Debouncing |
| 39 | 3.4 הטמעה על רכיב ה-[8] FPGA |
| 39 | 3.4.1 Constraint Specification – מפרט אילוצים |
| 41 | 3.4.2 סינטזה – Synthesis |
| 43 | 3.4.3 Place & Route – מיקום וחיוון |
| 45 | 3.4.4 Bitstream Generation and Device Programming |
| 45 | 3.5 יצירת קוד תוכנה לקישור בין החומרה למשתמש |
| 47 | 4 תוצאות |
| 47 | 4.1 הפעלת אלגוריתם MD6 |
| 47 | 4.1.1 תהליך הטמעת אלגוריתם MD6 על רכיב ה-FPGA |
| 53 | 4.1.2 תהליך הפעלת גיבוב המידע על רכיב ה-FPGA |
| 60 | 4.2 אימות תכנון החומרה בסימולציה ModelSim |
| 60 | 4.2.1 תהליך אימות תכנון החומרה בסימולציה ModelSim |
| 61 | 4.2.2 הפעלת אימות תכנון החומרה בסימולציה ModelSim |
| 63 | 4.3 אימות תכנון החומרה על ערכת הפיתוח |
| 63 | 4.3.1 תהליך אימות תכנון החומרה על ערכת הפיתוח |



| | |
|----------|---|
| 64 | 4.3.2 הפעלת אימות תכנון החומרה על ערכת הפיתוח |
| 67 | 5 דיוונים |
| 67 | 5.1 השוואת ספרות |
| 68 | 5.1.1 השוואת תוצאות אימות תכנון החומרה בסימולציה ModelSim |
| 69 | 5.1.2 השוואת תוצאות אימות תכנון החומרה על ערכת הפיתוח |
| 71 | 5.2 השוואת לתוכנה |
| 74 | 6 מסקנות וסיכום |
| 75 | מקורות מידע ומאמרים |
| 76 | נספח א – קישורים למסמכים וקבצי הפרויקט |
| 77 | נספח ב – פירוט תוכן תיקיית הפרויקט |
| 78 | נספח ג – הגשה לתחרות |

רשימת אйורים

| | | |
|-------------|--|---------|
| איור 2.1 – | תקשרות בין אליס לבוב מותקפת על ידי איב [10]. | 11..... |
| איור 2.2 – | מצב הפעולה המקבילי [3]. | 17..... |
| איור 2.3 – | מצב הפעולה הטורי [3]. | 18..... |
| איור 2.4 – | מצב הפעולה ההיברידי [3]. | 18..... |
| איור 2.5 – | מצב הפעולה ההיברידי U [3]. | 19..... |
| איור 2.6 – | פרישת מזהה הצומת הייחודי U [3]. | 19..... |
| איור 2.7 – | פרישת מילת הקרה V [3]. | 20..... |
| איור 2.8 – | לולאת החישוב מוצגת כאוגר האזה לא לנארו [3]. | 20..... |
| איור 2.9 – | אופן הפעולה של לולאת החישוב [3]. | 21..... |
| איור 2.10 – | פונקציית הדחיסה f נראית כפעולת הצפנה ואחריה פעולה חיתוך [3]. | 22..... |
| איור 3.1 – | ערכת הפיתוח Basys-3 [11]. | 23..... |
| איור 3.2 – | מבנה שליחת המידע בתקשורת UART מסוג 8N1 [12]. | 24..... |
| איור 3.3 – | דיגרמת בלוקים של התכון כולל המעטפת. | 25..... |
| איור 3.4 – | דיגרמת הבלוקים של השימוש.....receiver. | 25..... |
| איור 3.5 – | סימבול בлок-h-.....receiver. | 27..... |
| איור 3.6 – | ה-sbus של תמי המידע. | 27..... |
| איור 3.7 – | סימבול בлок-h-.....MD6 Mode. | 29..... |
| איור 3.8 – | בלוק-h-.....MD6 Mode. | 29..... |
| איור 3.9 – | סימבול בлок-h-.....cf. | 30..... |
| איור 3.10 – | אחז רכיבי החומרה המשומשים בתכנון הראשוני של בлок-h-.....cf. | 30..... |
| איור 3.11 – | אחז רכיבי החומרה המשומשים בתכנון הסופי של בлок-h-.....cf. | 31..... |
| איור 3.12 – | בלוק-h-.....cf. | 31..... |
| איור 3.13 – | סימבול בлок-h-N. מבחו. | 33..... |
| איור 3.14 – | בלוק-h-N. | 33..... |
| איור 3.15 – | סימבול בлок-h-S. | 34..... |
| איור 3.16 – | גודל בлокי-h-rom האסינכרוני ברכיב-h-FPGA. | 34..... |
| איור 3.17 – | סימבול בлок-h-rom rshift lshift. | 35..... |
| איור 3.18 – | סימבול בлок-h-rom rshift lshift. | 35..... |
| איור 3.19 – | סימבול בлок-h-loop A computation. | 36..... |
| איור 3.20 – | סימבול בлок-h-transmitter. | 37..... |
| איור 3.21 – | סימבול בлок-h-Button debouncing. | 38..... |
| איור 3.22 – | בלוק-h-Button debouncing. | 38..... |
| איור 3.23 – | חיבור תקשורת-h-UART בערכת הפיתוח. | 39..... |
| איור 3.24 – | חיבור הנוריות והחלצנים לפינים בערכת הפיתוח. | 40..... |
| איור 3.25 – | מיקום הcptורים והנוריות על ערכת הפיתוח. | 40..... |
| איור 3.26 – | כלי החומרה המנוצלים לאחר סינטזה. | 41..... |
| איור 3.27 – | מיקום רכיבי החומרה המשומשים לאלגוריתם. | 43..... |
| איור 3.28 – | מפת הכניסות והיציאות של הרכיב. | 44..... |
| איור 3.29 – | כלי החומרה המנוצלים לאחר מיקום וחיווט. | 44..... |
| איור 3.30 – | ניתוח הזמןן של התכנון. | 44..... |
| איור 4.1 – | חלון הפתיחה של כלי התוכנה VIVADO. | 47..... |
| איור 4.2 – | Create a New Vivado Project. | 48..... |
| איור 4.3 – | חלון-h-Project Name. | 48..... |
| איור 4.4 – | חלון-h-Project Type. | 49..... |
| איור 4.5 – | Add Sources. | 49..... |

| | |
|---------|--|
| 50..... | .Add Constraints |
| 50..... | .Default Part |
| 51..... | .New Project Summary |
| 51..... | איור 4.9 – החלון הראשי של ניהול הפרויקט. |
| 52..... | .HARDWARE MANAGER |
| 52..... | איור 4.11 – חלון HARDWARE MANAGER לאחר החיבור לרכיב החומרה. |
| 53..... | איור 4.12 – חלון Program Device להטמעת התכנון. |
| 53..... | איור 4.13 – מציאת מספר-h-COM במנהל התתקנים. |
| 54..... | איור 4.14 – חלון ההתחלה של תוכנית הגיבוב. |
| 54..... | איור 4.15 – חלון INTRODUCTION (המבוא). |
| 55..... | איור 4.16 – חלון INSTRUCTION (ההוראות). |
| 55..... | איור 4.17 – חלון Definition questions (שאלות ההגדלה). |
| 56..... | איור 4.18 – חלון השגיאה על אי מילוי תאים. |
| 56..... | איור 4.19 – חלון Message (ההודעה). |
| 56..... | איור 4.20 – חלון השגיאה על הכנסת מידע לא תואם. |
| 57..... | איור 4.21 – חלון Key (המפתח) עם תיבת הכנסת המידע. |
| 57..... | איור 4.22 – חלון Key (המפתח) עם ערך ברירת המחדל. |
| 58..... | איור 4.23 – חלון ה-r & L .d. |
| 58..... | איור 4.24 – חלון number COM. |
| 59..... | איור 4.25 – חלון השגיאה על הכנסת מספר COM שגוי. |
| 59..... | איור 4.26 – חלון the hashed message (ההודעה המוגבהת). |
| 60..... | איור 4.27 – שליחת המידע האكريדי Rx-D. |
| 61..... | איור 4.28 – קבלת המידע המוגבב דרך ה-Dx. |
| 62..... | איור 4.29 – Change directory בכל התוכנה ModelSim. |
| 63..... | איור 4.30 – החלון הראשי של הסימולציה בכל התוכנה ModelSim. |
| 63..... | איור 4.31 – תוצאות הגיבוב ביחיד עם צורת הגלים של אותן הסופיות. |
| 64..... | איור 4.32 – הכנסת מספר כניסת-h-COM. |
| 65..... | איור 4.33 – הכנסת מספר וקיטורי הבדיקה. |
| 65..... | איור 4.34 – תצוגת הוקטור הראשון המוגבב בתוכנה. |
| 66..... | איור 4.35 – תצוגת וקטור החומרה שהתקבל. |
| 66..... | איור 4.36 – קובץ cs להשוואה בין התוצאות. |
| 67..... | איור 5.1 – מידע הקלט של הדוגמא הראשונה [3]. |
| 67..... | איור 5.2 – תוצאות הגיבוב של הדוגמא הראשונה [3]. |
| 68..... | איור 5.3 – תוצאות הגיבוב של הדוגמא הראשונה [3]. |
| 68..... | איור 5.4 – חלון צורות הגלים של הקלט והפלט של הדוגמא הראשונה. |
| 69..... | איור 5.5 – התאמת שאלות ההגדלה לדוגמא הראשונה. |
| 69..... | איור 5.6 – התאמת הודעה לדוגמא הראשונה. |
| 70..... | איור 5.7 – התאמת המפתח לדוגמא הראשונה. |
| 70..... | איור 5.8 – התאמת z & L & d לדוגמא הראשונה. |
| 71..... | איור 5.9 – תוצאות הגיבוב של הדוגמא הראשונה בחומרה. |

רשימת טבלאות

| | |
|---------|---|
| 14..... | טבלה 1: וקטורי הקבועים – Q |
| 15..... | טבלה 2: וקטורי הקבועים – S |
| 16..... | טבלה 3: וקטורי ההאה l>.t |
| 16..... | טבלה 4: קבועי עמדות ההקהה – t |
| 23..... | טבלה 5: מאפייני ערכת הפעיתות |
| 26..... | טבלה 6: הקלט והפלט של התכנון. |
| 26..... | טבלה 7: האותות הפנימיים של התכנון. |
| 27..... | טבלה 8: הקלט והפלט של בлок ה-receiver |
| 28..... | טבלה 9: האותות הפנימיים של בлок ה-receiver |
| 29..... | טבלה 10: הקלט והפלט של בлок ה-MD6 Mode |
| 30..... | טבלה 11: האותות הפנימיים של בлок ה-MD6 Mode |
| 32..... | טבלה 12: הקלט והפלט של בлок ה-cf |
| 32..... | טבלה 13: האותות הפנימיים של בлок ה-cf |
| 33..... | טבלה 14: הקלט והפלט של בлок ה-N |
| 33..... | טבלה 15: האותות הפנימיים של בлок N |
| 35..... | טבלה 16: הקלט והפלט של בлок ה-A iterative |
| 36..... | טבלה 17: הקלט והפלט של בлок ה-A computation loop |
| 36..... | טבלה 18: באוטות הפנימיים של בлок ה-A computation loop |
| 37..... | טבלה 19: הקלט והפלט של בлок ה-transmitter |
| 37..... | טבלה 20: האותות הפנימיים של בлок ה-transmitter |
| 38..... | טבלה 21: הקלט והפלט של בлок ה-Button debouncing |
| 38..... | טבלה 22: האותות הפנימיים של בлок ה-Button debouncing |
| 67..... | טבלה 23: המידע הנכנס לאלגוריתם בסימולציה |

1 מבוא

בעידן המסומן על ידי צמיחה בלתי פוסקת של נתונים דיגיטליים, אבטחת המידע ושלמותו הפכה לחשיבות עליונה. פונקציות גיבוב קרייפטוגרפיות ממלאות תפקיד מכריע בעניין זה, ומציאות אמצעי להגן על נתונים רגישים על ידי הפיכתם לערך בגודל קבוע. פונקציית hash MD6, הידועה בעמידותה בפני התקפות קרייפטוגרפיות שונות, הtgtalta כמעמדת ראוייה להבטחת שלמות הנתונים. פרויקט זה מתמקד בישום של MD6 הן בפלטפורמות החומרה והן בפלטפורמות התוכנה, במטרה לספק תובנות לגבי מהירות העיבוד והיעילות שלהן.

המטרה העיקרית של פרויקט זה היא להעיר את הביצועים של יישום ה-*hash* MD6 בחומרה לעומת תוכנה, ולגלוות את נקודות החזק והחולשה שלהם. על ידי הבנת הפרשנות בין שני היישומים הללו, אנו שואפים לתרום ידע רב ערך בתחום הצפנה ואבטחת המידע.

רכיבי FPGA מתוכנים לבצע עיבוד מקביל, המאפשרים חישובי גיבוב מרובים להתרחש בו-זמנית. בנגדוד לגיבוב מבוסס תוכנה, שבו פעולות מבוצעות ברצף של מעבד. רכיבי FPGA יכולים לעבוד נתחי נתונים מרובים במקביל. מקבילות זו מאייצה משמעותית את תהליכי הגיבוב.

התמקדות של הפרויקט בישום חומרה משמעותית במיוחד בהיחס למשאים המוגבלים בסביבות חומרה. ניתוח הביצועים של MD6 באופטימיזציה של ניצול המשאים, מאפשר פעולות קרייפטוגרפיות ייעילות יותר במכשורים בעלי יכולות עיבוד מוגבלות. ניתן לכוון יישומי חומרה כך שניצלו ביעילות רכיבים לוגיים זיכרון, תוך הבטחה שפעולות הגיבוב חסכנותיות למשאים.

המחקר ההשוואי בין יישומי חומרה ותוכנה הוא המפתח למתן הבנה מקיפה של הפעולות הכרוכות בכך. ניתוח זה לימד לנו היבטי הישום המתאים ביותר בהתבסס על מקרי שימוש ספציפיים ומוגבלות משאים. הניתוח ההשוואי בין יישומי חומרה ותוכנה משמש מדריך למפתחים וארכיטקטוי מערכות בבחירה הגישה המתאימה ביותר בהתבסס על הדרישות הספציפיות שלהם. הדרך זו חיונית לפיתוח מערכות מאובטחות ויעילות בתחוםים שונים.

התובנות של הפרויקט לגבי ביצועי MD6 מביאות לשיפור האבטחה בפרוטוקולי תקשורת. היכולת לבחור בין יישומי חומרה ותוכנה בהתבסס על החווקות שלהם תורמת לפיתוח ערכוי תקשורת מאובטחים יותר, ומהזקפת את האמון באינטראקציות דיגיטליות. בפרט בחומרה ניתן לשלב מודלי אבטחת חומרה (HSM) ואלמנטים מאובטחים עם FPGA כדי להבטיח שפעולות גיבוב מבוצעות בסביבה מאובטחת, תוך הגנה על נתונים רגישים מפני התקפות פיזיות.

פרויקט זה מלא תפקיד מרכזי בקידום ידע קרייפטוגרפי, אופטימיזציה של ניצול המשאים ומתן הדרישה מעשית לעיבוד נתונים מאובטח. ההטמעה והනיתוח ההשוואי של MD6 בפלטפורמות חומרה ותוכנה תורמים הן להבנה המדעית של פונקציות ה-*hash* הצפנה והן לישומים המעשיים שלהן, במטרה סופית לשפר את אבטחת המידע בעולם יותר ויותר דיגיטלי.

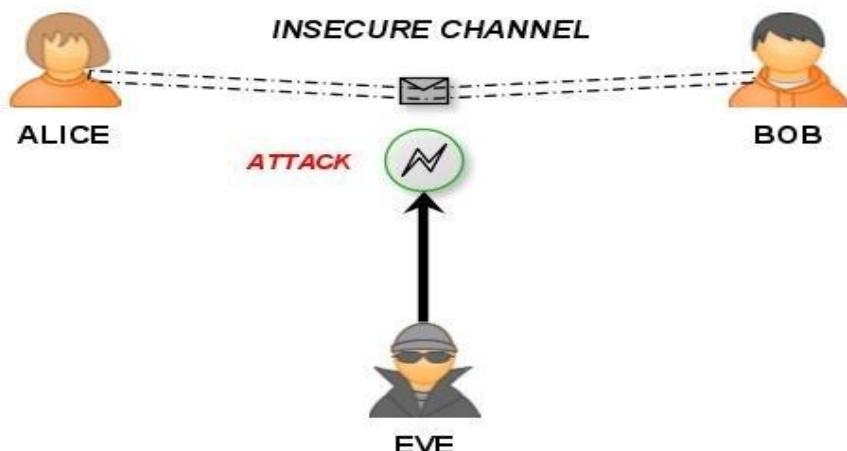
2 רקע תיאורטי

2.1 מבוא לקריפטוגרפיה [1]

קריפטוגרפיה היא תחום של פיתוח טכניקות לתקשורת מאובטחת בנסיבות צדדים שלישים הנקרים יריבים. הוא עוסק בפיתוח וניתוח פרוטוקולים המונעים מצדים שלישיים זדוניים לאחזר מידע המשותף בין שני גורמים ובכך לעקוב אחר היבטים השונים של אבטחת מידע. תקשורת מאובטחת מתייחסת לתרחיש שבו יריב לא יכול לגשת להודעה או נתונים המשותפים בין שני הצדדים. בקריפטוגרפיה, יריב הוא ישות זדונית, שמטרתה לאחזר מידע או נתונים יקרים ובכך לעורר את עקרונות אבטחת המידע. סודיות נתונים, שלמות נתונים, אימות ואי-הכחשה הם עקרונות הליבה של הצפנה המודרנית.

- **סודיות:** מתייחסת לכללים והנחות מסוימים המבוצעים בדרך כלל במסגרת הסכמי סודיות המבטיחים שהמידע מוגבל לאנשים או למקום מסוימים.
- **שלמות:** הנתונים מתייחסת לשמירה והבטחה שהנתונים ישארו מדויקים ועקביים לאור כל מחזור החיים שלו.
- **אימות:** הוא תהליך לוודא שפיסט הנתונים שנتابע על ידי המשתמש שייכת אליו.
- **אי-הכחשה:** מתייחסת ליכולת לוודא שאדם או צד הקשורים לחוצה או תקשורת אינם יכולים להכחיש את האותנטיות של חתימתם על המסמך שלהם או על שליחת הודעה.

נראה לדוגמא 2 משתתפים לאחת קוראים אליס (השלחת) ולשני בוב (המקבל). כעת אליס רוצה לשלוח הודעה לבוב דרך ערוץ מאובטח. התהילה הוא כדלקמן. הודעה השולח, או לעיתים נקראת Plaintext, מומרת לצורה בלתי ניתנת לקריאה באמצעות מפתח - k. הטקסט המתתקבל נקרא ה-Ciphertext. תהיליך זה ידוע בשם הצפנה. בזמן הקבלה, ה-Ciphertext מומר בחזרה לטקסט פשוט באמצעות אותו מפתח - k, כך שניתן לקרואו אותו על ידי המקלט. תהיליך זה ידוע בשם פענות.



איור 2.1 – תקשורת בין אליס לבוב מותקפת על ידי אייב [10].

ישנם מספר סוגים של קריפטוגרפיה, כל סוג עם תוכנות ויישומים ייחודיים משלו. חלק מהסוגים הנפוצים ביותר של קריפטוגרפיה כוללים:



- הצפנה סימטרית:** סוג זה של קריפטוגרפיה כולל שימוש במפתח יחיד להצפנה ופענוח הנתונים. גם השולח וגם המתקבל משתמשים באותו מפתח, אותו יש לשומר בסוד כדי לשמר על אבטחת התקשרות.
- הצפנה אסימטרית:** המכונה גם קריפטוגרפיה של מפתח ציבורי, משתמשת בזוג מפתחות מפתח ציבורי ומפתח פרטי כדי להצפין ולפענח נתונים. המפתח הציבורי זמין לכל אחד, בעוד המפתח הפרטי נשמר בסוד על ידי הבעלים.
- פונקציות גיבוב - Hash:** פונקציה גיבוב היא אלגוריתם מתמטי הממיר נתונים בכל גודל לפلت בגודל קבוע. לעיתים קרובות נעשה שימוש בפונקציות גיבוב כדי לאמת את שלמות הנתונים ולהבטיח שלא טיפלו בהם. אלגוריתם MD5 נמצא תחת קטגורית פונקציות גיבוב קריפטוגרפית, וכן נרצה להרחיב על פונקציה הגיבוב בפרק הבא.

2.2 פונקציה גיבוב – HASH [2]

פונקציה גיבוב היא פונקציה חד-כיוונית הממיר הקלט באורך כלשהו לפلت באורך קבוע וידוע מראש. פונקציה גיבוב מתוכננת כך שכל שינוי בקלט יגרום לשינוי משמעותי בפלט. בדרך זו ניתן להתחמך עם עליית הבטחת שלמות מסרים גדולים, על ידי השוואת הערך המגויב שלהם במקום השוואתם ישירות. בשל היזוות קטנה משמעותית, קל יותר להגן על הערך המגויב מאשר על המסר המקורי.

פונקציות גיבוב קריפטוגרפיות הן מבני הבסיס של ההצפנה המודרנית ומשמשות כחתימות דיגיטליות, קודי אimoto, שמירת סיסמות ומחולל מספרים פסידו-אקראים. בישומים שאינם קריפטוגרפים הן משמשות לעיתים כמוזהה ייחודי של קובץ לצורך בדיקת שלמותו או נכונותו וכן להיות קבצים זהים.

פונקציה גיבוב קריפטוגרפית בטוחה מקיימת את התנאים הבאים:

- Pre-Image Resistance:** בהינתן פلت של פונקציה גיבוב, תהליך מציאת ערך הקלט המתאים הינו קשה.

מאפיין זה מגן מפני תוקף שמחזק בערך ה-hash וברצונו למצוא את הקלט המתאים.

- Second Pre-Image Resistance:** בהינתן קלט כלשהו, קשה למצוא קלט אחר המוביל לאותו פلت של פונקציה הגיבוב.

במילים אחרות, אם פונקציה גיבוב A עבר קלט x מיצרת ערך גיבוב (x)A, קשה למצוא כל קלט אחר y כך ש-(x)A = (y)A .

מאפיין זה של פונקציה גיבוב מגן מפני תוקף שמחזק בקלט כלשהו ובגיבוב המתאים לו וברצונו להחליף קלט אחר כערך לגיטימי במקום ערך הקלט המקורי.

:Collision Resistance

- קשה מבחינה חשובה למצוא שני קלטים שונים שיבילו אותו פلت של פונקציית הגיבוב.

במילים אחרות, עבור פונקציית גיבוב A קשה למצוא x ו- y כך ש- $(y) = H(x)$ כאשר $y \neq x$.

ראו לציין כי פונקציית גיבוב "דוחשת" מידע באורך מסוים למידע מגובב באורך קטן יותר. אי-כך, לא יתכן שלפונקציית גיבוב לא יהיה התנשויות. מאפיין זה רק מסב את העובדה שקשה למצוא התנשויות אלו.

מאפיין זה מקשה מאוד על מציאת שני ערכי קלט המובילים אליו גיבוב. כמו כן, במידה ופונקציית הגיבוב מקיימת מאפיין זה, היא תקיים גם את שני המאפיינים הקודמים.

2.3 אלגוריתם MD6 [3]

אלגוריתם MD6 הוא פונקציית גיבוב קריפטוגרפית שפותחה על ידי Ron Rivest מהמכון הטכנולוגי של מסצ'וסטס (MIT) וצוות מומחי הצפנה בראשותו. אלגוריתם MD6 הוגש כאחת מהצעות האלגוריתמים לתחום תקן הגיבוב-3 SHA-3 של NIST [4].

האלגוריתם פועל באמצעות מבנה Merkle-Damgård [5], כלומר הוא מעבד נתוני קלט בבלוקים ודווחס כל בלוק לפلت בגודל קבוע. אלגוריתם MD6 כולל גודל בלוק גמיש, המאפשר לו לעבד ביעילות נתונים גדולים משתנים. הוא כולל גם מספר תכונות חדשות, כגון פונקציות דחיסה מקבילה ואלגוריתם עדכון הودעות מבוסס עץBINARI. כל מילה באלגוריתם מוגדרת כ-64 סיביות.

2.3.1 הקלט והפלט של ה-MD6

הקלט ל-MD6 הן כדלקמן:

- M - ההודעה המיועדת להתגבב (חוובה).
- d - אורך ההודעה המוגובבת בסיביות (חוובה).
- K - ערך מפתח (אופציונלי).
- L - בקרת מצב (אופציונלי).
- r - מספר סיבובים (אופציונלי).

כニיסות החובה היחידות הן ההודעה M שיש לגיבוב ואורך ההודעה המוגובבת d. לכנייסות האופציונליות יש ערכי ברירת מחדל אם לא מסופק ערך כלשהו.

הפלט של MD6 הוא A - מחראת סיביות באורך של d סיביות.

2.3.1.1 ההודעה – M

הקלט הראשון ל-MD6 הוא ההודעה M אותה משתמש רוצה לגיבב, אורך ההודעה m צריך להיות בגודל $2^{64} < m \leq 0$ סיביות.

2.3.1.2 אורך ההודעה המוגובבת – d

הקלט השני ל-MD6 הוא אורך הסיביות d של ההודעה המוגובבת אשר ערכיו יכולים לנوع בין $512 \leq d < 0$.



ד חיב להיות ידוע בתחום חישוב הגיבוב, מכיוון שהוא לא רק קובע את אורך הפלט ה- MD6 הסופי, אלא גם משפייע על חישוב ה- MD6 בכל פעולות ביןיהם אורכי ה-p כפי שנדרש מ-3 SHA המ: .224, 256, 384, 512

2.3.1.3 המפתח – K

הקלט הבא ל- MD6 הוא המפתח K המשמש להוספת אבטחה על ההודעה הנשלחת, אורך המפתח k צריך להיות בגודל $k < 512$ סיביות. המפתח הוא K = 0 ואורךו k = 0 סיביות.

2.3.1.4 פרמטר בקרת המצב – L

הקלט הבא ל- MD6 הוא פרמטר בקרת המצב L, המשמש לבחירת אחד מצבי הפעולה של ה- MD6 אשר יפורטו להלן [פרק 2.3.3](#). אורך פרמטר בקרת המצב L הוא $L \leq 64$.

פרמטר בקרת המצב L הוא אופציוני, ברירת המחדל כאשר משתמש בוחר לא להכניס את הקלט הוא $L = 64$.

2.3.1.5 מספר הסיבובים – r

הקלט הבא ל- MD6 הוא מספר הסיבובים r, המשמש למספר הסיבובים של הפעלת פונקציית הדחיסה של ה- MD6 כמפורט להלן [פרק 2.3.4](#). אורך מספר הסיבובים הוא $r < 168$.

מספר הסיבובים הוא אופציוני, ברירת המחדל כאשר המשתמש בוחר לא להכניס את הקלט הוא $r = d/4 + 40$.

2.3.2 קבועים ב- MD6

באלגוריתם MD6 יש מספר קבועים אשר משתמשים בהם בחישוב פונקציית הדחיסה:

- וקטורי הקבועים – Q
- וקטורי הקבועים – S
- וקטורי קבועי הazea (shift) – r&l
- קבועי עמדות ההקשה (tap position) – t

2.3.2.1 וקטורי הקבועים Q

וקטור הקבועים Q זהו וקטור אשר מכיל 15 מילימ של 64 סיביות. הוקטור משמש כחלק מבlok הנתונים אשר נכנס לכל פונקציית דחיסה באlgorigthm MD6 כמפורט לעיל [פרק 2.3.4](#).

ערci ה-Q מוצגים בטבלה 1.

טבלה 1: וקטורי הקבועים – Q.

| | | | | | |
|-----------|--------------------|-----------|--------------------|-----------|--------------------|
| 0 | 0x7311c2812425cfa0 | 1 | 0x6432286434aac8e7 | 2 | 0xb60450e9ef68b7c1 |
| 3 | 0xe8fb23908d9f06f1 | 4 | 0xdd2e76cba691e5bf | 5 | 0xcd0d63b2c30bc41 |
| 6 | 0x1f8ccf6823058f8a | 7 | 0x54e5ed5b88e3775d | 8 | 0x4ad12aae0a6d6031 |
| 9 | 0x3e7f16bb88222e0d | 10 | 0x8af8671d3fb50c2c | 11 | 0x995ad1178bd25c31 |
| 12 | 0xc878c1dd04c4b633 | 13 | 0x3b72066c7a1552ac | 14 | 0xd6f3522631effcb |

2.3.2.2 וקטור הקבועים S

וקטור הקבועים S זהו וקטור אשר מכיל 168 מילימ של 64 סיביות. הוקטור משמש כחלק מחישוב פונקציה הדחיסה כאשר בכל סיבוב משתמש במילה אחרת ב-S לפי מספר הסיבוב כמו술ר לעיל [פרק 2.3.4](#).

ערכי ה-S מוצגים בטבלה.

טבלה 2: וקטורי הקבועים – S.

| | | | | | |
|------------|---------------------|------------|--------------------|------------|--------------------|
| 0 | 0x0123456789abcdef | 1 | 0x0347cace1376567e | 2 | 0x058e571c26c8eadc |
| 3 | 0xa0a1cec3869911f38 | 4 | 0x16291870f3233150 | 5 | 0x3e5330e1c66763a0 |
| 6 | 0x4eb7614288eb84e0 | 7 | 0xdf7f828511f68d60 | 8 | 0xedee878b23c997e1 |
| 9 | 0xbadd8d976792a863 | 10 | 0x47aa9bafeb25d8e7 | 11 | 0xcc55b5def66e796e |
| 12 | 0xd8baeb3dc8f8bbfd | 13 | 0xe165147a91d1fc5b | 14 | 0xa3cb28f523a234b7 |
| 15 | 0x6497516b67646dcf | 16 | 0xa93fe2d7eaec961e | 17 | 0x736e072ef5fdaa3d |
| 18 | 0x95dc0c5dcfde5a | 19 | 0x3aa818ba9bb972b5 | 20 | 0x475031f53753a7ca |
| 21 | 0xcdb0636b4aa6c814 | 22 | 0xda7084d795695829 | 23 | 0xe6f1892e2ef3f873 |
| 24 | 0xaff2925c79c638c7 | 25 | 0x7cf5a6b8d388790f | 26 | 0x89facff1a710bb1e |
| 27 | 0x12e55d626a21fd3d | 28 | 0x37cbfac4f462375a | 29 | 0x5c963709cce469b4 |
| 30 | 0xe93c6c129dec9ac8 | 31 | 0xb36898253ffdbf11 | 32 | 0x55d1b04b5bdef123 |
| 33 | 0xfb2e097b7b92366 | 34 | 0xfb2e097b7b92366 | 35 | 0x0dfb03dc96a7ce7b |
| 36 | 0x1ae70539296a52d6 | 37 | 0x27cf0a7372f4e72c | 38 | 0x6c9f16e7c5cd0978 |
| 39 | 0xb92f2f4e8f9f1bd0 | 40 | 0x435f5c9d1b3b3c21 | 41 | 0xc5aff9bb36577462 |
| 42 | 0xca5e33f748abace5 | 43 | 0xd6ac656f9176d56b | 44 | 0xff588ade22c96ff7 |
| 45 | 0x8da1973c6593904f | 46 | 0x1a42ac78ef26a09f | 47 | 0x2685d8f1fa69c1be |
| 48 | 0x6f0a7162d4f242dc | 49 | 0xbd14a2c5adc4c738 | 50 | 0x4b39c70a7f8d4951 |
| 51 | 0xd5624c14db1fdb2 | 52 | 0xfc4d829b63a7ce5 | 53 | 0x848970524854b56b |
| 54 | 0x0913a0a490adeff7 | 55 | 0x1336c1c9217e104e | 56 | 0x357d431362d8209c |
| 57 | 0x5becb427e5b041b8 | 58 | 0xe4d6484eef40c2d0 | 59 | 0xa9bcd09dfa814721 |
| 60 | 0x726961bad503c963 | 61 | 0x96d383f5ae065be6 | 62 | 0x3fb6856a7808fc6d |
| 63 | 0x4c7d8ad4d01134fa | 64 | 0xd8ea9729a0236d54 | 65 | 0xe1d5ac52606797a9 |
| 66 | 0xa2bad8a4e0eaa8f3 | 67 | 0xa2bad8a4e0eaa8f3 | 68 | 0xa2bad8a4e0eaa8f3 |
| 69 | 0x7a96c425e798bc9d | 70 | 0x7a96c425e798bc9d | 71 | 0x0d6bd095f6422ed5 |
| 72 | 0x1bd661aac884532a | 73 | 0x24bc83d5910ce574 | 74 | 0x6969852a221d0fc8 |
| 75 | 0xb3d28a54643f1010 | 76 | 0x54b596a8ec5b2021 | 77 | 0x83e5dd22dd4bc0e5 |
| 78 | 0x83e5dd22dd4bc0e5 | 79 | 0x04ca7a45be96416b | 80 | 0x0994b68a5928c3f6 |
| 81 | 0x1239ef94b271444c | 82 | 0x36621da944c3cc98 | 83 | 0x5ec43bd38d8655b0 |
| 84 | 0xef8875261f08eec0 | 85 | 0xbc10aa4c3a111301 | 86 | 0x4831d69854232503 |
| 87 | 0xd0726fb0ac674f06 | 88 | 0xf0f49de17cebd10d | 89 | 0x91f9bb43ddf6631b |
| 90 | 0x32e2f486bfc88537 | 91 | 0x57c5298d5b918f4e | 92 | 0xfc8b539bb722919c |
| 93 | 0x8917e5b64a65a2b9 | 94 | 0x133e0bec94eec7d3 | 95 | 0x356c15592df94826 |
| 96 | 0x5bd82ab37fd3d86c | 97 | 0xe4a057e7dba678f8 | 98 | 0xa940ed4eb768b951 |
| 99 | 0x73811a9d4af1fba3 | 100 | 0x940337bb95c23ce6 | 101 | 0x38076df62f84756d |
| 102 | 0x400f9b6c7b0caffa | 103 | 0xc01eb4d8d61dd054 | 104 | 0xc02de931a83e60a9 |
| 105 | 0xc05a1262705881f3 | 106 | 0xc0a426c4c0b18247 | 107 | 0xc1484f098142868f |
| 108 | 0xc390dc1202858b9f | 109 | 0xc4317824050e9cbf | 110 | 0xc873b0480e19b5df |
| 111 | 0xd0f6e0901832ee3f | 112 | 0xf1fd01a03045125f | 113 | 0x92eb03c0408f26bf |
| 114 | 0x37d70500811b4bdf | 115 | 0x5cbf0a010237dc3e | 116 | 0xe96f1603044a745c |
| 117 | 0xb3df2e070c94acb9 | 118 | 0x54af5e0f1d2dd5d3 | 119 | 0xf95ffe1f3e7e6e26 |

| | | | | | |
|-----|--------------------|-----|--------------------|-----|--------------------|
| 120 | 0x83ae3e3f58d8926d | 121 | 0x045c7e7fb1b1a6fb | 122 | 0x08a8bef4342cb56 |
| 123 | 0x1151ff7c86855dac | 124 | 0x33b23cf9090ff6f8 | 125 | 0x54747973121a2b50 |
| 126 | 0xf8f8b2e724345da0 | 127 | 0x81e1e74f6c4cf6e1 | 128 | 0x02c20c9ffc9d2b63 |
| 129 | 0x078419bedd3f5de6 | 130 | 0x0c0833fdb5bf66c | 131 | 0x1810657a58b62af8 |
| 132 | 0x20308af4b1485f50 | 133 | 0x607197694290f1a0 | 134 | 0xa0f2acd3852122e0 |
| 135 | 0x61f5d9260e634761 | 136 | 0xa2fa724c18e7c9e2 | 137 | 0x67e4a69831ea5a65 |
| 138 | 0xac9cfb043f4feeaa | 139 | 0x79925de087cd3375 | 140 | 0x8234fb410b9f65ca |
| 141 | 0x06793483173b8e15 | 142 | 0x0ee369872a56922a | 143 | 0x1fc7938f74a9a674 |
| 144 | 0x2c8ea59fcd72cac8 | 145 | 0x791dcbbe9ec55f10 | 146 | 0x832a55fd398ff120 |
| 147 | 0x0554eb7b531a2361 | 148 | 0x0bb914f7a63445e2 | 149 | 0x1463296e684cce64 |
| 150 | 0x38c752dcf09d52e8 | 151 | 0x418fe739c13fe770 | 152 | 0xc21e0c72825a09c0 |
| 153 | 0xc62c18e504b41a01 | 154 | 0xce58314b0d4c3e03 | 155 | 0xdea062971e9c7207 |
| 156 | 0xef4087af393ca60f | 157 | 0xbd818ddf525dca1f | 158 | 0x4a029b3fa4be5e3f |
| 159 | 0xd605b47e6d58f25e | 160 | 0xfe0ae8fcfeb126bd | 161 | 0x8e151179d9434bdb |
| 162 | 0x1e3b22f2b287dc37 | 163 | 0x2e674765450a744e | 164 | 0x7ecfcccb8e14ac9c |
| 165 | 0x8f9e5916182dd5b8 | 166 | 0x1c2cf22c307e6ed1 | 167 | 0x2859265840d89322 |

2.3.2.3 וקטורי הزاזה $r \& z$

וקטורי הקבועים $r \& z$ הם וקטורי הزاזה r (ימין) ו- z (לשמאלי) אשר כל אחד מכיל 16 מספרים הקסדיים, הווקטורים משמשים לחלק מחישוב פונקציית הדחיסה, כאשר בכל סיבוב מחשבים 16 מילים חדשות כמוסבר להלן בפרק [2.3.4](#) לכל מילה משתמשים בערך אחר של הزاזה לפי המיקום בוקטור.

ערכי ברירת המחדל מוצגים בטבלה 3.

טבלה 3: וקטורי הزاזה $r \& z$.

| $(i - n) \% 16$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| r_{i-n} | 10 | 5 | 13 | 10 | 11 | 12 | 2 | 7 | 14 | 15 | 7 | 13 | 11 | 7 | 6 | 12 |
| l_{i-n} | 11 | 24 | 9 | 16 | 15 | 9 | 27 | 15 | 6 | 2 | 19 | 8 | 15 | 5 | 31 | 9 |

2.3.2.4 קבועי עמדות ההקשה t

קבועי עמדות ההקשה t משמשות חלק מחישוב פונקציית הדחיסה, כאשר הקבועים בוחרים על אלו מילים מתוך בלוק הנתונים פונקציית הדחיסה תעשה את פעולות חישוב.

קבועי עמדות ההקשה t צריכים להיות בתחום $n = 89 < t_{0-4} < c = 16$.

ערכי ברירת המחדל מוצגים בטבלה 4.

טבלה 4: קבועי עמדות ההקשה $-t$.

| t_0 | t_1 | t_2 | t_3 | t_4 |
|-------|-------|-------|-------|-------|
| 17 | 18 | 21 | 31 | 67 |

2.3.3 מצב הפעולה של MD6

באלגוריתם MD6 יש 3 מצבים פעולה:

- 1) מקבילי – עץ היררכי
- 2) טורי – מבוסס על מבנה Merkle-Damgård
- 3) היברידי – שילוב של מקבילי וטורי

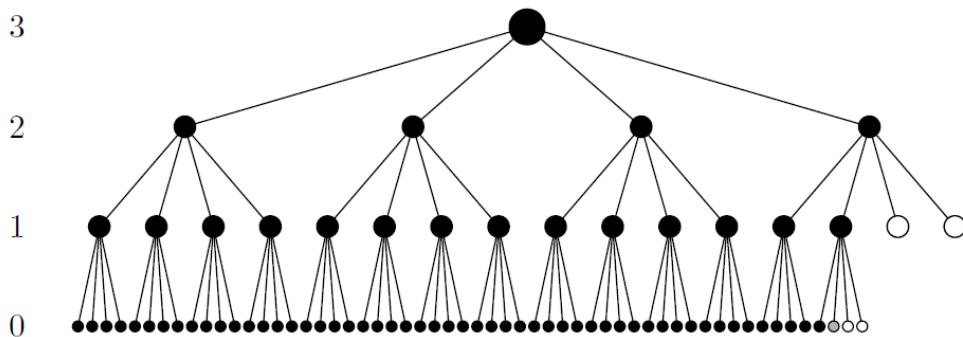
בחירת אופן הפעולה של האלגוריתם מתאפשר לפי ערך הפרמטר בקרת המצב – L.

- **מצב פעולה מקבילי:**

במצב מקבילי אשר גם קרוי מצב הפעולה הסטנדרטי, האלגוריתם מחלק את המידע המתתקבל לבLOCKים בגודל 16 מילימ' - קלומר 1024 סיביות בכלוק המידע זה נקרא בлок B, וכל 4 בלוקים נכנים לפונקציית דחיסה אחת (נראה בפרק 2.3.4) אשר הפלט שלה זהו בכלוק בגודל 16 מילימ', זאת אומרת שמספר הבלוקים מחלק בכל רמה ב-4. לכן מספר הרמות המקסימלי האפשרי הוא $27 = \left(\frac{2^{64}}{1024}\right) \log_4$.

ערך בחירת המחדל של פרמטר בקרת המצב הוא $L = 1$ אשר מבטיח שאופן הפעולה יהיה בצורה המקבילה שהוא אופן הפעולה הסטנדרטי.

level



איור 2.2 – מצב הפעולה המקבילי [3].

נסתכל באյור 2.2 – הרמה התחלה 0 היא הרמה התחתונה כאשר כל עיגול הוא בлок של 16 מילימ', העיגול האפור מסמן שהבלוק מכיל את סוף ההודעה שנשארה אשר קטן מ-16 מילימ' ולכן הוא מרופד באפסים, והעיגולים הלבנים מסמנים את הבלוקים המלאים באפסים.

כל קבוצה של 4 בלוקים מרמה 0 נכנים לפונקציית דחיסה ברמה 1 אשר מוציאה כפלט 16 מילימ', והם העיגולים השחורים המופיעים ברמה 2 וכן הלאה בכל הרמות, במקרה זה אפשר לראות שנשאר בлок אחד של נתונים ברמה 3 וכן זויה הרמה الأخيرة.

- **מצב פעולה טורי:**

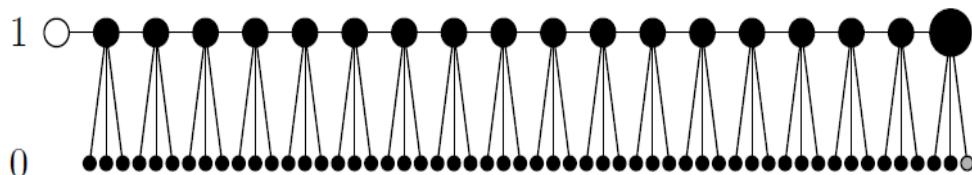
במצב פעולה טורי, האלגוריתם כמו במצב הפעולה המקבילי מחלק את ההודעה לבLOCKים של 16 מילימ', אבל לעומת מצב הפעולה המקבילי, מצב הפעולה הטורי עובד עם שתי רמות בלבד.

כאשר בرمה 0 מופיעים כל בלוקי המידע המחולקים ל-16 מילימ', וברמה 1 מופיעים כל פונקציות הדחיסה הנוצרות לצורך הגיבוב ובקרה הcy שמאלית וקטור איתחול של 16 מילימ' של אפסים – "initialization vector" או בקיצור IV.

לפונקציית הדחיסה הראשונה נכנסים 3 בלוקים מרמת ה-0 (הcy שמאלית) ובלוק האפסים מרמה 1.

לפונקציית הדחיסה השנייה נכנסים 3 הבלוקים הבאים מרמת ה-0 והפלט מפונקציית הדחיסה הראשונה וכן הלאה כמפורט באירור 2.3 .

בשביל להשתמש במצב הטורי ערך פרמטר בקרת המצב צריך להיות $0 = L$.
level



איור 2.3 – מצב הפעולה הטורי [3].

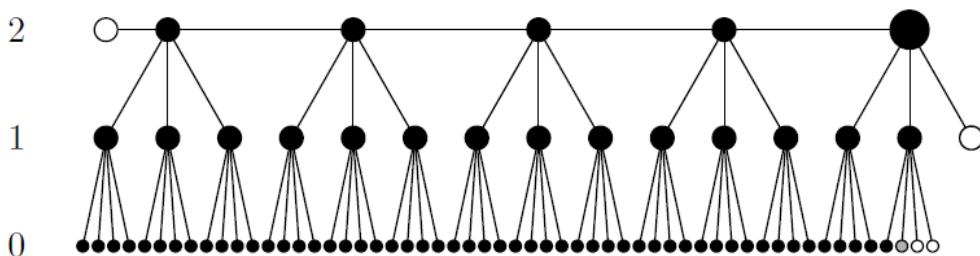
מצב פעולה היברידי:

- מצב הפעולה ההיברידי אחד בין מצב הפעולה המקבילי והטורי. במידה ומצד אחד, ערך ה-L שנקבע גדול מ-0, אך מצד שני, קטן מס' הדרגות הנוצר לחישוב מקבילי מלא, מצב הפעולה יהיה היברידי.

בשלב הראשון, האלגוריתם יתנהג באופן מקבילי לפי מס' הדרגות שנקבע.
בשלב השני, מצב הפעולה יהפוך לטורי, עד לקבלת תוצאה הגיבוב.

לשם ההמחשה, במידה $0 = 1 = L$, ובמידה ואורך המידע גדול מספיק, המשמעות היא שברמה מס' 1, מצב הפעולה הינו מקבילי וברמה מס' 2, מצב הפעולה הינו טורי עד לסוף החישוב וקבלת הגיבוב הרצוי. ניתן לראות זאת כמפורט באירור 2.4.

level



איור 2.4 – מצב הפעולה היברידי [3].

2.3.4 פונקציית הדחיסה

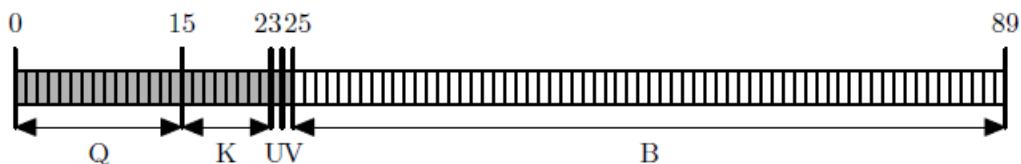
פונקציית הדחיסה, היא פונקציית החישוב המרכזית של אלגוריתם MD6, כל פונקציה דחיסה מקבלת 64 מילימטרים של 64 סיביות שהם 4096 סיביות של הودעה, ומוסיפה גיבוב של 16 מילימטרים של 64 סיביות שהם 1024 סיביות מידע. בפרק זה נפרט על מבנה הפונקציה ואופן פעולתה.

2.3.4.1 מבנה פונקציית הדחיסה

קלט פונקציה הדחיסה הוא בלוק מידע B בעל 64 מילימטרים, והוא נתמך בתוך בלוק נתונים A בעל 89 מילימטרים אשר מורכב מכמה חלקים כמו צבאיור 2.5.

- Q – וקטור הקבועים באורך 15 מילימטרים
- K – המפתח באורך 8 מילימטרים
- U – מזהה הצומת הייחודי באורך מילה אחת
- V – מילת הבקרה האורכת מילה אחת
- B – בלוק המידע בגודל 64 מילימטרים

על הכניסות B, K, Q פירטנו בפרקם לעיל, נפרט על הכניסות U, V.



איור 2.5 – מצב הפעולה ההיברידי [3].

2.3.4.1.1 מזהה הצומת הייחודי U

תפקידו של מזהה הצומת הייחודי U להוסיף לחישוב של כל פונקציה דחיסה את המיקום שלו במצב הפעולה באלגוריתם ע"י index ו-level, level, כאשר index זה מספר השלב ו-level זה המיקום באותו שלב או במילימטרים אחרים הערך הסידורי של פונקציית הדחיסה הנוכחיית, 7 הבתים הראשונים של ט מכילים את ערך ה-index וה-byte index שנשאר מכיל את ערך ה-level, כמו צבאיור 2.6.



איור 2.6 – פריסת מזהה הצומת הייחודי U [3].

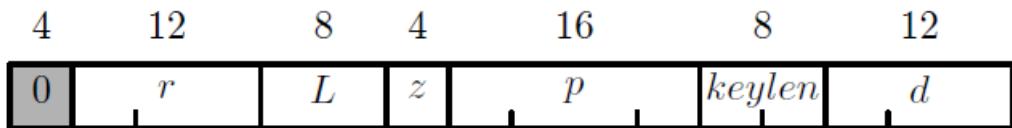
2.3.4.1.2 מילת הבקרה V

밀ת הבקרה V מורכבת מ-6 חלקים שונים כמפורט באיור 2.7.

נפרט:

- d – אורך ההודעה המוגבהת, באורך 12 סיביות
- Keylen – אורך המפתח K בבתים, באורך של 8 סיביות
- p – מספר סיביות הריפוד של בלוק המידע B, באורך של 16 סיביות
- z – שווה 1 כאשר מדובר בפונקציית הדחיסה الأخيرة אחרת שווה ל-0, באורך 4 סיביות

- ל – פרמטר בקרט המצב, באורך של 8 סיביות
- ר – מספר הסיבובים, באורך של 12 סיביות
- 4 סיביות של אפסים



איור 2.7 – פריסת מילת הבקשה [\[3\]](#).

2.3.4.2 פועלת החישוב של פונקציית הדחיסה

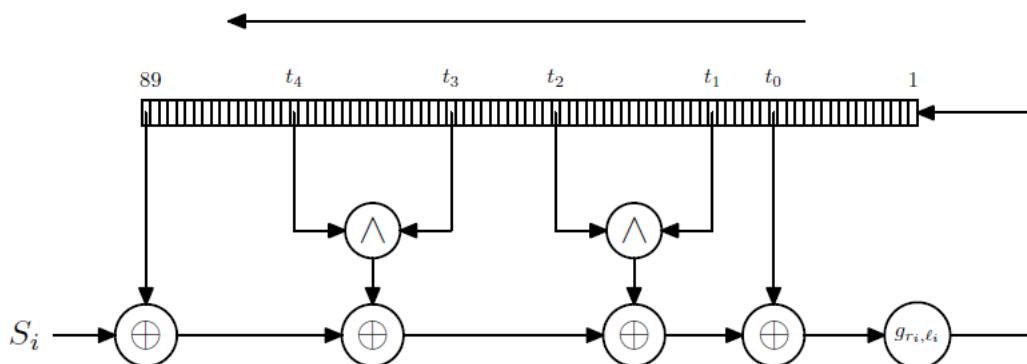
כמו שהוסבר בפרק הקודם, פונקציית הדחיסה מעבדת את בלוק הנתונים A, ומפעילה את לולאת החישוב המוצגת [במשואה 1](#).

(1)

$n = 89, t = 16$

```
for j in range(0, r)
    for i in range(n + j * t, n + (j + 1) * t)
        {
             $x = S_j \oplus A_{i-n} \oplus A_{i-t_0}$ 
             $x = x \oplus (A_{i-t_1} \& A_{i-t_2}) \oplus (A_{i-t_3} \& A_{i-t_4})$ 
             $x = x \oplus (x \gg r_{(i-n)\%16})$ 
             $A_i = x \oplus (x \ll l_{(i-n)\%16})$ 
        }
    }
```

אפשר להציג את לולאת החישוב כאוגר הואה בעל משוב לא לינארי כמו באיור 2.8.



איור 2.8 – לולאת החישוב מוצגת כאוגר הואה לא לינארי [\[3\]](#).

בכל סיבוב, הפונקציה מחשבת 16 מילים חדשות, המילה המחשבת הראשונה תתווסף לבlok הנתונים A כך שכךת גודלו יהיה 90 מילים.

בשורת החישוב הראשונה, עושים XOR בין מילה בוקטור S המשתנה בכל סיבוב, לבין שני מילים בבלוק הנתונים כאשר הבחירה של המילים נעשית ע"י index המיקום ואחד קבוע עמדות ההק莎 t .

בשורה השנייה עושים XOR עם התוצאה של השורה הראשונה עם 2 תוצאות של AND הנעשית בין שני מילים מבלוקי הנתונים אשר מזוהות ע"י קבוע עמדות ההקsha t .

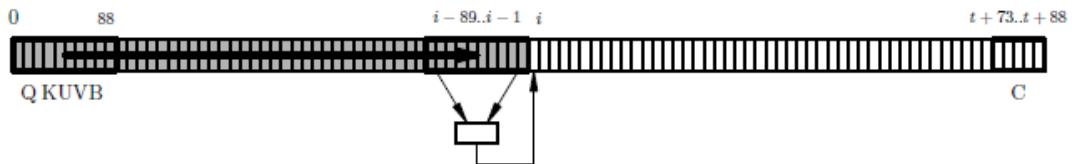
בשורה השלישי עושים XOR עם התוצאה של השורה השנייה ועם אותה תוצאה אשר מזוהה ימינה לפי וקטורי הazea r .

בשורה הרביעית עושים XOR עם התוצאה של השורה השלישי ועם אותה תוצאה אשר מזוהה שמאליה לפי וקטורי הazea l .

התוצאה מוכנסת לבlok הנתונים במיקום של ה-index $-i$.

קבוע עמדות ההקsha t כמו שהסבירנו לעיל נמצאים בטוחה הנ"ל $89 < t_0 < 16$ ולכן:

- כל סיבוב בחישוב משתמש רק ב-89 המילים ב-index היכי גבוהה של בלוק הנתונים (89 המילים האחרונות), כמפורט באIOR 2.9, מה שמאפשר חיסכון בזיכרון כאשר בכל סיבוב נוסיף את ה-16 המילים החדשות ונוריד את 16 המילים הראשונות מבלוק הנתונים.
- משומם שאוון תלות של צעד אחד בפלט של האחר לפחות 16 צעדים, ניתן לבצע בחומרה 16 צעדים בתוך סיבוב בעליית שעון אחת. בכך ניתן לחשב פונקציית דחיסה מלאה ב- 2 מחזורי שעון.

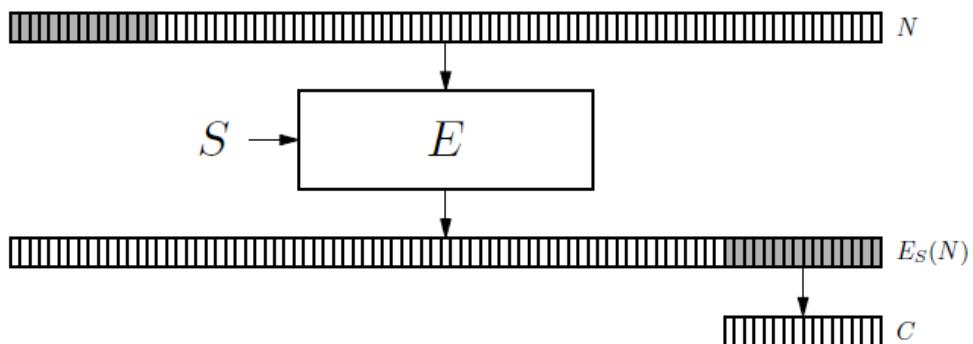


IOR 2.9 – אופן הפעולה של לולאת החישוב [3].

2.3.4.3 פלט פונקציית הדחיסה

פלט פונקציית הדחיסה זהו 16 המילים האחרונות שחשבנו בלולאת החישוב של פונקציית הדחיסה כמו בIOR 2.10

במידה ומדובר בפונקציית הדחיסה الأخيرة, מתוך 16 המילים לוקחים את d הסיביות האחרונות אשר יהו את הגיבוב הסופי.



איור 2.10 – פונקציית הדחיסה f נראית כפעולות הצפנה ואחריה פעולה חיתוך [3].

3 פיתוח ושיטות

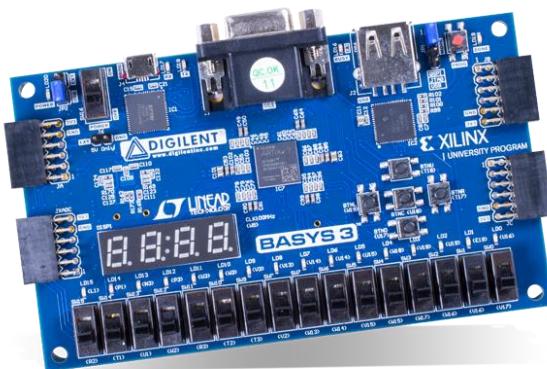
במסגרת הפרויקט, מומש אלגוריתם MD6 אשר מכיל פונקציה דחיסה אחת בלבד אשר פועלת לפי מצב הפעולה הסטנדרטי, עקב מחסור במשאבי רכיב החומרה כפי שIOSBR להלן.

קישור למימוש מצב הפעולה (הנמצאים בתיקיית הפרויקט) נמצא [בנספח א](#).

פרק זה מחולק באופן הבא:

- ערכת הפיתוח ורכיב ה-FPGA
- מעתפת להעברת נתונים
- מימוש האלגוריתם בשפת חומרה
- הטעמה על רכיב ה-FPGA
- יצירת קוד תוכנה עם תוספת ממוקש משתמש

3.1 ערכת הפיתוח ורכיב ה-FPGA [8]



.אирו 3.1 – ערכת הפיתוח Basys-3 [11].

במסגרת הפרויקט התכנון הוטמעה על רכיב FPGA מסוג 7-Artix של חברת Xilinx-AMD, הכלול בערכת הפיתוח Basys-3 של חברת Digilent. כמפורט באירור 3.1.

טבלה 5 מציגה את המאפיינים העיקריים של ערכת הפיתוח.

טבלה 5: מאפייני ערכת הפיתוח.

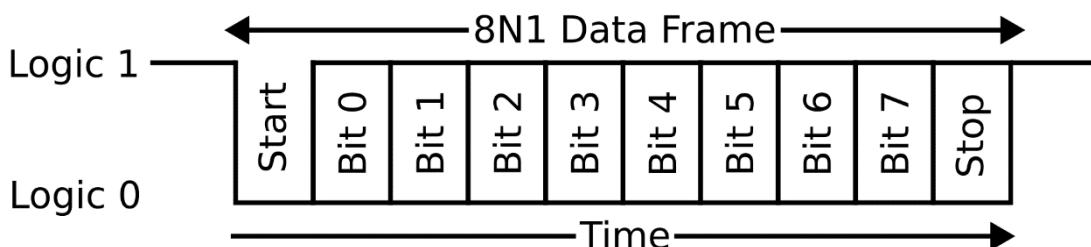
| תבונה | תיאור |
|----------------------|--|
| רכיב ה-FPGA | Artix-7 XC7A35T-1CPG236C |
| ממשקים בניות ויציאות | USB-UART • USB-UART Bridge • יציאת VGA של 12 סיביות • יציאת USB HID Host לעבריים מקלדות ודיברונות • |
| זיכרון | 32Mb Serial Flash |
| מסך | מסך אחורית 4 ספירות ב-7 פלחים |
| מצלמות ונויריות | 16 מצלמים • 16 נוירות לד • 5 כפتورים • |
| শעונים | מתנד בעל תדר של 100MHz |

| | | |
|---|--------|--------------|
| XADC Pmod עבר אוותות 3 יציאות Pmod | • • | יציאות הרחבה |
| 63,280 תאים לוגיים ב-5200 חלקיים (כל חלק מכיל ארבעה ZUTS עם 6 כניסה ו-8 FFs) 1800Kb של זיכרון RAM בлок מהיר | • • | אמצעים |

3.2 מטפחת להעברת נתונים

בכדי לגיבב את ההודעה, תחיליה יש "להקשר את הקרקע" ולהזכיר את המטפחת שתנהל את המידע ש מגיע מהמחשב לוח ומשדר בהזרה. לשם כך מימשו פרוטוקול תקשורת UART מסוג 8N1 [9].

(8 סיביות, ללא סיבית זוגיות, ועם סיבית עצירה אחד) כמפורט באיור 3.2. הסיבה שבחרנו בפרוטוקול זה היא משומן שאנו משתמשים בערכת הפיתוח-h3 Basysmicro USB [8]. קיים בה רכיב UART מובנה בכוניסת ה-

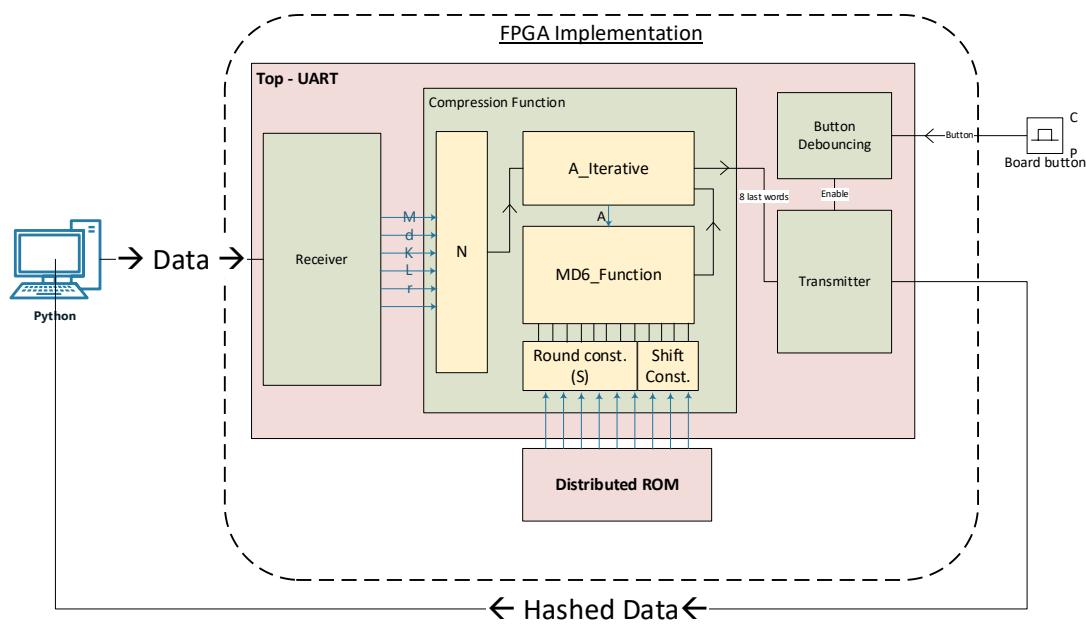


איור 3.2 – מבנה שליחת המידע בתקשורת UART מסוג 8N1 [12].

המידע המתתקבל מהמשתמש יכול להיכתב בייצוג בינארי, הקסדצימלי או ASCII. באמצעות פרוטוקול-h3-TX נשלח מידע מסווג byte בלבד. נדרש להמיר את המידע המתתקבל מהמשתמש, למידע מסווג byte ורק אז להעביר דרך UART.

בנוסף, כדי להקל על החומרה מחישובים מיוחדים (רוטציות, ריפוד באפסים, חישוב אורכי וקטור...), המידע מוקן מראש, כך שהוא מרופד באפסים בגודלים קבועים כדי לקטalg את סוג המידע למקומות המתאים באлогריתם וכמו כן, תוכך כדי שהמידע מתתקבל דרך UART, הוא מסווג ב-Big-Endian כך שאין צורך לעשות רוטציה כלשהי. חזץ מהמידע המתתקבל מהמשתמש, נוסף מידע לצורך בקרה ובכך הושג חיסכון בחישובים בחומרה שבזבזים משאבים קריטיים.

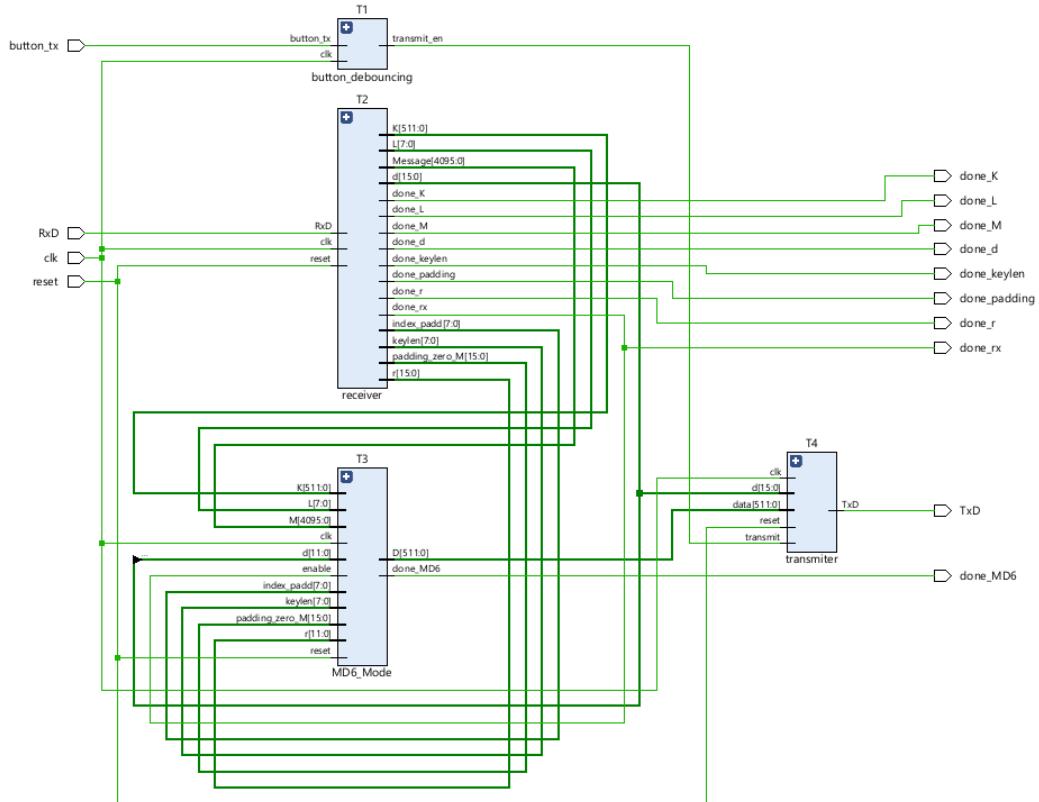
באיור 3.3 ניתן לראות דיאגרמת בלוקים של התוכנו כולל מטפחת העברת המידע.



איור 3.3 – דיאגרמת הבלוקים של התוכן כולל המעטפת.

3.3 מימוש האלגוריתם בשפת חומרה

בפרק זה נפרט על כתיבת קוד RTL-Verilog שהetuמענו על רכיב ה-FPGA כמפורט בדיאגרמת הבלוקים באירור 3.4.



איור 3.4 – דיאגרמת הבלוקים של המימוש.

פירוט ה-I/O והאותות הפנימיים של התכונן מופיע בטבלאות 6 ו-7.

טבלה 6: הקלט והפלט של התכונן.

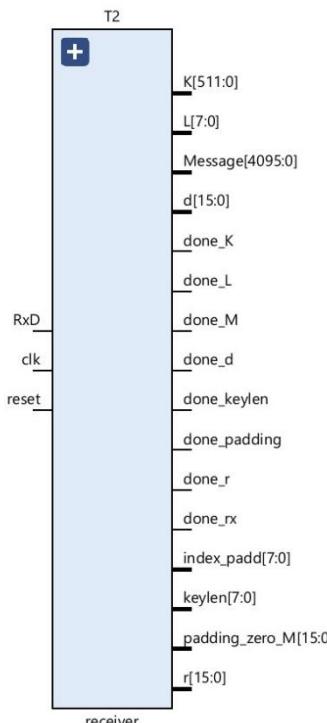
| שם האות | גודל | קלט/פלט | תיאור |
|--------------|----------|---------|-------------------------------------|
| Clk | 1 סיביות | קלט | תדר השעון של הרכיב ZH00MHz |
| Reset | 1 סיביות | קלט | אות האיפוס |
| Button tx | 1 סיביות | קלט | אות לשילוח המידע המגובב חזרה |
| RxD | 1 סיביות | קלט | סיבית קבלת המידע ע"י UART-U |
| TxD | 1 סיביות | פלט | סיבית החזרת המידע ע"י UART-U |
| Done M | 1 סיביות | פלט | נורט סימון לסיום קבלת ההודעה |
| Done d | 1 סיביות | פלט | נורט סימון לסיום קבלת אורך הגיבוב |
| Done K | 1 סיביות | פלט | נורט סימון לסיום קבלת המפתח |
| Done L | 1 סיביות | פלט | נורט סימון לסיום קבלת פרט בקרת המצב |
| Done r | 1 סיביות | פלט | נורט סימון לסיום קבלת מספר הסיבובים |
| Done keylen | 1 סיביות | פלט | נורט סימון לסיום קבלת אורך המפתח |
| Done padding | 1 סיביות | פלט | נורט סימון לסיום קבלת אורך הריפוד |
| Done MD6 | 1 סיביות | פלט | נורט סימון לסיום קבלת כל המידע |

טבלה 7: האותות הפנימיים של התכונן.

| שם האות | גודל | תיאור | մԵԼՈԿ | մԵԼՈԿ |
|----------------|-------------|------------------------------|-------------|-------------------------|
| Message | 4096 סיביות | ההודעה | MD6 Mode | receiver |
| D | 16 סיביות | אורך ההודעה המגובבת | MD6 Mode | receiver |
| K | 512 סיביות | המפתח | MD6 Mode | receiver |
| L | 8 סיביות | פרט בקרת המצב | MD6 Mode | receiver |
| R | 16 סיביות | מספר הסיבובים | MD6 Mode | receiver |
| Keylen | 8 סיביות | אורך המפתח בbyteים | MD6 Mode | receiver |
| padding zero M | 16 סיביות | אורך ריפוד ההודעה | MD6 Mode | receiver |
| index padd | 8 סיביות | מספר פונקציות הדחיסה | MD6 Mode | receiver |
| Hash | 512 סיביות | המידע המגובב | transmitter | MD6 mode |
| transmit en | 1 סיביות | אות בקרה לשילוח המידע המגובב | transmitter | Button debouncing block |

כפי שניתן לראות באירור 3.4, ההיררכיה הראשונה בתכונן מחולקת ל-4 בלוקים.

- – בלוק קבלת המידע ע"י UART – receiver
- – הבלוק המכיל את תכונן MD6 Mode.
- – בלוק החזרת המידע ע"י UART-U – transmitter
- – בלוק לביטול רעשים בלחצן החזרת המידע – button debouncing



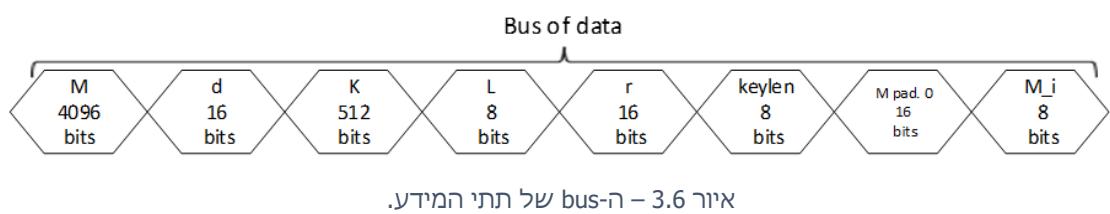
איור 3.5 – סימבול בлок ה-receiver.

3.3.1 Receiver בлок
כפי שהוסבר לעיל העברת המידע לרכיב החומרה וקלטנו חזרה נועשתה ע"י תקשורת טוריות – UART.

בלוק ה-receiver כמפורט באיוור 3.5, לקבלת המידע לגיבוב, המידע מתתקבל ב-sus אחד של סיביות המורכב מכמה סוגים של תתי מידע, כפי שמתואר באיוור 3.6. בCLK יש שליטה על סדר המידע המתתקבל כך שכל חלק וחולק יגיע למקום הנכון. בשבייל להתחם receiver מוציא את ביציאה אשר מדליק נורה על ערכת הפיתוח Basys3 בכל פעם שהוא מסיים לקבל חלק מהמידע.

לאחר קבלת כל המידע, ה-receiver מעביר את המידע לאחר הסידור לבlok ה-MD6_Mode.

פירוט ה-IO והאותות הפנימיים של ה-receiver מופיע בטבלאות 8 ו-9.



איור 3.6 – ה-sus של תתי המידע.

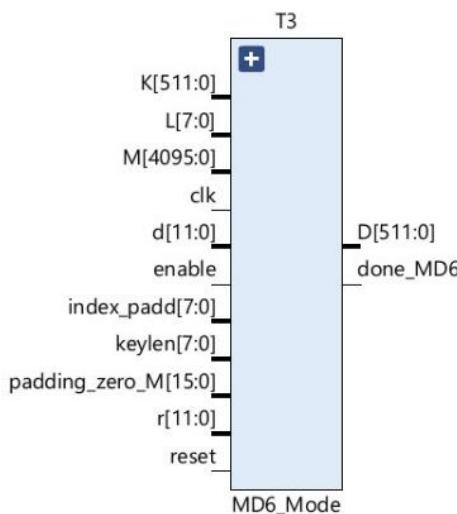
טבלה 8: הקלט והפלט של בлок ה-receiver.

| תיאור | קלט/פלט | גודל | שם אות |
|---|---------|-------------|----------------|
| תדר השעון של הרכיב 100MHz | קלט | 1 סיביות | Clk |
| אות האיפוס | קלט | 1 סיביות | Reset |
| קבלת המידע ע"י ה-UART | קלט | 1 סיביות | RxD |
| ההודעה | פלט | 4096 סיביות | Message |
| אורך ההודעה המוגבהת | פלט | 16 סיביות | D |
| המפתח | פלט | 512 סיביות | K |
| פרמטר בקרת המצב | פלט | 8 סיביות | L |
| מספר הסיבובים | פלט | 16 סיביות | R |
| אורך המפתח בbytes | פלט | 8 סיביות | Keylen |
| אורך ריפוד ההודעה | פלט | 16 סיביות | padding zero M |
| מספר פונקציות הדחיסה | פלט | 8 סיביות | index M |
| נורט סימון לקבלת ההודעה בהצלחה | פלט | 1 סיביות | done M |
| נורט סימון לקבלת אורך הודעת הגיבוב בהצלחה | פלט | 1 סיביות | done d |
| נורט סימון לקבלת המפתח בהצלחה | פלט | 1 סיביות | done K |
| נורט סימון לקבלת פרמטר בקרת המצב בהצלחה | פלט | 1 סיביות | done L |
| נורט סימון לקבלת מספר הסיבובים בהצלחה | פלט | 1 סיביות | done r |
| נורט סימון לקבלת אורך המפתח בהצלחה | פלט | 1 סיביות | done keylen |

| | | | |
|-------------------------------------|-----|----------|---------------------|
| נורט סימון לקבלת אורך הריפוד בהצלחה | פלט | 1 סיביות | done padding |
| נורט סימון לסיום קבלת כל המידע | פלט | 1 סיביות | done rx |
| נורט סימון לסיום גיבוב ההודעה | פלט | 1 סיביות | done MD6 |

טבלה 9: האותות הפנימיים של הבלוק ה-*receiver*.

| שם האות | גודל | תיאור |
|----------------------------|-------------|--|
| data reg M | 4096 סיביות | אוגר לשמירת ההודעה |
| data reg d | 16 סיביות | אוגר לשמירת אורך ההודעה המגובבת |
| data reg K | 512 סיביות | אוגר לשמירת המפתח |
| data reg L | 8 סיביות | אוגר לשמירת פרמטר בקרת המצב |
| data reg r | 16 סיביות | אוגר לשמירת מספר הסיבובים |
| data reg keylen | 8 סיביות | אוגר לשמירת אורך המפתח בתים |
| data padding | 16 סיביות | אוגר לשמירת אורך ריפוד ההודעה |
| data reg index padd | 8 סיביות | אוגר לשמירת מספר פונקציות הדחיסה |
| Index byte M | 3 סיביות | אות עזר לסידור ההודעה |
| Word M | 11 סיביות | אוגר למניית בתי ה Hodouah שהתקבלו |
| Index d | 3 סיביות | אוגר למניית בתי אורך ה Hodouah שהתקבלו |
| Index byte K | 3 סיביות | אות עזר לסידור ההודעה |
| Word K | 8 סיביות | אוגר למניית בתי המפתח שהתקבלו |
| Index L | 2 סיביות | אוגר למניית בתי אורך המצביע שהתקבלו |
| Index r | 3 סיביות | אוגר למניית בתי מספר הסיבובים בתים שהתקבלו |
| Index keylen | 2 סיביות | אוגר למניית בתי אורך המפתח בתים שהתקבלו |
| Index padding | 3 סיביות | אוגר למניית בתי אורך ריפוד ההודעה שהתקבלו |
| Index index padd | 2 סיביות | אוגר למניית בתי מספר פונקציות הדחיסה שהתקבלו |
| Shift | 1 סיביות | אות הפעלה להזנת מידע |
| State | 1 סיביות | אוגר בקרת המצביעים של מוכנות המצביעים |
| Nextstate | 1 סיביות | אות בקרה למעבר מצב ל מצב במוכנות המצביעים |
| Clear bitcounter | 1 סיביות | אות בקרה לאיפוס bitcounter |
| Inc bitcounter | 1 סיביות | אות בקרה להעלאת bitcounter |
| Clear samplecounter | 1 סיביות | אות בקרה לאיפוס samplecounter |
| Inc samplecounter | 1 סיביות | אות בקרה להעלאת samplecounter |
| Bitcounter | 4 סיביות | מונה של 4 סיביות לספירה עד 9 עבור קליטת UART |
| Samplecounter | 2 סיביות | מונה דגימה של 2 סיביות לספורה עד 4 עבור oversampling |
| Counter | 14 סיביות | מונה של 14 סיביות לספירת קצב ה- Baud rate |
| Rxshiftreg | 10 סיביות | אוגר הזנת סיביות |



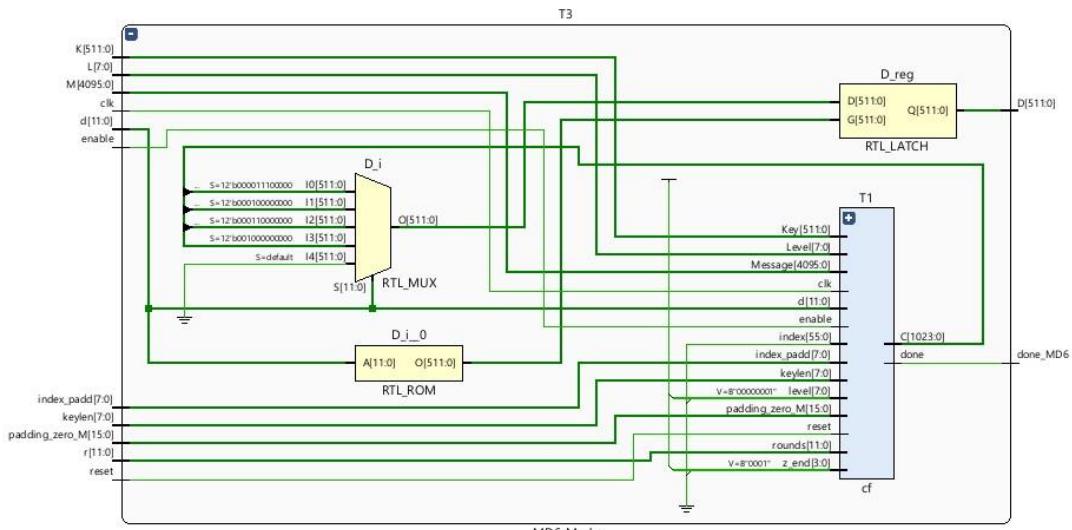
.איור 3.7 – סימבול בлок ה-MD6 Mode

3.3.2 בлок ה-MD6 Mode

בלוק ה-MD6 Mode מקבל את המידע האחרון לאחר סידורו מבלוק ה-receiver ומוציא את הودעה המוגובבת כמו צג באיור 7.

בלוק ה-MD6_Mode מכיל את פונקציה הדחיסה אשר מוציאה פלט של 16 מיילים ומהם בлок ה-A (אורך ההודעה המוגובבת שבחר המשתמש) הסיביות האחרונות ורפס אוטם (לפי הצורך) לגודל של 512 סיביות כמו צג באיור 8.

פירוט ה-IO והאותות הפנימיים של ה-MD6 Mode מופיע בטבלאות 10-11.



.איור 3.8 – בлок ה-MD6 Mode

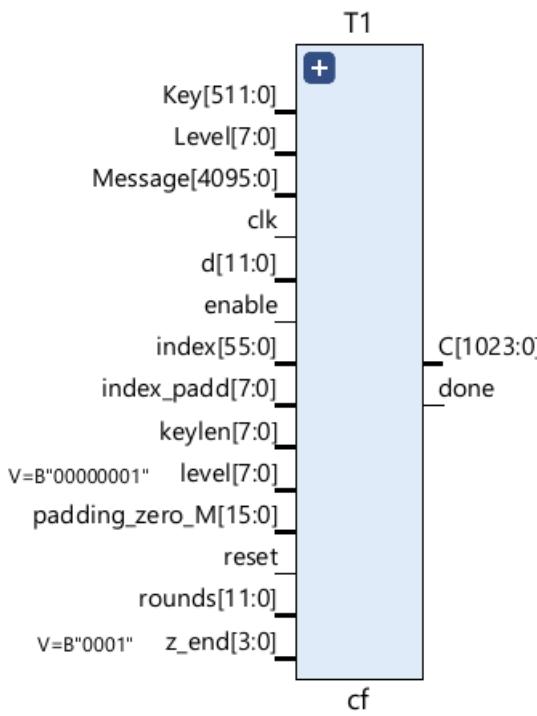
טבלה 10: הקלט והפלט של בлок ה-MD6 Mode

| שם אות | גודל | קלט/פלט | תיאור |
|----------------|-------------|---------|----------------------------------|
| Clk | 1 סיביות | קלט | תדר השעון של הרכיב 100MHz |
| Reset | 1 סיביות | קלט | אות האיפוס |
| Enable | 1 סיביות | קלט | אות בקרה להפעלת האלגוריתם ההודעה |
| Message | 4096 סיביות | קלט | המפתח |
| D | 16 סיביות | קלט | אורך ההודעה המוגובבת |
| K | 512 סיביות | קלט | פרמטר בקרה המצב |
| L | 8 סיביות | קלט | מספר הסיבובים |
| R | 12 סיביות | קלט | אורך המפתח בתבים |
| Keylen | 8 סיביות | קלט | אורך ריפוד ההודעה |
| padding zero M | 16 סיביות | קלט | מספר פונקציות הדחיסה |
| index padd | 8 סיביות | קלט | |

| | | | |
|-------------------------------|-----|------------|----------|
| ההודעה המוגבהת | פלט | 512 סיביות | D |
| נורת סימון לסיום גיבוב ההודעה | פלט | 1 סיביות | Done MD6 |

טבלה 11: האותות הפנימיים של בлок ה-MD6 Mode.

| שם האות | גודל | תיאור |
|---------|-------------|---|
| i | 1024 סיביות | מחבר בין יציאת פונקציית הדחיסה לבחירת אורך היציאה |

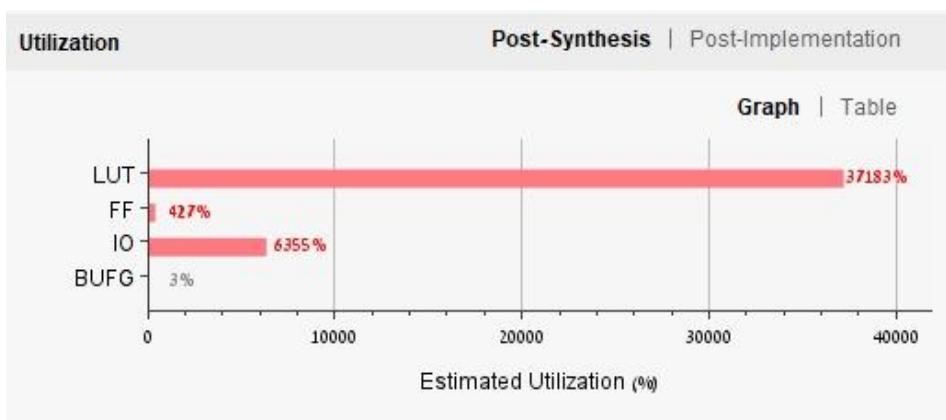


איור 3.9 – סימבול בлок ה-cf.

באים 3.9 זו פונקציית הדחיסה המרכזית של אלגוריתם MD6 כמפורט [בפרק 2.3](#).
במסגרת הפרויקט תוכננה תחילה פונקציית הדחיסה בצורה מופשטת. הפונקציה מכילה אוגר שגודלו 64 סיביות (89 המילימ' הראשונות בבלוק הנתונים יחד עם מספר הסיבובים המksamלי 168 כפול 16 מילימ' שנוצרות בכל סיבוב), והפעלו לולאה אשר מחשבת 16 מילימ' בכל סיבוב ומכניסה אותן לבלוק הנתונים לפי הסדר, ובסוף היא מוציאה את 16 המילימ' האחרונות שנוצרו לאחר סיום לולאת החישוב.

החיסרון בתכנון זה הינו הדרישת לעודף משאבי החומרה הנדרכים למימוש התכנון (ביחס ל-spec של ה-3 Basys3).

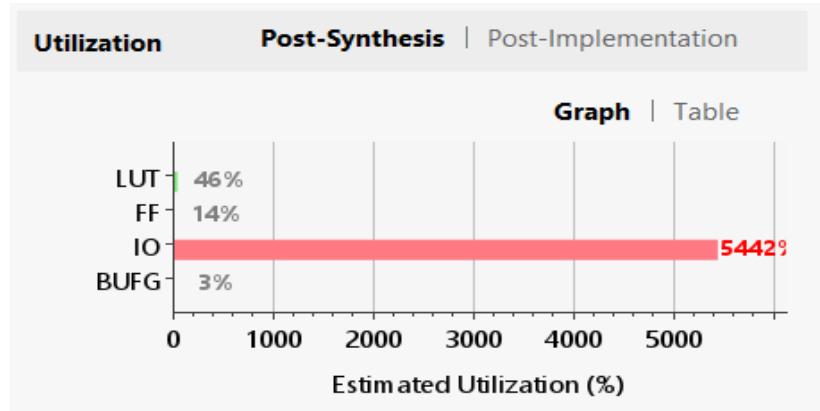
בסביבות 400% שימוש ב-FFs-1 (37,000%) שימוש ב-LUTs). עצם זה שקיים אוגר אשר מכיל גודל של 2,777 מילימ', שבו עושים שימושים לוגיים ומשתמשים ב-FFs-1 ו-LUTs-1 לצורך שמירת המידע, גורם להריגה כה גדולה מכמויות המשאבים המוקצים כמתואר באים 3.10 (אין להתייחס באים לחלק של מספר היציאות והכניות, כיון שהוא בניתוח בлок ה-cf בלבד, ולא עם מעטפת ה-UART).



איור 3.10 – אחוז רכיבי החומרה המשומשים בתכנון הראשוני של בлок ה-cf.

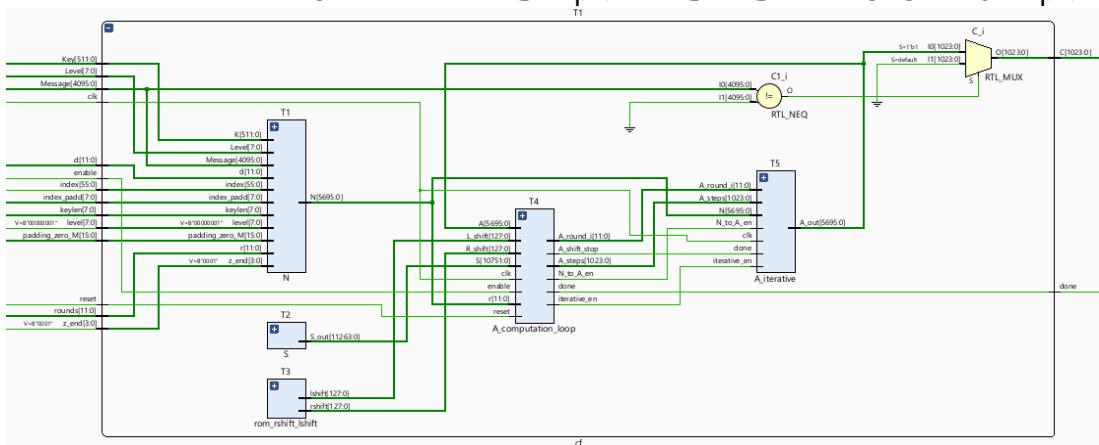
בשלב הראשון, מעריכו חישובים לא-אלגוריתמיים, כגון חישוב ריפוד אפס, סיבוב סדר הסיביות של הוקטור (כדי לקבל סדר Big-Endian) או חישוב המוצא של ערך ה-*h-sus*.
לחילופין, נעשה שימוש בנתונים שהתקבלו מהקלט של פרוטוקול התקשרות UART ומונף כוח העיבוד של המחשב בו פועל התוכנו לביצוע חישובים אלו בתוכנה. בנוסף, נעשה שימוש ב-*rom* המבואר המבנה של ה-*FPGA* כדי למנע עומס יתר של משאבי ה-LUT.

בשלב השני, לוLAT החישוב הראשית יושמה באמצעות מודל logic combinational. המשמעות היא שיש פחות שטחה של ערכי ביניים וביצעו חישובים בכמה שפותחות *FFs/LUTs*. למעשה נמצאה יתרות באלגוריתם בכר שנשמרו וקטורים גם שכבר לא היו בתוקף. החישובים מבוססים רק על 89 המילימ"מ האחרונות של הוקטור A (בלוק הנתונים). לכן, במקום לאחסן את כל הערכים שאינם בשימוש ובמקום לשמר את A כוקטור גדול של כל המילימ"מ שחוובנו, הושגה דרך ליישם גישה איטרטיבית להיכソン בשיטה, על ידי ביצוע פעולה Shift להכנסת 16 המילימ"מ החדשות לתוך וקטור באורך קבוע של 89 מילימ"מ. כך הושג חסוך מאד גדול ברכבי חומרה והצלחנו לעמוד במספר המשאים המוגבל כמו צג באירור 3.11.



איור 3.11 – איחר רכבי החומרה המשמשים בתוכנו הסופי של בלוק ה-*cf*.

בלוק ה-*cf* האיטרטיבי מורכב מכמה בלוקים כמו צג באירור 3.12.



איור 3.12 – בלוק ה-*cf*.

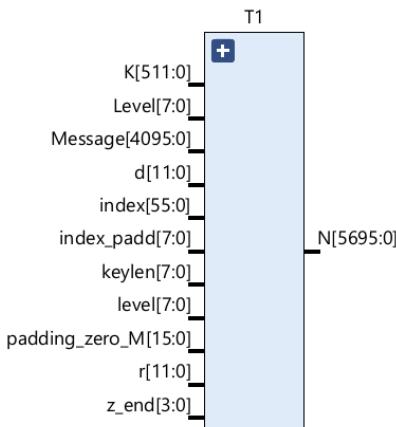
פירוט ה-IO והאותות הפנימיים של ה-*cf* מופיע בטבלאות 12 ו-13.

טבלה 12: הקלט והפלט של בлок ה-cf.

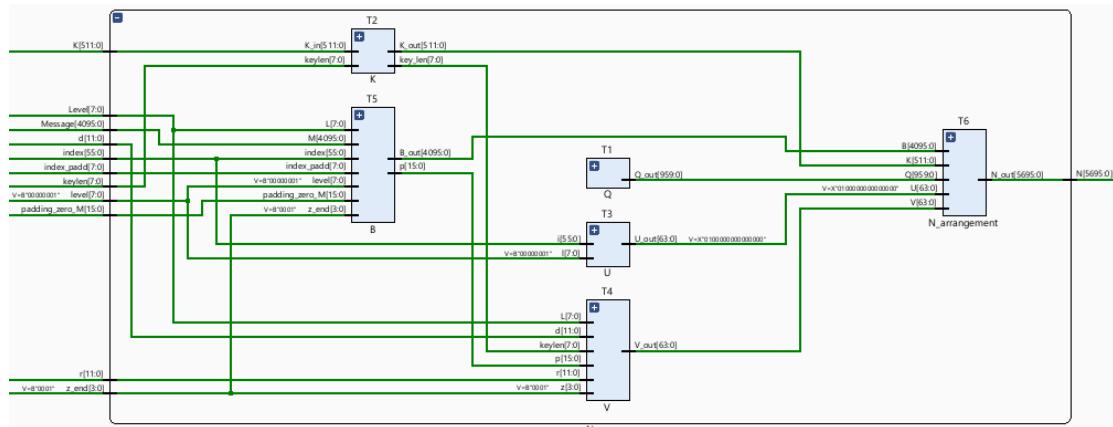
| שם האות | גודל | קלט/פלט | תיאור |
|----------------|-------------|---------|--|
| Clk | 1 סיביות | קלט | תדר השעון של הרכיב ZMH100 |
| Reset | 1 סיביות | קלט | אות האיפוס |
| Enable | 1 סיביות | קלט | אות בקרה להפעלת האלגוריתם ההודעה |
| Message | 4096 סיביות | קלט | אורק ההודעה המוגבהת |
| D | 16 סיביות | קלט | המפתח |
| Key | 512 סיביות | קלט | פרמטר בקרת המצב |
| Level | 8 סיביות | קלט | מספר הסיבובים |
| Rounds | 12 סיביות | קלט | אורק המפתח בbiteים |
| Keylen | 8 סיביות | קלט | אורק ריפוד ההודעה |
| padding zero M | 16 סיביות | קלט | מספר פונקציות הדחיסה |
| index padd | 8 סיביות | קלט | השלב הבוכחי שבו נמצאת פונקציית הדחיסה |
| Level | 8 סיביות | קלט | האינדקס הנוכחי שבו נמצאת פונקציית הדחיסה |
| Index | 56 סיביות | קלט | שווה 1 אם זאת פונקציית הדחיסה الأخيرة |
| z end | 4 סיביות | קלט | 16 המילים האחרונות שנוצרו |
| C | 1024 סיביות | פלט | נורט סימון לסיום פונקציית הדחיסה |
| Done | 1 סיביות | פלט | |

טבלה 13: האותות הפנימיים של בлок ה-cf.

| שם האות | גודל | mblock | לבולוק | תיאור |
|--------------|--------------|-------------------|--------------------|---------------------------------------|
| N | 5696 סיביות | | A iterative | בלוק הנתונים N |
| A steps | 1024 סיביות | | A iterative | 16 המילים הנוצרות בפונקציית הדחיסה |
| A round i | 12 סיביות | | A iterative | מספר הסיבוב הבוכחי |
| N to A en | 1 סיביות | | A iterative | אות בקרה להכנסת בלוק הנתוני הראשוני |
| Iterative en | 1 סיביות | | A iterative | אות בקרה לתחילת פעולה פונקציית הדחיסה |
| A shift stop | 1 סיביות | | A iterative | אות בקרה לסיום פעולה פונקציית הדחיסה |
| S | 10752 סיביות | | A computation loop | וקטור הקבועים S |
| Rshift | 128 סיביות | rom rshift lshift | A computation loop | קבועי ההזזה ימינה |
| Lshift | 128 סיביות | rom rshift lshift | A computation loop | קבועי ההזזה שמאליה |
| R | 12 סיביות | N | A computation loop | מספר הסיבובים |
| A | 5696 סיביות | A iterative | A computation loop | בלוק הנתונים הנכנס לוולאת החישוב |



איור 3.13 – סימבול בлок ה-N.



איור 3.14 – בлок ה-N.

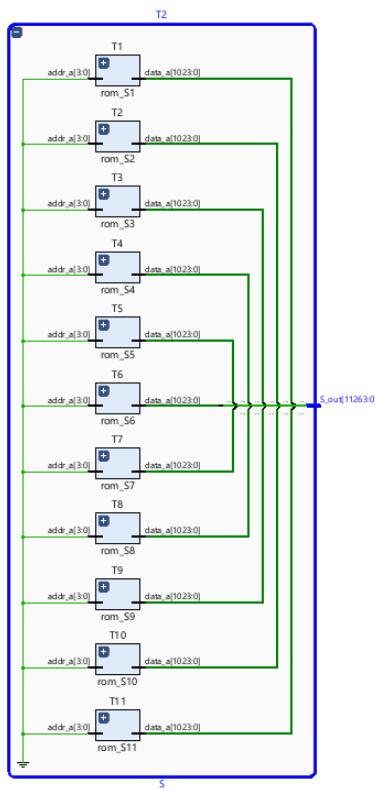
טבלה 14: הקלט והפלא של בлок ה-N.

| שם האות | גודל | קלט/פלט | תיאור |
|----------------|-------------|---------|--|
| Message | 4096 סיביות | קלט | ההודעה |
| d | 16 סיביות | קלט | אורך ההודעה המוגובבת |
| Key | 512 סיביות | קלט | המפתח |
| Level | 8 סיביות | קלט | פרמטר בקרת המצב |
| rounds | 12 סיביות | קלט | מספר הסיבובים |
| keylen | 8 סיביות | קלט | אורך המפתח בbiteים |
| padding zero M | 16 סיביות | קלט | אורך ריפוד ההודעה |
| index padd | 8 סיביות | קלט | מספר פונקציות הדחיסה |
| level | 8 סיביות | קלט | השלב הנוכחי שבו נמצאת פונקציית הדחיסה |
| Index | 56 סיביות | קלט | האינדקס הנוכחי שבו נמצאת פונקציית הדחיסה |
| z end | 4 סיביות | קלט | שוואה 1 אם זאת פונקציית הדחיסה الأخيرة |
| N | 5696 סיביות | פלט | בלוק הנתונים הראשוני - N |

טבלה 15 האותות הפנימיים של בлок N.

| שם האות | גודל | טבולה | טבולה |
|---------|------------|-------|-----------------|
| Q to N | 960 סיביות | Q | וקטור הקבועים Q |

| | | | | |
|---------------|---|--------------------|-------------|--------------------|
| N arrangement | K | המפתח | 512 סיביות | K to N |
| N arrangement | K | אורך המפתח בביטים | 8 סיביות | Keylen to V |
| N arrangement | U | מזהה הצומת הייחודי | 64 סיביות | U to N |
| V | B | אורך ריפוד ההודעה | 16 סיביות | P to V |
| N arrangement | V | AMILת הבקרה | 64 סיביות | V to N |
| N arrangement | B | ההודעה | 4096 סיביות | B to N |



איור 3.15 – סימבול בлок ה-S.

3.3.2.1.2 בлок ה-S

בלוק ה-S הוא בлок המכיל את וקטור הקבועים S, אשר משמשים כחלק מחישוב פונקציית הדחיסה כמפורט בפרק 2.3.2.

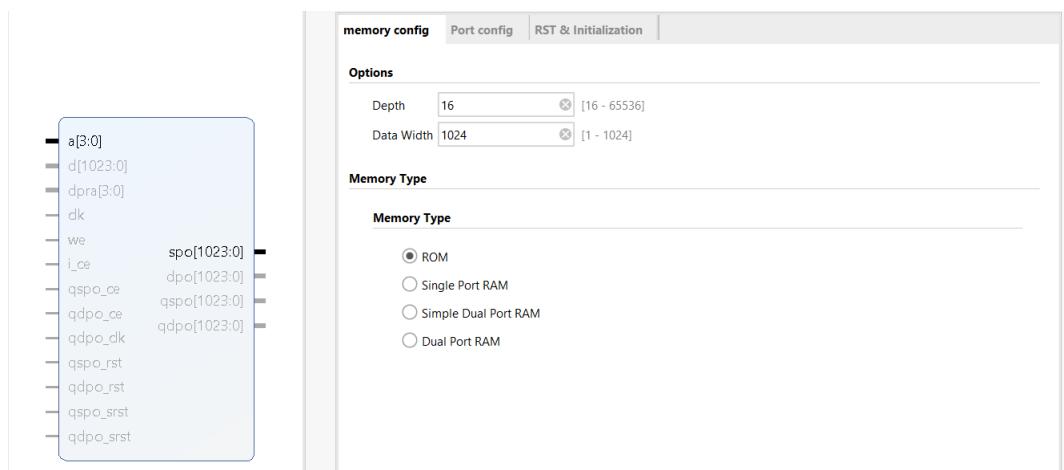
בלוק ה-S מורכב מ-11 בлокי rom אסינכרוניים אשר כל בлок מכיל 1024 סיביות מוקטור הקבועים כמפורט באירור 3.15. (המידע מוכל בקבצי mem)

בלוקי ה-rom האסינכרוניים ברכיב ה-GA-FPGA בו מומש התכנון, מסוגלים להכיל 65,536 כתובות אשר בכל כתובת נוכנסים 1024 סיביות כמפורט באירור 3.16.

כדי להבהיר את הקבועים בפעימה אחת, נבחרה כתובת קבועה בכניסה לבlokי ה-rom, ובכל בлок, המידע אוחSEN רק בכתובת זו.

גודל וקטור ה-S הוא $10752 = 168 \cdot 64$, ובכל בлок נכנס מידע בגודל 1024 סיביות, לכן מספר בлокי ה-rom שהוצרכו זה $\frac{10752}{1024} = 10.5 \rightarrow 11$ (הблוק האחרון).

של ה-rom מכיל בחציו הראשון את 512 האחרונים של וקטור הקבועים S וחציו השני מכיל אפסים).



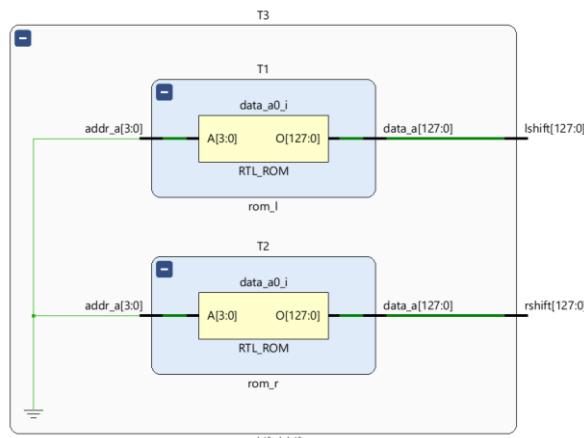
איור 3.16 – גודל בлокי ה-rom האסינכרוני ברכיב ה-FPGA.

3.3.2.1.3 בлок ה-rom rshift Ishift

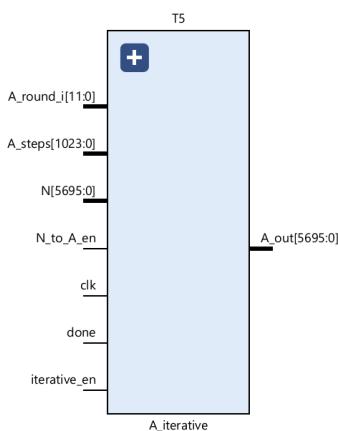
בלוק ה-`rshift` ו-`Ishift` מכיל את קבועי הרצה `rshift` ו-`Ishift` אשר משמשים כחלק מחישוב פונקציית הדחיסה כמפורט בפרק 2.3.2.

כמו בлок S גם הבלוק `rom rshift Ishift` מורכב מבLOCKי `rom` אסינכרוניים כמוzeigt באירור 3.17.

קבועי הרצה `rshift` ו-`Ishift` מכילים כל אחד 16 קבועים, כאשר הוקצה לכל קבוע גודל של byte אחד, לכן בסה"כ ל-`rshift` ול-`Ishift` מספיק בלוק `rom` אחד לכל אחד, כי כל אחד צריך מקום של $128 = 8 * 16$ סיביות.



אייר 3.17 – סימבול הבלוק ה-`rom rshift Ishift`



אייר 3.18 – סימבול הבלוק ה-`rom rshift Ishift`

A iterative בлок `A iterative` תפקido לייצור את האיטרטיביות של התכנון, כאשר הוא מקבל את הבלוק הנדרטנים ההתחלתי `A` ואת 16 המילימ החדשנות שנוצרו בבלוק החישוב כמוzeigt באירור 3.18.

בסיבוב הראשון הוא מוציא בבלוק החישוב את בלוק הנדרטנים `A`, ולאחר מכן בכל סיבוב בכל אחד מ-16 המילימ הרשונות של הבלוק ומוסיף את 16 המילימ החדשנות שהתקבלו בסוף הבלוק, כך שבכל סיבוב י יצא שהוא אוגר 89 מילימ.

פירוט ה-IO של ה-`A iterative` מופיע בטבלה 16 (אין אותן פנימיות).

טבלה 16: הקלט והפלט של הבלוק ה-`A iterative`.

| שם האות | גודל | קלט/פלט | תיאור |
|---------------------------|-------------|---------|---|
| <code>Clk</code> | 1 סיביות | קלט | תדר השעון של הרכיב 100MHz |
| <code>N to A en</code> | 1 סיביות | קלט | אות בקרה להבנתה בлок הנדרטנים הראשוניים |
| <code>Iterative en</code> | 1 סיביות | קלט | אות בקרה לתחילה פעולה פונקציית הדחיסה |
| <code>Done</code> | 1 סיביות | קלט | אות בקרה לתחילה האיטרטיביות |
| <code>A round i</code> | 12 סיביות | קלט | מספר הסיבוב הנוכחי |
| <code>N</code> | 5696 סיביות | קלט | בלוק הנדרטנים <code>N</code> |
| <code>A step</code> | 1024 סיביות | קלט | 16 המילימ החדשנות בפונקציית הדחיסה |
| <code>A out</code> | 5696 סיביות | קלט | בלוק הנדרטנים החדש |

3.3.2.1.5 בлок ה-loop A computation

בלוק ה-loop A computation מכיל את האלגוריתם של לולאט פונקציית הדחיסה כרך שבכל סיבוב הבלוק מקבל בלוק נתונים של 89 מילימ ומוסיא 16 מילימ חדשות כמפורט במספר 3.19, כמפורט באירור 3.19.

הблוק בכל סיבוב מחשב 16 מילימ חדשות ומעביר אותן לבlok ה-e-iterative A שמוסיף אותן לבlok הנתונים ומוריד את 16 המילימ הראשונות בבלוק הנתונים, וכרך בлок ה- iterative A מוחזר בлок נתונים חדש לבlok loop A computation ווחזר חלילה עד סיום מספר הסיבובים המתකבל מהמשתמש.

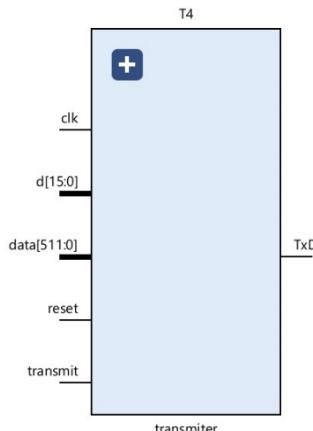
פירוט ה-IO והאותות הפנימיים של ה- A computation סעודה מופיע בטבלה 17 ו-18.
אירור 3.19 – סימבול בлок ה-loop .

טבלה 17: הקלט והפלט של בлок ה-loop A computation

| שם האות | גודל | קלט/פלט | תיאור |
|--------------|--------------|---------|---------------------------------------|
| Clk | 1 סיביות | קלט | תדר השעון של הרכיב 200MHz |
| Reset | 1 סיביות | קלט | אות האיפוס |
| Enable | 1 סיביות | קלט | אות בקרה להפעלת האלגוריתם |
| R | 12 סיביות | קלט | מספר הסיבובים |
| S | 10752 סיביות | קלט | וקטור הקבועים S |
| R shift | 128 סיביות | קלט | קובעי ההזהה ימינה |
| L shift | 128 סיביות | קלט | קובעי ההזהה ימיןנה |
| A | 5696 סיביות | קלט | בלוק הנתונים בכניסה |
| N to A en | 1 סיביות | פלט | אות בקרה להבנתה בлок הנתונים הראשוני |
| Iterative en | 1 סיביות | פלט | אות בקרה לתחלת פעולות פונקציית הדחיסה |
| Done | 1 סיביות | פלט | אות בקרה לתחלת האיטרטיביות |
| A shift stop | 1 סיביות | פלט | אות הבקרה לסיום האיטרטיביות |
| A round i | 12 סיביות | פלט | מספר הסיבוב הנוכחי |
| A steps | 1024 סיביות | פלט | 16 המילימ החדשות שנוצרו |

טבלה 18: אותות הפנימיים של בлок ה-loop A computation

| שם האות | גודל | תיאור |
|---------|-----------|---------------------|
| J | 12 סיביות | מונה למנית הסיבובים |
| State | 3 סיביות | מצבים המكونת מצבים |



איור 3.20 – סימולר של מודול transmitter.

3.3.3 בлок ה-transmitter

לאחר סיום גיבוב המידע יש לשלווה את המידע המגובב חזרה אל המשטמש, בлок ה-transmitter כמפורט באIOR 3.20 מקבל את המידע המגובב ואת אורך ושולח אותו byte אחר byte לשליחת תקשורת UART, שליחת המידע נעשית לאחר קבלת אות מהמשתמש ע"י להיצעה על לחץ שליחת המידע בערכת הפיתוח.

פירוט ה-IO והאותות הפנימיים של ה-transmitter מופיע בטבלה 19-20.

טבלה 19: הקלט והפלט של בлок ה-transmitter.

| שם האות | גודל | קלט/פלט | תיאור |
|----------|------------|---------|-------------------------|
| Clk | 1 סיביות | קלט | תדר השעון של הרכיב 2MHz |
| reset | 1 סיביות | קלט | אות האיפוס |
| transmit | 1 סיביות | קלט | אות בקרה לשילוח המידע |
| d | 16 סיביות | קלט | אורך המידע המגובב |
| data | 512 סיביות | קלט | המידע המגובב |
| TxD | 1 סיביות | קלט | שליחת המידע ע"י ה-UART |

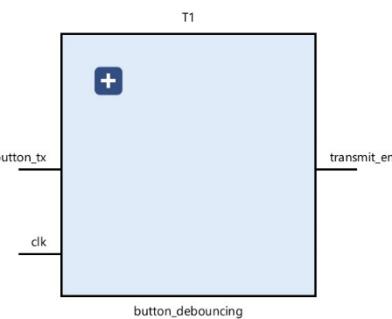
טבלה 20: האותות הפנימיים של בлок ה-transmitter.

| שם האות | גודל | תיאור |
|---------------|------------|---|
| state | 1 סיביות | אוגר בקרת המצביעים של מכונת המצביעים |
| nextstate | 1 סיביות | אות בקרה למעבר ממצב למצב במכונת המצביעים |
| shift | 1 סיביות | אות הפעלה להזנת מידע |
| load | 1 סיביות | אות בקרה כדי להתחיל לטען את הנתונים לתוך אוגר ההזנה ולהוסיף סיביות התחלת ועצירה |
| clear | 1 סיביות | אות בקרה לאיפוס ה-bitcounter |
| index | 7 סיביות | אוגר ערך למינון הסיביות לשילוח |
| Index reg | 7 סיביות | אוגר ערך למינון הבטים לשילוח |
| rightshiftreg | 640 סיביות | אוגר שמירת המידע המרופד בסיביות התחלת ועצירה |
| bitcounter | 11 סיביות | מונה למינית הסיביות בשילוח byte המרופד |
| counter | 14 סיביות | מונה למינית קבוע ה-rate baud |

3.3.4 בлок ה-Button Debouncing

קבלת המידע המוגובב חזקה נעשית ע"י לחיצת כפתור בערכת הפיתוח-h3-Basys3, כאשר לוחצים על כפתור הוא עלול לקפוץ פיזית עקב המגעים המכניים שבתוכו יוצרים חיבורים וניתוקים געויים ומהירים ואלו יכולים לגרום לצירתאות שווא מרובים במקום אחת יחד.

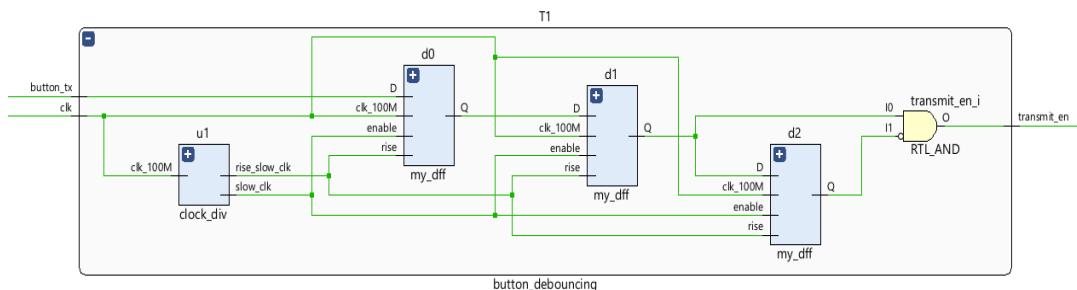
לפתרון הבעיה תוכנן בлок ה-**Button Debouncing** כמפורט באIOR 3.21.



AIOR 3.21 – סימבול בлок ה-**button_debouncing**

כדי לבטל את אותן השווא, נוצר שעון בעל תדרות נמוכה ביחס לשעון הראשי, כך

שישאפשר לדגם את אותן ולדלג על אותן השווא. לפי שעון זה, אותן מועבר בשלשה FFs אשר לוכדים אותן בשלושה מהווים שעון שונים כמפורט באIOR 3.22, וכן כל אותן השווא שנוצרו מסוננים.



. AIOR 3.22 – בлок ה-**button debouncing**.

פירוט ה-IO והאות הפנימיות של ה-block **button debouncing** מופיע בטבלאות 21 ו-22.

טבלה 21: הקלט והפלט של בлок ה-**button debouncing**.

| שם האות | גודל | קלט/פלט | תיאור |
|-------------|----------|---------|----------------------------------|
| Clk | 1 סיביות | קלט | תדר השעון של הרכיב 100MHz |
| Button tx | 1 סיביות | קלט | אות שליחת המידע המתקבל מהמשתמש |
| Transmit en | 1 סיביות | קלט | אות שליחת המידע אחריו סיכון רעים |

טבלה 22: אותן הפנימיות של בлок ה-**button debouncing**.

| שם האות | גודל | תיאור |
|---------------|----------|-----------------------------|
| Rise slow clk | 1 סיביות | דילג לעלייה שעון בשעון איטי |
| Slow clk | 1 סיביות | השעון האיטי |
| Q1 | 1 סיביות | אות יציאה מה-FF הראשון |
| Q2 | 1 סיביות | אות יציאה מה-FF השני |
| Q2_bar | 1 סיביות | היפוך ה-Q2 |
| Q0 | 1 סיביות | אות היציאה מה-FF השלישי |

3.4 הטעמה על רכיב FPGA

הטעמת המימוש על הרכיב כוללת את החלקים הבאים:

- Constraint Specification – מפרט אילוצים
- Synthesis – סינתזה
- Place & Route – מיקום וחיזוק
- Bitstream generation and Device program – גיבוב גיבוב ותוכנה

שלבי ההטמעה נעשו בכלים התוכנה VIVADO של חברת Xilinx, גרסה 2022.1.

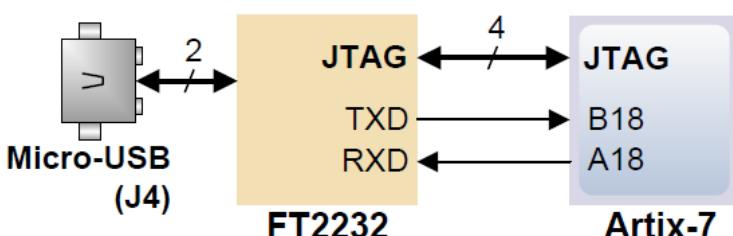
3.4.1 – מפרט אילוצים Constraint Specification

mprt האילוצים מצין אילוצים שונים כדי להנחות את תהליכי הסינתזה והיישום. אילוצים אלה כוללים אילוצי שעון (לדוגמה, תדר שעון, תחומי שעון), אילוצי קלט/פלט (לדוגמה, הקצאות פינים) ומגבליות זמן (למשל, זמני הגדרה והחזקקה).

mprt האילוצים נכתב בקובץ (Xilinx Design Constraints) המבוסס טקסט ומשמש לצורך הגדרת האילוצים של התוכן.

להלן פירוט האילוצים:

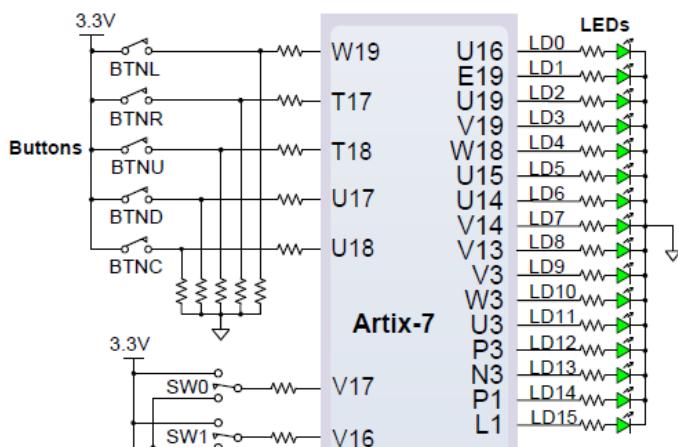
- :Clocks הוגדר אילוץ זמן מחוזר של 10ns או במילימרים אחורות תדר של 100MHz ב-duty cycle של 50%.
- :Input & Output delay הוגדר שבכניתם המידע ובחזרתו מהרכיב לא יהיה זמן עיכוב (לא נדרש כי כל המידע נכנס באותו מקום אחד אחרי השני).
- :Configuration settings הוגדר שהמתח שבו יפעלו פיני תצורת ה-FPGA הוא CFGBVS VCCO והוא הוגדר את מתח התצורה של ה-FPGA ל-3.3 וולט.
- Pin locations & voltages כל הפינים הוגדרו לרמת מתח לוגית של LVCMOS33 – זאת אומרת רמה לוגית של 3.3V CMOS.
- להלן פירוט חיבוריו הייחודיים לפינים:
 - השעון : clk – מחובר לפין W5 אשר שם מחובר המתנד של Basys3.
 - תקשורת ה-UART:
 - A18 – מחובר לפין TxD
 - B18 – מחובר לפין RxD
- בפנים אלו מחובר רכיב ה-UART של מערכת הפיתוח Basys3 כמפורט באיור 3.23.



איור 3.23 – חיבורו תקשורת ה-UART בערכת הפיתוח.

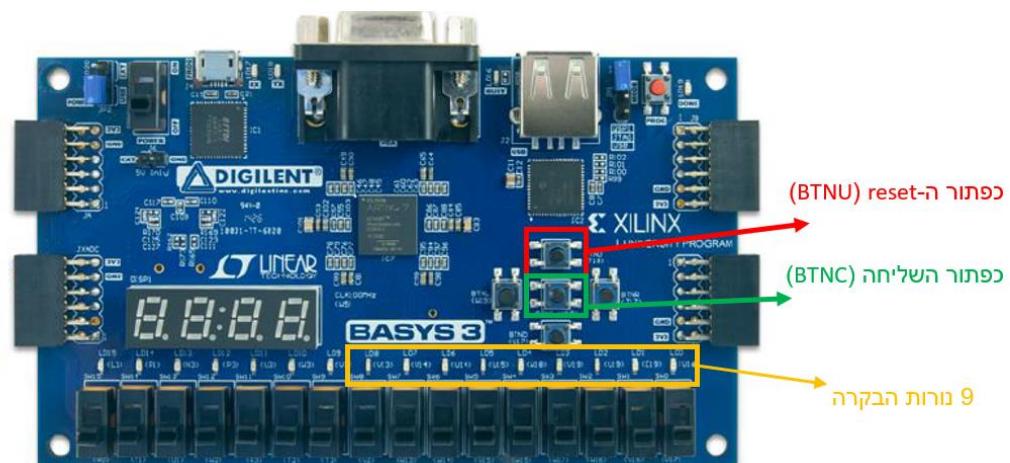
- כפתורי הלחיצה:
 - מחובר לפין T18 אשר מחובר לכפתור הלחיצה BTNU.
 - מחובר לפין U18 אשר מחובר לכפתור הלחיצה BTNC.
- כמפורט באירור 3.24.

- נורות:
 - M – מחובר לפין U16 אשר מחובר לנורה 0. LD0
 - d – מחובר לפין E19 אשר מחובר לנורה 1. LD1
 - Done K – מחובר לפין U19 אשר מחובר לנורה 2. LD2
 - Done L – מחובר לפין V19 אשר מחובר לנורה 3. LD3
 - Done r – מחובר לפין W18 אשר מחובר לנורה 4. LD4
 - Done keylen – מחובר לפין U15 אשר מחובר לנורה 5. LD5
 - Done padding – מחובר לפין U14 אשר מחובר לנורה 6. LD6
 - Done rx – מחובר לפין V14 אשר מחובר לנורה 7. LD7
 - Done MD6 – מחובר לפין V13 אשר מחובר לנורה 8. LD8
- כמפורט באירור 3.24.



איור 3.24 – חיבור הנורות והלחצנים לפינים בדרכת הפיתוח.

את מיקום כפתורי הלחיצה והנורות על דרכת הפיתוח אפשר לראות באירור 3.25.



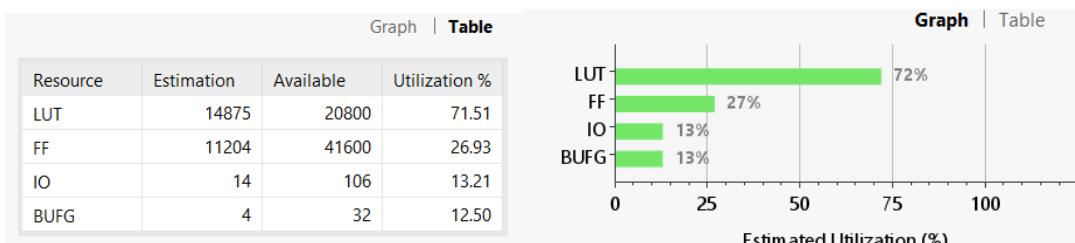
איור 3.25 – מיקום הכפתורים והנורות על דרכת הפיתוח.

סינתזה Synthesis 3.4.2

סינתזה מלאת תפקיד מהותי בהמרת תיאור חומרה בرمאה גבוהה (הכתוב בשפת Verilog ל-netlist שהוא ייצוג ברמה נמוכה של התכnuן שnitnu להשתמש בו לצורך אופטימיזציה והטמעה).

התוכנה לוקחת את קוד ה-RTL ומתרגם אותו לייצוג מבני של התכnuן. לאחר מכן מבצעת טכניקות אופטימיזציה שונות כדי לשפר את ביצועי התכnuן, ניטול השטח וצricht החישמל, ע"י הפחמת אוגרים מיותרים ויעול החיות בין החלקים השונים. לאחר מכן היא מempה את התכnuן המסתונאי על פנוי רכיב-h- FPGA שבו השתמשנו.

כפי שהוסבר בפרק 3.3.2 בחלקי קוד RTL והסימולציה של הדוח המצורפים [בנספח א](#). התכnuן המקורי (רף ה-cf) לאחר העברתו בסינתזה השתמש ביחס ל-spec של ה-Basys3, בסביבות 400% שימוש ב-FFs ו-37,000% שימוש ב-LUTs מרכבי החומרה הקיימים ברכיב-h FPGA, כמו גם באירור 3.10. לאחר המעבר לתכnuן האיתרטיבי, התכnuן לאחר סינתזה עבר שינוי משמעותי החומרה בו הוא משתמש כמו גם באירור .3.26.



איור 3.26 – כליה החומרה המנותלים לאחר סינתזה.

התוכנה לוקחת בחשבון את האילוצים המתקיים בתכnuן ונונתת משוב לעשות שינויים כדי להתאים את התכnuן כנדרש.

בנוסף במקרה של שגיאות או כאשר התוכנה מזיהה בעיות היכולות לאגורם לכשלים בתכnuן או דברים הנחtinyים ליעול, התוכנה שולחת משוב בדמות "שגיאה" במקרה של שגיאות ו"ازהרות" במקרה של בעיות או דברים הנחtinyים לשיפור.

בתכnuן שלנו, יש 205 אזהרות המת balloות בתהליך הסינתזה. נפרט ונסביר את הסיבה לכך מופיעות.

- אזהרה: [Synth 8-7129]

כמות אזהרות: 100

הסביר:

הזהרה מצינית שכלי הסינתזה זיהה שיתכן שחלק מהאותות בתכnuן עברו אופטימיזציה, אך הם לא הוסרו. אזהרה זו נוצרת בדרך כלל כאשר יש אותות בתכnuן שאינם בשימוש או שאין להם השפעה על הפונקציונליות של התכnuן.

בתכnuן שלנו:

הסיבה לאזהרה היא כי פונקציית הדחיסה בנוייה כך שהיא לא משתמשת בכל המידע, כיון שהוא תלוי בשימושם של קבועים $t_4 - t_1$. משום קבועים אלו באלגוריתם לא חיבים להיות בערכיהם שונים להם ומה בעלי שינוי, אז המידע שלא בשימוש בתכnuן הנוכחי נוצר.

- אזהרה: [Synth 8-3917]

כמות אזהרות: 100

הסבר:

הזהרה מצינית שכלי הסינטזה זיהה מצב שעלול לגרום למחוקת או להתנהגות לא מוגדרת במהלך פעולת התוכן. אזהרה זו נוצרת בדרך כלל כאשר יש התנגשות פוטנציאלית בין שני אותן או יותר בתוכן.

בתוכנו שלנו:

הסיבה לאזהרה היא כי נכנס ערך 0 לאותות בתחילת התוכן, ואין התנגשויות עם אותות אחרים. כזכור שהאיפוס של האותות נדרש כדי להשתמש בתוכן באופן רב פעמי.

- אזהרה: [Synth 8-689]

כמות אזהרות: 1

הסבר:

הזהרה מצינית שכלי הסינטזה זיהה שיש אי התאמה בין רוחב אות או חיבור יצאה בתוכן לבין רוחב היציאה המתאימה במופע מודול.

בתוכנו שלנו:

הסיבה לאזהרה היא כי 512 הביטים האחרונים שמקבלים מה-rom לא מועברים לפקטור S. אנחנו לא מעבירים אותם משום שהם לא נדרכים כי הם לא חלק מהקבועים, וכל בית שם שווה 0. הסיבה שלהם נמצאים זה בשביל שכל הגדים של ה-rom יהיו באותו הגודל.

- אזהרה: [Synth 8-6014]

כמות אזהרות: 3

הסבר:

הזהרה מצינית שכלי הסינטזה זיהה אלמנט רציף בתוכן שאינו בשימוש והוסר.

בתוכנו שלנו:

האלמנטים המذוברים נחוצים לאתחול התוכנית לשימוש חור בכל סוג המצביעים בקוד.

- אזהרה: [Synth 8-327]

כמות אזהרות: 1

הסבר:

הזהרה מצינית שכלי הסינטזה זיהה משתנה בתוכן שלא מוקצת לו ערך בכל מצב אפשרי, ויצר latch לאחסון הערך של המשתנה.

בתכנון שלנו:

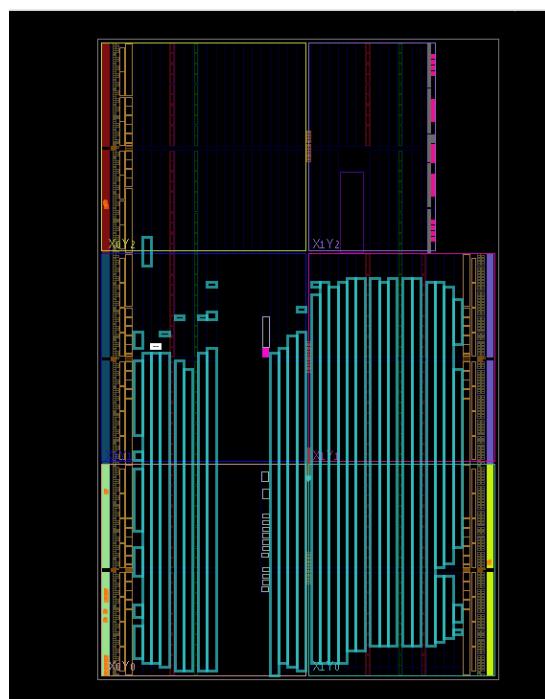
ה-P נוצר כדי לקבל את החלק היוצא של המידע המוצפן, אך הוא מוגדר רק לערכים ספציפיים כי הוא מוגדר במספר אפשרויות מוגבל.

Place & Route 3.4.3 – מיקום וחיווט

מיקום וחיווט - במהלך שלב זה כל התוכנה לוקח את ה-netlist שנוצר בשלב הסינטזה וממפה פיזית את הלוגיקה הדיגיטלית על המשאבים הפיזיים של ה-FPGA.

בתחילת התוכנה קובעת היכן כל אלמנט לוגי צריך להיות ממוקם ברכיב FPGA. זה הכרוך בבחירה תאים לוגיים ספציפיים ודגלגים כדי למקם את הביצועים ולמזרע את עיבובי החיווט. המטרה היא ליעל את המיקום עבור גורמים כמו התפשטות האותות, ניהול השטח ועמידה במוגבלות הזמן.

לאחר מיקום האלמנטים הלוגיים, התוכנה קובעת כיצד לחבר אותם באמצעות משאבי החיווט הזמינים, כגון חיבורים, חוטים ומתקנים הנitinens להתקנות-ב-FPGA. חיווט יעיל חיוני כדי למזרע עיבוביים ולעמוד בדרישות הזמן. באIOR 2.27 אפשר לראות את התקנון ממוקם על רכיב ה-FPGA, ובאיור 2.28 רואים את מפת הכניסות והיציאות של הרכיב.

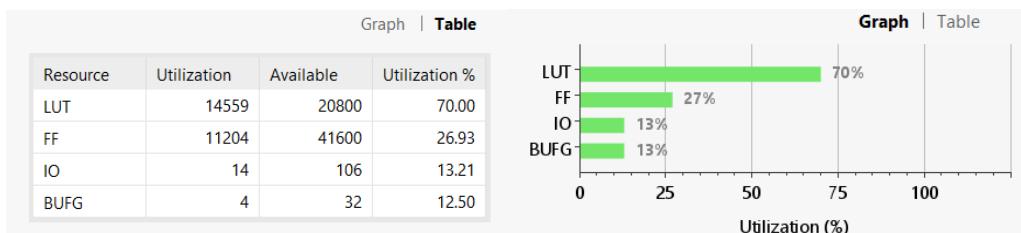


איור 3.27 – מיקום רכיבי החומרה המשומשים לאלגוריתם.



איור 3.28 – מפת הכנסיות והיציאות של הרכיב.

כלי התוכנה עוקב אחר השימוש במשאבי FPGA כמו **LUT**, **DLLGs**, **מרבבים** ו**קשרים הדדיים** כדי להבטיח ניצול יעיל. מטרתו היא לסייע בזיהוי משאבים ברכיב. השימוש במשאבים הסופי על הרכיב מוצג באיור 3.29.



איור 3.29 – כלוי החומרה המונוצלים לאחר מיקום וחיווט.

בנוסף, כלי התוכנה מבצע את ניתוח התזמון של התכנון כדי להבטיח שהוא עומד במוגבלות הזמן ובאלגוריתם הקיימים, באיור 3.30 אפשר לראות את ניתוח התזמון של התכנון.

Design Timing Summary

| Setup | Hold | Pulse Width |
|---|---|--|
| Worst Negative Slack (WNS): 0.171 ns | Worst Hold Slack (WHS): 0.010 ns | Worst Pulse Width Slack (WPWS): 4.500 ns |
| Total Negative Slack (TNS): 0.000 ns | Total Hold Slack (THS): 0.000 ns | Total Pulse Width Negative Slack (TPWS): 0.000 ns |
| Number of Failing Endpoints: 0 | Number of Failing Endpoints: 0 | Number of Failing Endpoints: 0 |
| Total Number of Endpoints: 28154 | Total Number of Endpoints: 28154 | Total Number of Endpoints: 11205 |

All user specified timing constraints are met.

איור 3.30 – ניתוח התזמון של התכנון.



Bitstream Generation and Device Programming 3.4.4

לאחר שהמיקום והחויט מצלחים, כלי התוכנה מייצר את קובץ ה-Bitstream שמנדר את ה-A-FPGA עם הלוגיקה המתוכנת, כך שהיא ניתנת להטמעת התוכן על רכיב ה-FPGA.

3.5 ייצור קוד תוכנה ל קישור בין החומרה למשתמש

כפי שהוסבר לעיל האלגוריתם הוא אלגוריתם גיבוב המקבל מידע מהמשתמש ומהיזיר אותו מגובב חזרה.

כדי לקשר בין החומרה לבין המשתמש ולהתאים את המידע לשילוחה לרכיב החומרה, נוצר קובץ הפעלה exe בשם MD6_CF.exe.

קובץ הפעלה נכתב בשפת Python עם ממשק משתמש (GUI) אשר מאפשר את הפעלת הגיבוב בצורה נוחה.

נפרט על התהליך העובר על המידע מקבלתו מהמשתמש עד לשילוחו לרכיב החומרה.

- קבלת המידע מהמשתמש:**

כמו所述 בפרק 2.3 באלגוריתם MD6 יש מידע קלט שהוא אופציוני (המשתמש בחור אם להכניס אותו). לכן התוכנה תשאל את המשתמש איזה מידע אופציוני הוא בחור להכניס ואיזה לא, כאשר המשתמש בחור לא להכניס מידע אופציוני התוכנה תכניס למידע האופציוני את ערכיו בירית המחדל.

בנוסף להודעה והמפתח יכולים להתקבל ע"י המשתמש בשלושה סוגי פורמטים שונים byte, ASCII ו-binary. לכן התוכנה תשאל את המשתמש באיזה פורמט הוא רוצה להכניס את ההודעה ואת המפתח.

לאחר סיום "שאלות ההגדלה", המשמש מכניס כל מידע ומידע בגודל

- ייצור מידע עזר:**

לאחר קבלת המידע התוכנה יוצרת 3 נתוני עזר אשר מועברים בנוסף לרכיב החומרה:

- keylen – אורך המפתח ב-byte.
- Padding M – אורך ריפוד המידע להודעה.
- Index M – מספר פונקציות הדחיסה.

- המרת המידע לפורמט byte:**

העברת המידע דרך התקשרות הטורית UART ע"י ספריית serial בשפת Python מתאפשר רק כאשר המידע המועבר הוא בפורמט byte لكن כל המידע עובר המרה לפורמט זה.

- ריפוד המידע באפסים וסידור:**

כדי שכל בלוק מידע יועבר למיקומו הנכון ברכיב החומרה, כל בלוק מידע מרופד באפסים עד כדי גודלו המקורי.

- 512 bytes – M ◦
- 2 bytes – d ◦
- 8 bytes – K ◦

- 1 byte – L ○
- 2 bytes – r ○
- 1 byte – keylen ○
- 2 bytes – padding M ○
- 1 byte – Index M ○

בנוסף חוץ מההוודה והמפתח, כל בלוק מידע עובר היפוך כך שה-byte-first מועבר לסוף וכן הלאה, כדי להתאים את שליחית המידע לאלגוריתם.

- **שרשור כל המידע:**
לשם שליחת המידע דרך תקשורת הטורית, כל המידע משורשר אחד אחרי השני לשילוח בצורה יעילה ומהירה.
- **שליחת וקבלת המידע דרך תקשורת הטורית:**
כדי לשולח את המידע דרך תקשורת הטורית, התוכנה תבקש מהמשתמש להכנס את כנישת COM-ה אליה מחובר ערכות הפיתוח.
לאחר הכנסת מסטר COM, התוכנה שולחת את המידע המשורשר ע"י שימוש בספריית serial בשפת Python, ע"י שימוש בפקודה `ser.write()` ומקבלת את המגובה חוזרת מהמשתמש ע"י פקודת `.ser.read()`.

4 תוצאות

בפרק זה מוצגים הפעלת התכנון, הסימולציות ותוצאות הבדיקות שנעדו לאמות התכנון.

4.1 הפעלת אלגוריתם MD6

פרק זה מפרט את הפעלת אלגוריתם MD6 שימוש בפרויקט, צעד אחר צעד.

בביצוע הפעלת האלגוריתם יש שני חלקים מרכזיים:

- הטמעת אלגוריתם MD6 על רכיב FPGA.
- הchèלט גיבוב המידע על רכיב FPGA.

כדי שיהיה ניתן להפעיל את האלגוריתם, יש לוודא שהדברים הבאים נמצאים בהישג יד:

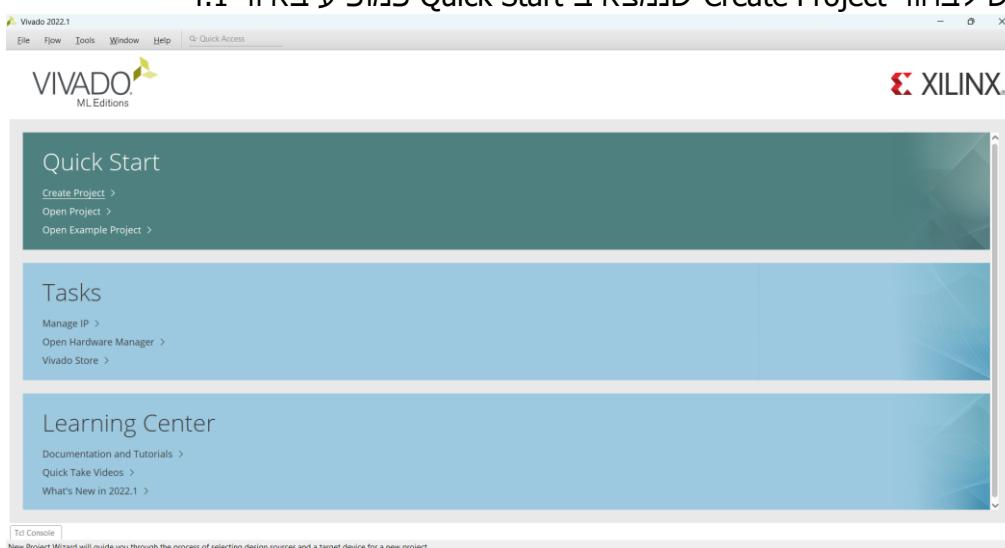
- תיוקנית הקבצים של הפרויקט.
- ערכת הפיתוח Basys3.
- כלי התוכנה OS VIVADO.

קישור לתקינות קבצי הפרויקט ומודריך בוידאו להפעלת האלגוריתם מצורפים [בנספח א'](#)

4.1.1 תחילת הטמעת אלגוריתם MD6 על רכיב ה-A

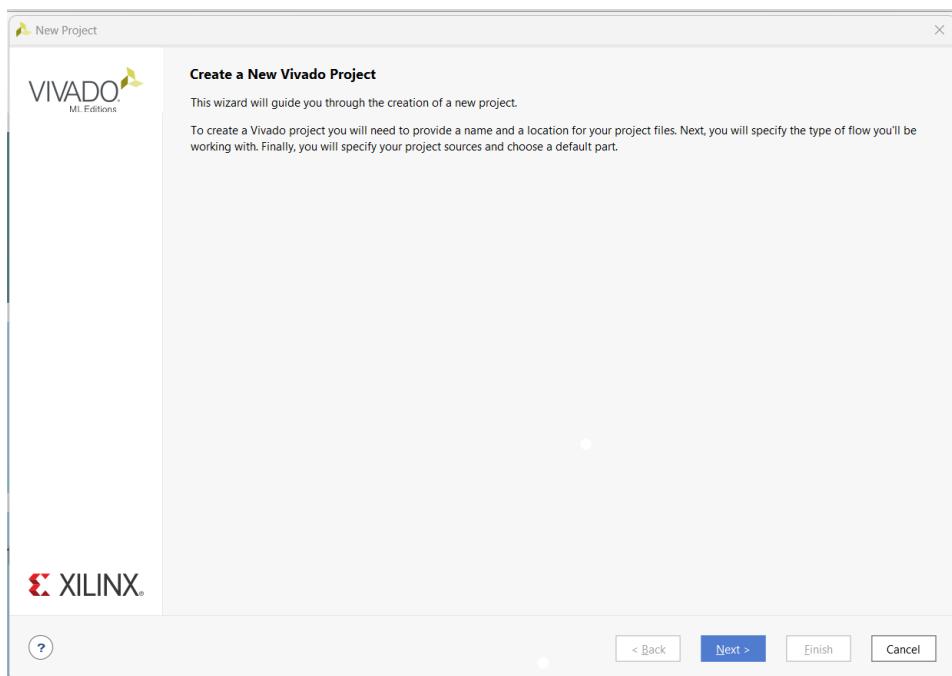
פרק זה מפרט את תחילת הטמעת אלגוריתם MD6 על רכיב ה-A.

- תחיליה יש להפעיל את כלי התוכנה VIVADO.
- יש לבחור Create Project שנמצא ב-Quick Start כמפורט באירור 4.1

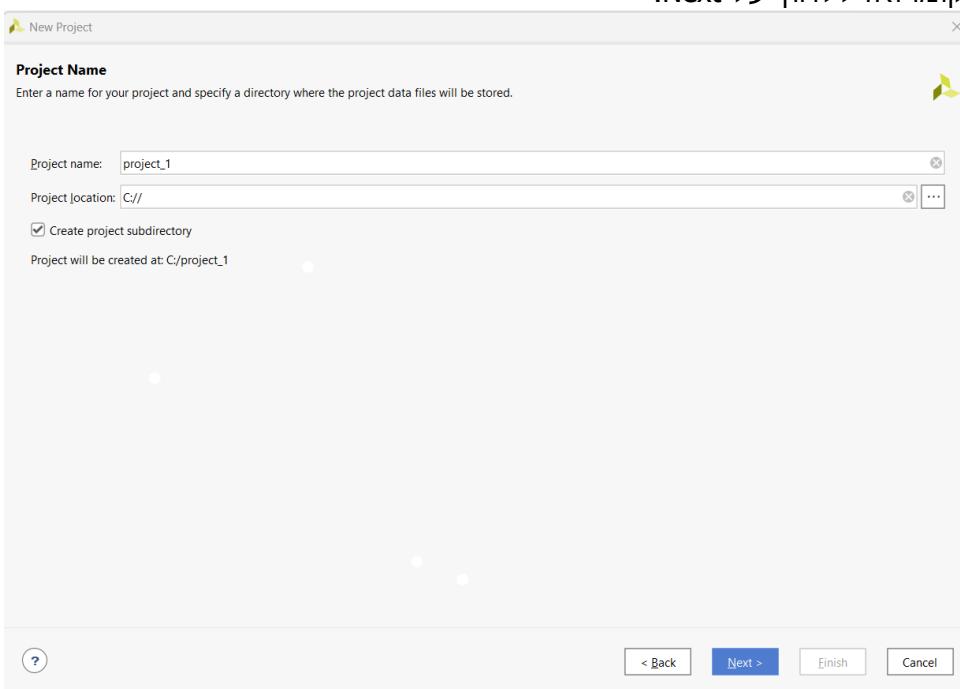


איור 4.1 – חילון הפתיחה של כלי התוכנה OS VIVADO.

- בחילון ה-A Create a New Vivado Project המופיע באירור 4.2, יש ללחוץ על Next.

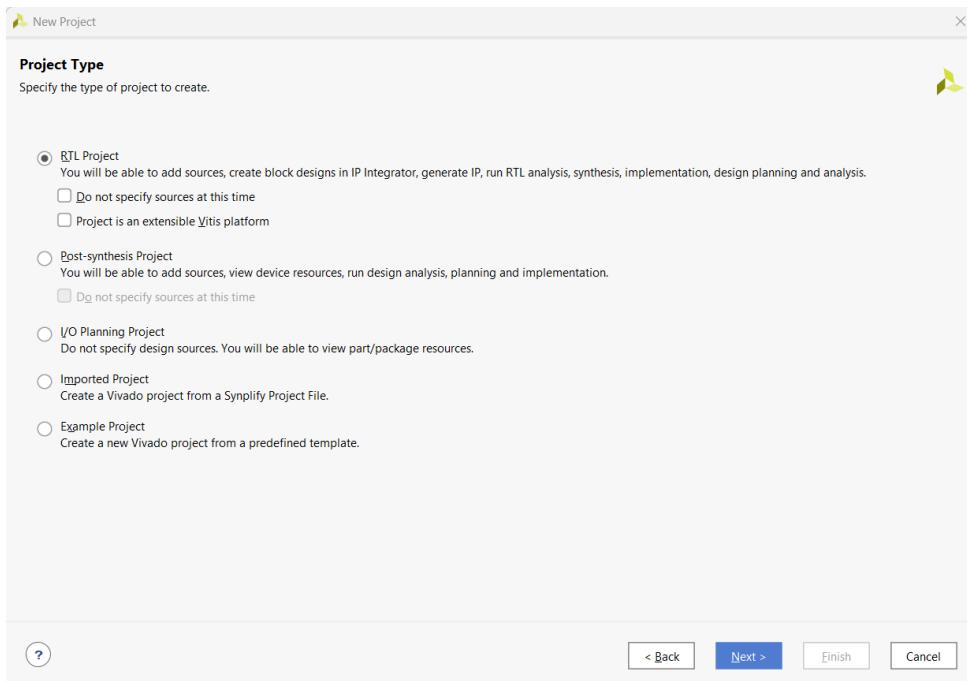


.איור 4.2 – חלון ה-Project Name המופיע באיור 4.3, יש לבחור את שם הפרויקט ואת מיקומו ואז ללחוץ על .Next



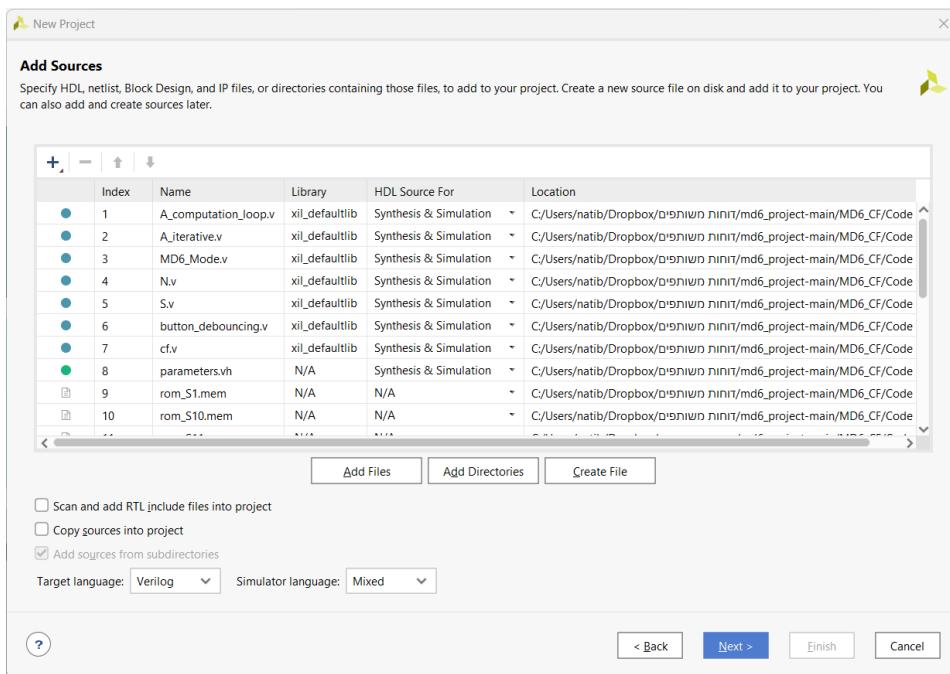
.איור 4.3 – חלון ה-Project Name המופיע באיור 4.4, יש לסמן את RTL Project Type וללחוץ על .Next

- בחילון ה-Project Type המופיע באיור 4.4, יש לסמן את RTL Project Type וללחוץ על .Next



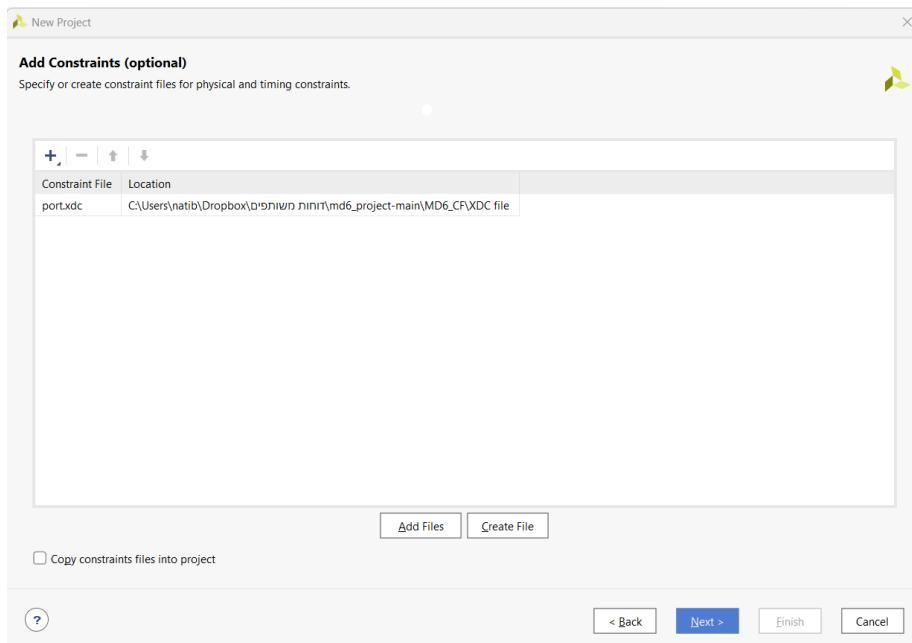
.איור 4.4 – חלון ה-Project Type

- בחלון ה-Add Sources המופיע באיור 4.5, יש ללחוץ על Add Files ולהעלות את קבצי ה-Source של תכנון האלגוריתם הנמצאים ב-- md6_project/main\MD6_CF_hardware\Code files בתיקיית הפרויקט, ולבסוף יש ללחוץ על .Next



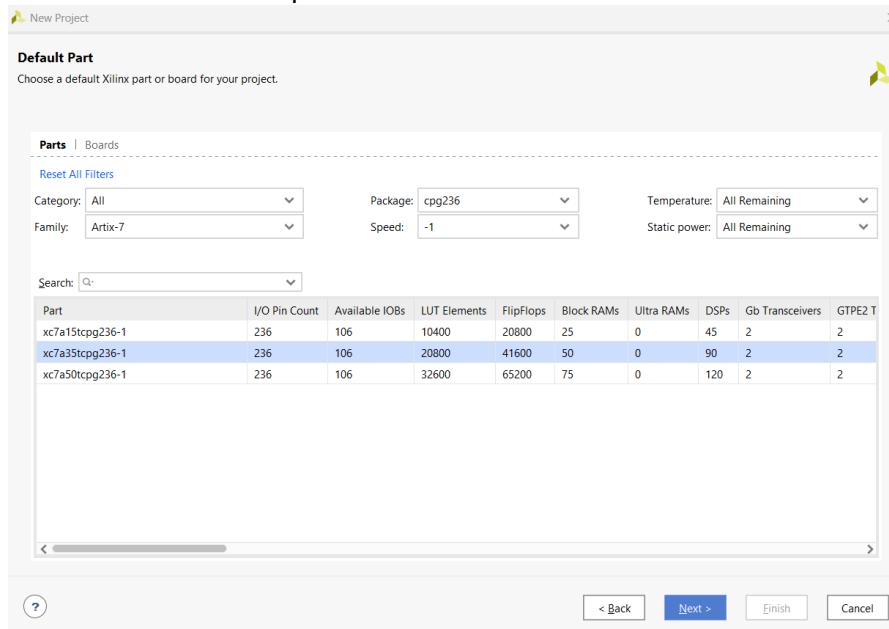
.איור 4.5 – חלון ה-Add Sources

- בחלון ה-Add Constraints המופיע באיור 4.6, יש ללחוץ על Add File ולהעלות את קובץ ה-XDC של תכנון האלגוריתם הנמצא ב-- md6_project/main\MD6_CF_hardware\XDC file בתיקיית הפרויקט, ולבסוף יש ללחוץ .Next



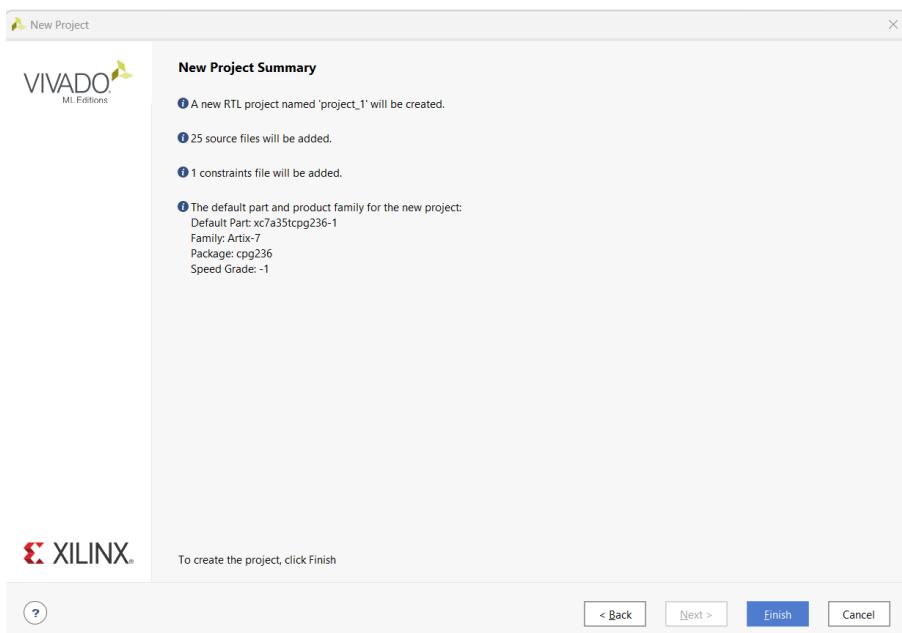
.איור 4.6 – חלון ה-Add Constraints

- בחלון ה-Part המופיע באיור 4.7, יש לבחור ב-Part את רכיב ה-Default Part המתאים – Artix-7 XC7A35T-1CPG236C וائز יש ללחוץ על Next.



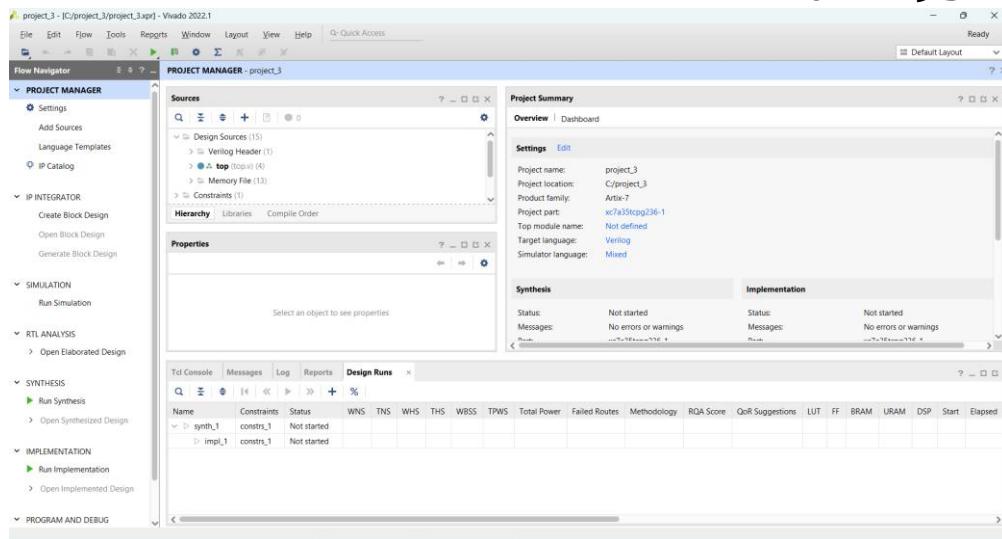
.איור 4.7 – חלון ה-Default Part

- בחלון ה-New Project Summary המופיע באיור 4.8, יש לבדוק שכל הקבצים הועלו ושהרכיב הנכון נבחר וائز יש ללחוץ על Next.



איור 4.8 – חלון New Project Summary.

התוכנית מייצרת את הפרויקט ופותחת את החלון הראשי של ניהול הפרויקט המופיע באיור 4.9.



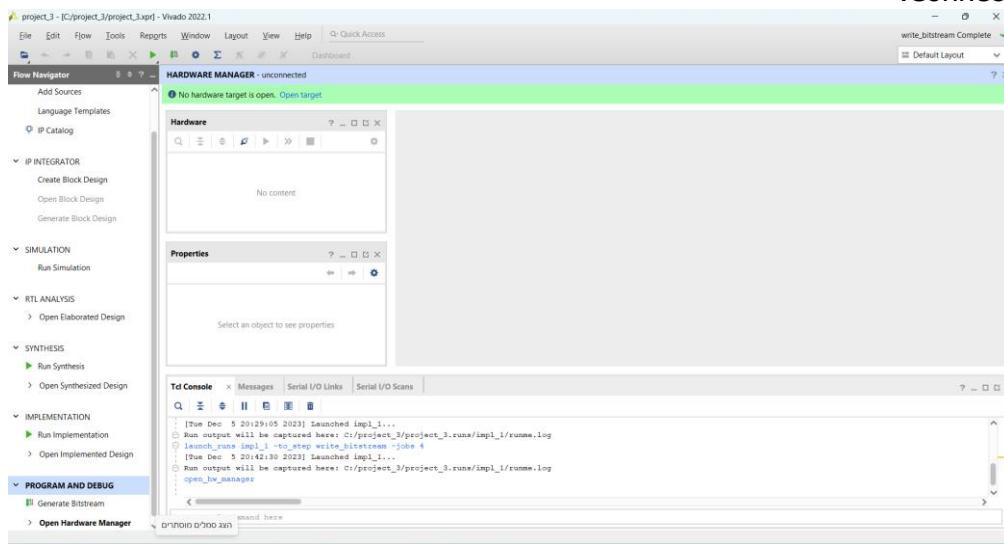
איור 4.9 – החלון הראשי של ניהול הפרויקט.

- יש להפעיל תחילה את הסינתזה ע"י לחיצה על Run Synthesis, שנמצא תחת הכוורת SYNTHESIS ב-PROJECT MANAGER.
- אחרי סיום תהליך הסינתזה, יש להפעיל את תהליך ה"מיקום וחיזוק" ע"י לחיצה על Run Implementation, שנמצא תחת הכוורת IMPLEMENTATION ב-PROJECT MANAGER.
- אחרי סיום המיקום וחיזוק, יש לייצר את קובץ Bitstream ע"י לחיצה על Generate Bitstream, שנמצא תחת הכוורת PROGRAM AND DEBUG ב-PROJECT MANAGER.
- אחרי סיום ייצור קובץ Bitstream, כדי להחיל את התומעה על רכיב Open Hardware Manager ע"י לחיצה על Open Hardware Manager.



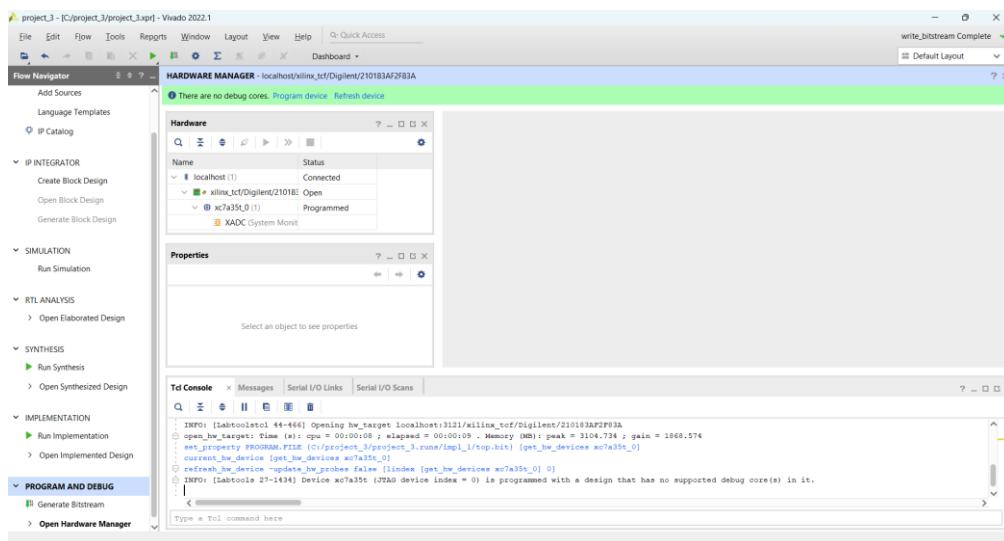
PROJECT - PROGRAM AND DEBUG Manager .MANAGER

- לאחר פתיחת חלון ה-HARDWARE MANAGER המופיע באIOR 4.10, כדי להתחבר עם כל הhardware בו מותם התכנון, יש ללחוץ על Open target הנמצא תחת כותרת החלון (בשורה הירוקה), ואז בחלון הקטן שיפתח יש ללחוץ על Connect.



αιור 4.10 – חלון ה-HARDWARE MANAGER

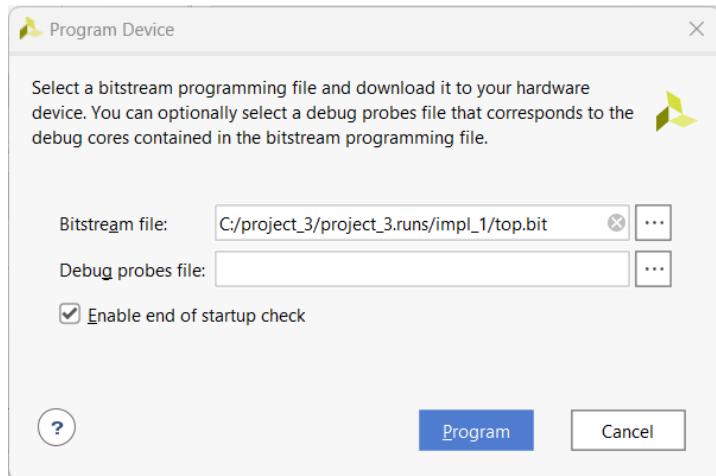
- לאחר שכל התוכנה מצא את הרכיב המועד להטמעה, יש ללחוץ על Program הניתן תחת כותרת חלון ה-HARDWARE MANAGER מבוטע באIOR 4.11.



αιור 4.11 – חלון ה-HARDWARE MANAGER לאחר החיבור לרכיב החומרה.



- לאחר הלחיצה על Program device יפתח חלון בו מופיע קובץ Bitstream המועד להטמעה על רכיב החומרה כמפורט באירור 4.12. יש ללחוץ על Program לצורך להטמעה.



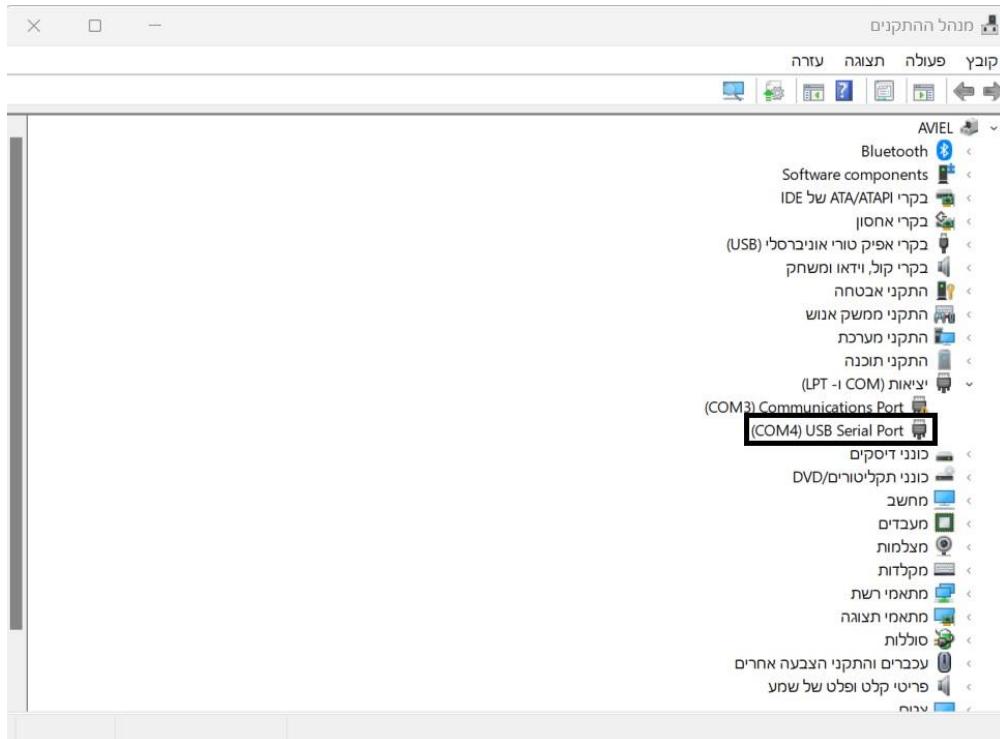
איור 4.12 – חלון ה-Program Device להטמעת התוכנו.

- כעת תוכנו אלגוריתם MD6 הוטמע בלוח החומרה ונitin לגיבוב דרכו מידע.

4.1.2 תהליך הפעלת גיבוב המידע על רכיב ה-FPGA

פרק זה מפרט את תהליך גיבוב אלגוריתם MD6 על רכיב ה-FPGA.

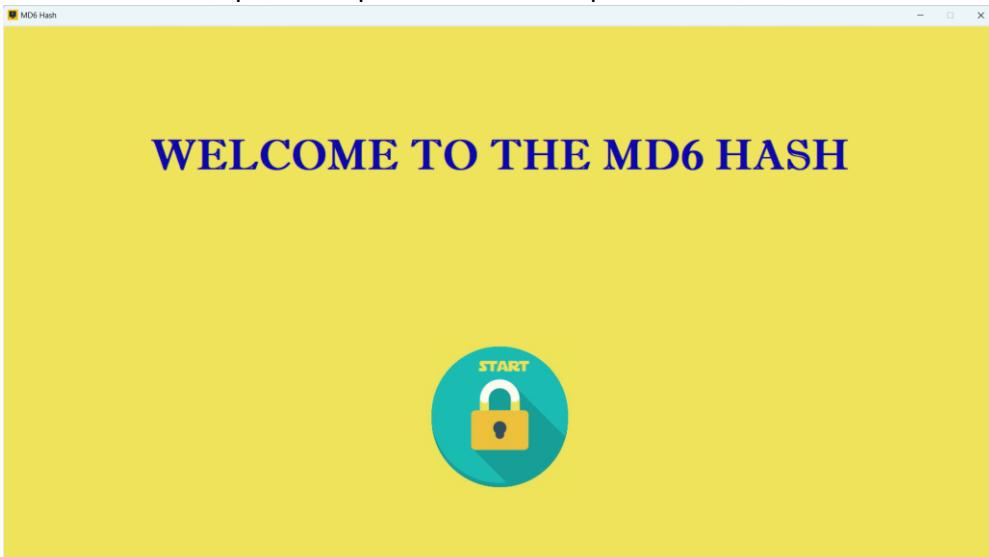
- קודם החלת הגיבוב יש לבדוק את מספר יציאת ה-**COM** בו מחובר רכיב החומרה. ניתן לבדוק זאת במנהל ההתקנים, כמו צג באירור 4.13.



איור 4.13 – מציאת מספר COM במנהל ההתקנים.

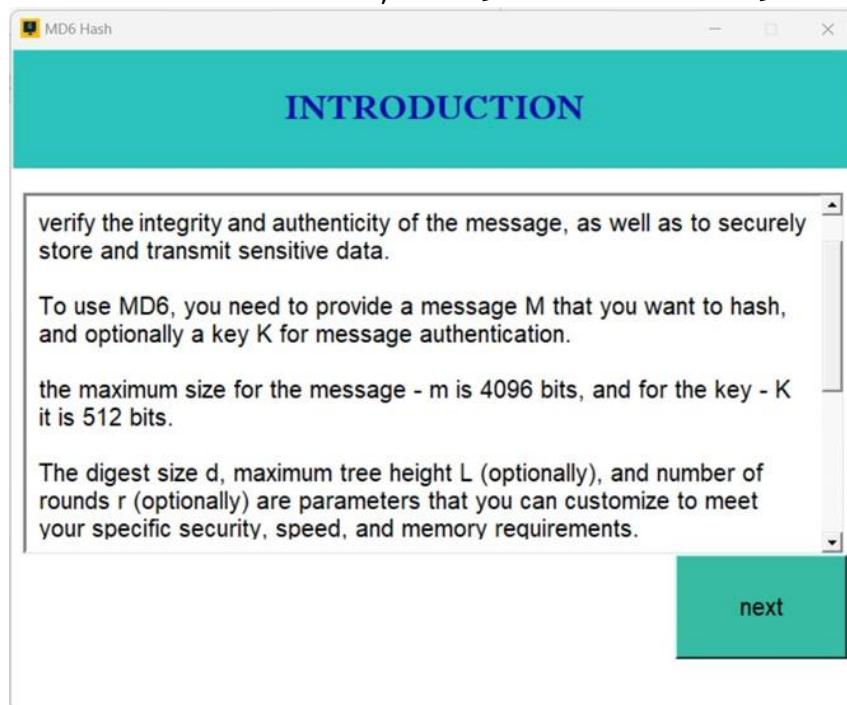


- יש ללחוץ על לחץ reset (BTNU) ברכיב החומרה כדי לאפס אותו.
- להפעלת תהליך הגיבוב יש לפתוח את תוכנית הגיבוב MD6_CF.exe אשר נמצאת ב-App\Executable files\MD6_CF_exe\GUI App md6_project-main.
- לאחר שהקובץ MD6_CF.exe הופעל, יפתח חלון ראשי של התוכנית כמפורט באירור 4.14. כדי להתחליל בתהליך הגיבוב יש ללחוץ על לחץ START.



איור 4.14 – חלון ההתחלה של תוכנית הגיבוב.

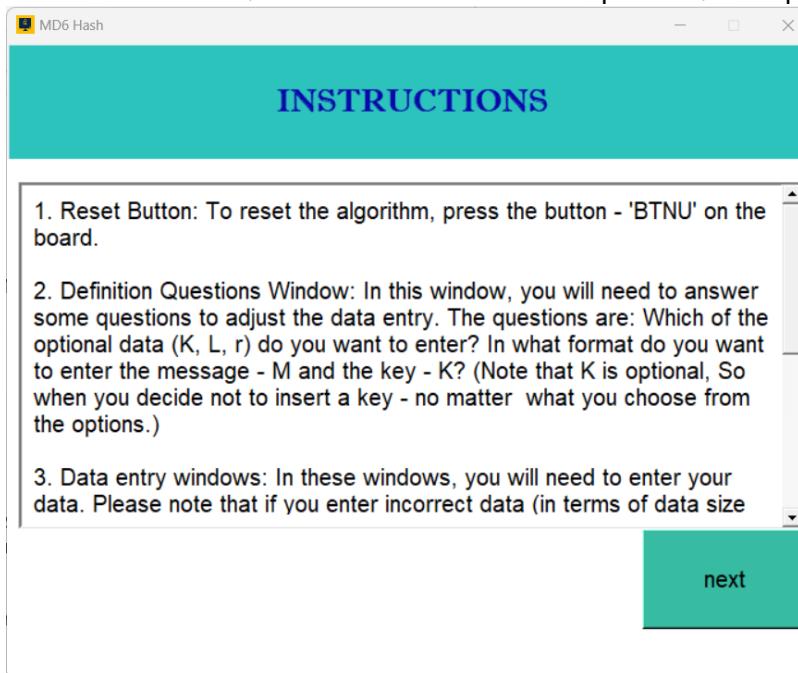
- לאחר לחיצה על כפתור START, יפתח חלון INTRODUCTION אשר מפרט על הגדלים ועל הפורמט של המידע הנכנס, כמו באירור 4.15.



איור 4.15 – חלון INTRODUCTION (המבוא).

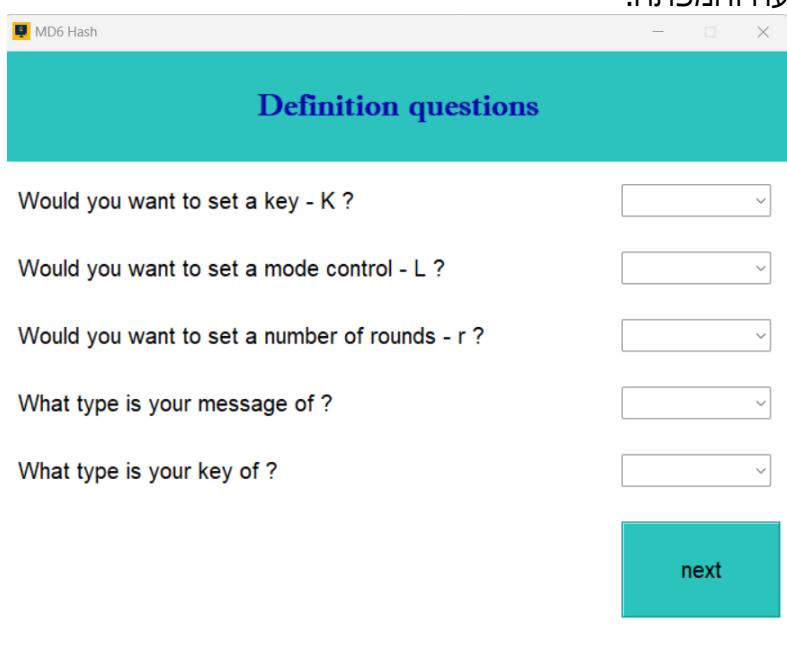


- לאחר לחיצה על כפתור ה-next, החלון הבא הוא חלון ה-INSTRUCTIONS בו מופיע בקצרה על תחילת הגיבוב בתוכנית זו כמפורט באירור 4.16.



איור 4.16 – חלון ה-INSTRUCTION (ההוראות).

- לאחר לחיצה על כפתור ה-next, החלון הבא הוא חלון ה-Definition questions, כמו צג באירור 4.17, שבו נבדק אם נדרש מידע אופציוני ובאיזה פורמט להכניס את ההודעה והמפתח.

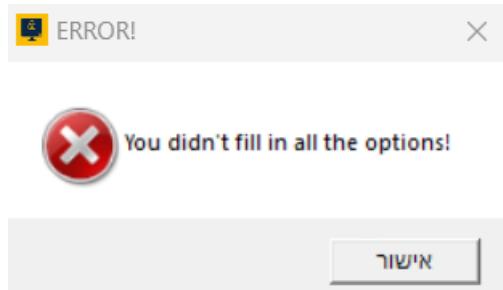


איור 4.17 – חלון ה-Definition questions (שאלות ההגדלה).

- לפני לחיצה על כפתור ה-Next, יש למלא את התשובות לכל שאלות ההגדלה, אחרית ישלח חלון שגיאה, כמו צג באירור 4.18.

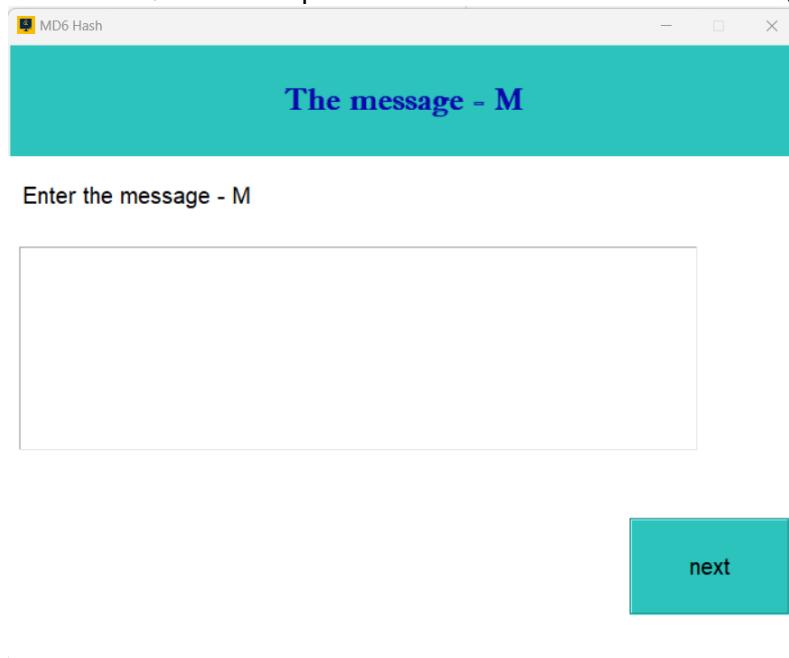


הודעת שגיאה כזו תשלח בכל פעם שלא יבחרו האופציות האפשרות בחלון מסויים.



איור 4.18 – חלון השגיאה על אי מילוי תאים.

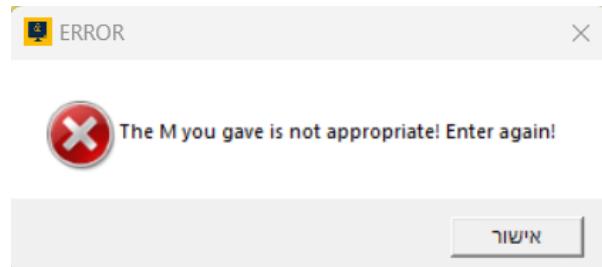
- לאחר מילוי כל השאלות ולחיצה על לחץ ה-next, יפתח חלון ה-Message (ההודעה) כמו צג באיור 4.19. בשלב זה יש להזין את ההודעה בפורמט הרצוי.



איור 4.19 – חלון ה-Message (ההודעה).

- יש לשים לב – אם ההודעה אינה בגודל הנכון או אינה בפורמט הנכון. התוכנית תשלח הודעת שגיאה כמו צג באיור 4.20.

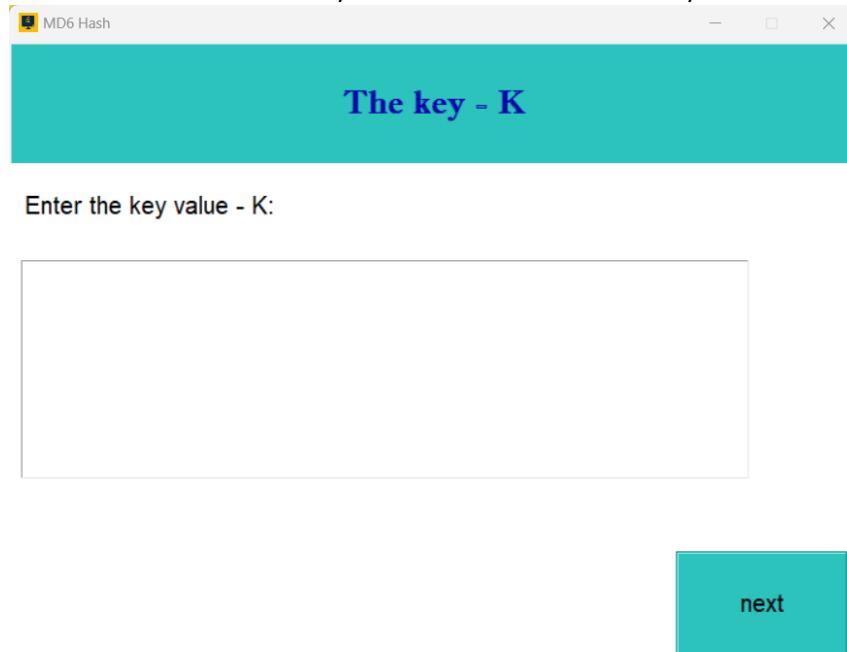
הודעת שגיאה כזו תשלח בכל פעם שלא נבחרות האופציות האפשרות בחלון מסויים.



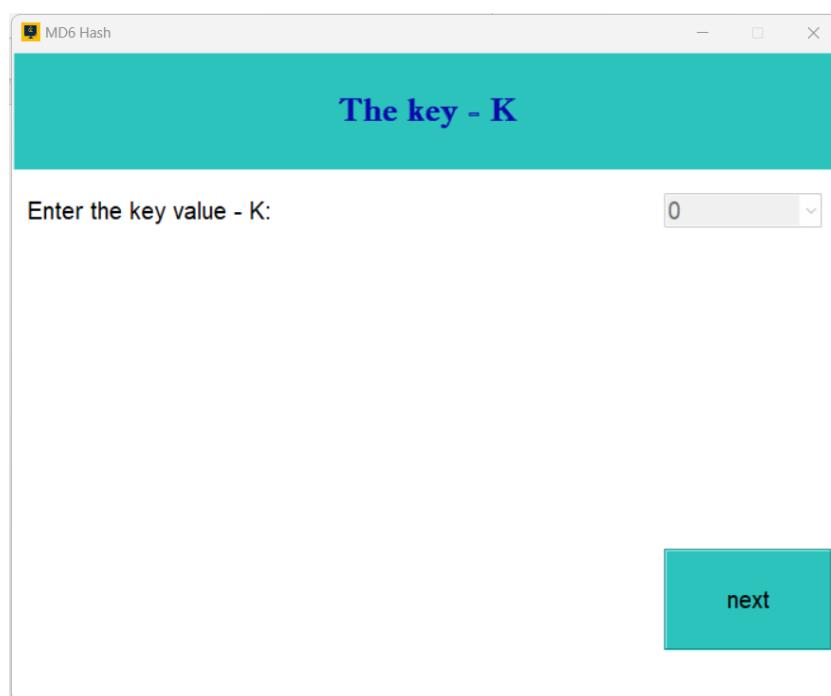
איור 4.20 – חלון השגיאה על הכנסת מידע לא תואם.



- לאחר מלאי הודעה ולחיצה על לחצן next, יפתח חלון Key. במידה ויסופק מפתח, יפתח חלון שבו יוכל לספק מפתח, כמו חלון Message, כמפורט באירור 4.21. ולא, יפתח חלון שמרתה את ערך בריית המודול של המפתח השווה ל-0, ללא אפשרות לשנות אותו, כמפורט באירור 4.22.



איור 4.21 – חלון Key (המפתח) עם תיבת הכנסת המידע.



איור 4.22 – חלון Key (המפתח) עם ערך בריית המודול.

- לאחר לחיצה על לחצן next, יפתח חלון Key, כמו בתמונה 4.23, בו יש לספק את אורך המידע המגווב, פרמטר בקרת המצביע ומספר הסיבובים.



פרמטר בקורת המצב ומספר הסיבובים הם כניסה אופציונליות, لكن במידה ולא יספקו, יופיע ערך ברירת המחדל שלהם, ללא אפשרות שינוי. לדוגמה, כדי שניתן לראות באIOR 4.23, לא סופק פרמטר בקורת המצב, אשר על כן מופיע ערך ברירת המחדל שלו, ללא אפשרות שינוי.

MD6 Hash

d & L & r

Choose the length of the hashed port - d

Choose the model control - L

Choose the number of rounds - r

next

.AIOR 4.23 – חלון ה-d & L & r.

- לאחר הכנסת המידע ולהחיצה על לחצן ה-next, יפתח חלון ה-COM, בו יש לספק את מספר ה-COM אליו מחובר רכיב החומרה, כמו צג באIOR 4.24.

MD6 Hash

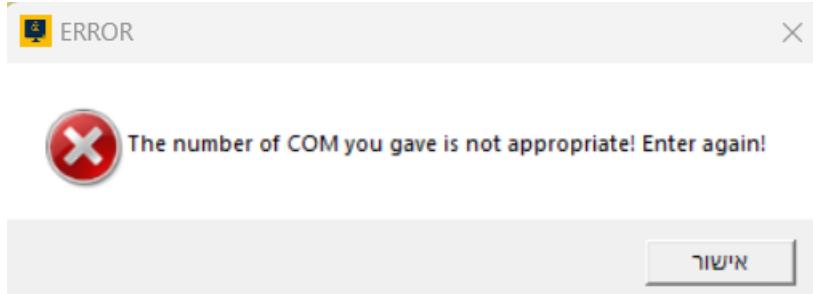
COM number

Enter your com - COM{number}

transmit

.AIOR 4.24 – חלון ה-COM number.

- כאשר נבחר מספר COM לא מתאים, ישלח חלון שגיאה כמו זה באיר 4.25.



איור 4.25 – חלון השגיאה על הכנסת מספר COM שגוי.

- לאחר הכנסת מספר COM ולחיצת על כפתור ה-transmit, המידע מועבר לגיבוב ברכיב החומרה.
- Basys3 כולל נורות LED המאפשרות לתת אינדיקציה לתהליכי שמרתחשים בלוח. נורות LD0 - LD8 מורות על חלקים של מידע שנקלט, כך שהדלקה של כלל הנורות מורה על סיום קליטת המידע. לאחר מכן ניתן לחוץ על לחץ השילוח BTNC אשר שולח את המידע המוגובב חוזה לתוכנית. את מיקום הנורות והלחצנים ניתן לראות באיר 3.25.
- לאחר לחיצה על לחץ השילוח, יפתח חלון - the hashed message אשר מציג את ההודעה המוגובבת, כמו באיר 4.26.



the hashed message

8854c14dc284f840ed71ad7ba542855ce189633e48c797a55121a74
6be48cec8

finish

איור 4.26 – חלון the hashed message (ההודעה המוגובבת).

- לסגירת התוכנית יש לחוץ על לחץ ה-finish.



4.2 אימות תכנון החומרה בסימולציה ModelSim

4.2.1 תהליך אימות תכנון החומרה בסימולציה ModelSim

במסגרת הפרויקט בוצע אימות תכנון החומרה בכלי הסימולציה ModelSim שمدמה את פעילות התכנון על רכיב החומרה. בשביל ליצור הדמיה של הكنيסות והיציאות, קובץ test bench עוטף את קוד החומרה ובו ננסים הרצויים בקלטיו התכנון. הסימולציה כוללת את החלקים הבאים:

- :Test Vector Sim.py**

קוד ה-Python מייצר מידע אקראי, מעביר אותו לפורמט הקסדצימלי ומשרשר אותו אחד לשני. לאחר מכן הוא מעביר את המידע לקובץ טקסט בשם `input` `.data.txt`.

בנוסף, הקוד מכיל מימוש של אלגוריתם MD6 בתוכנה, ואת המידע האקראי שנוצר הוא מעביר במירך של `bytes`. לאחר מכן ה-`tb` מפעיל לולאה שמעבירה כל byte ממיעד הקלט לתכנון נציגה, כמפורט באירור 4.27.

```

task send_byte ();
begin
    #104166 tb_RxD = 0;

    #104166 tb_RxD = data[count][0];
    #104166 tb_RxD = data[count][1];
    #104166 tb_RxD = data[count][2];
    #104166 tb_RxD = data[count][3];
    #104166 tb_RxD = data[count][4];
    #104166 tb_RxD = data[count][5];
    #104166 tb_RxD = data[count][6];
    #104166 tb_RxD = data[count][7];

    #104166 tb_RxD = 1;
end
endtask

```

איור 4.27 – שליחת המידע האקראי דרך ה-`RxD`.

לאחר סיום הגיבוב, ה-`tb` מקבל בחזרה את המידע המגובב ושומר אותו במערך בהתאם, כמפורט באירור 4.28.



```

task get_byte();
begin
    #104166 temp[0] = tb_TxD;
    #104166 temp[1] = tb_TxD;
    #104166 temp[2] = tb_TxD;
    #104166 temp[3] = tb_TxD;
    #104166 temp[4] = tb_TxD;
    #104166 temp[5] = tb_TxD;
    #104166 temp[6] = tb_TxD;
    #104166 temp[7] = tb_TxD;
    #104166 temp[8] = tb_TxD;
    #104166 temp[9] = tb_TxD;
end
endtask

```

איור 4.28 – קבלת המידע המגווב דרך ה-TxD.

לסימן, ה-tb מדפיס את הגיבוב בתוכנה ובחומרה לשם השוואה.

4.2.2 הפעלת אimoto תכנון החומרה בסימולציה ModelSim

פרק זה מפרט את תהליך הפעלת אimoto תכנון החומרה בסימולציה ModelSim של אלגוריתם MD6.

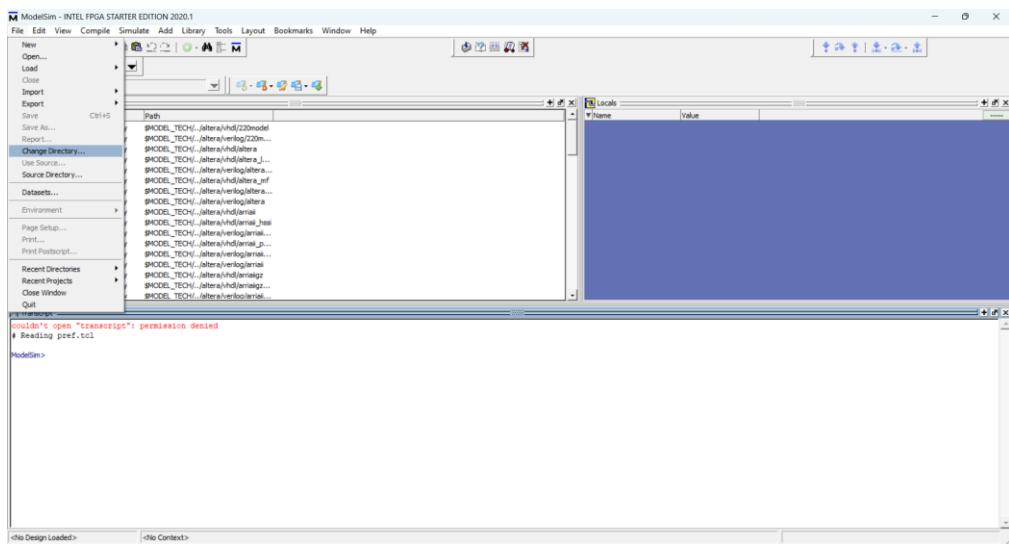
כדי שיהיה ניתן להפעיל את האלגוריתם, יש לוודא שהדברים הבאים נמצאים בהישג יד:

- תקינות הקבצים שלuproject.
- כל התוכנה ModelSim.
- שפת Python וסביבה מותאמת להפעלה.

קישור לתקינות קבציuproject להפעלת האלגוריתם מצורפים [בנספח א](#).

שלבי הפעלת סימולציה אimoto תכנון החומרה בסימולציה ModelSim:

- תחיליה יש להעביר את הקבצים הניצרכים לSIMULATION לתקינה אחת
 - קבצי Source הנמצאים ב--.md6_project\main\MD6_CF_hardware\Code files
 - קובץ ה-TB הנמצא ב--.md6_project\main\MD6_CF_hardware\Test.Bench
 - קובץ ה-Python Test_Sim.py הנמצא ב--.md6_project-main\Test_Vector_Sim.py Python Executable files\Test_vector_sim
- יש לפתח את קובץ Test_Vector_Sim.py Python, Test_Vector_Sim.py, ולהפעיל אותו. הקובץ מייצר את 2 קבצי ה-tekסט בהם משתמש קובץ ה-test bench: מידע קלט אקראי לחומרה ותוצאת הגיבוב בתוכנה.
- יש להפעיל את כל התוכנה ModelSim.
- יש ללחוץ על Change directory אשר נמצא תחת הכוורת file כמפורט באיוור 4.29, ויש לשנות את מקום התקינה לתקינה בה הועברו את הקבצים.



איור 4.29 – Change directory בкли התוכנה ModelSim.

- ב-transcript נוצר תקית עובדה ע"י שימוש בפקודה הבאה:

vlib work

- לאחר יצירת התקיה יש לкомפל את קבצי ה-Source ע"י שימוש בפקודה הבאה:

vlog N.v MD6_Mode.v cf.v

בתיקית קבצי הפרויקט מצורף קובץ טקסט בשם txt.txt אשר מכיל את הפקודות הרלוונטיות יחד עם כל קבצי ה-Source.

יש לשים לב שאין צורך לкомפל את קבצי ה-mem ואת קובץ ההידור (h), אלא רק לוודא שיימצאו באותה התקיה.

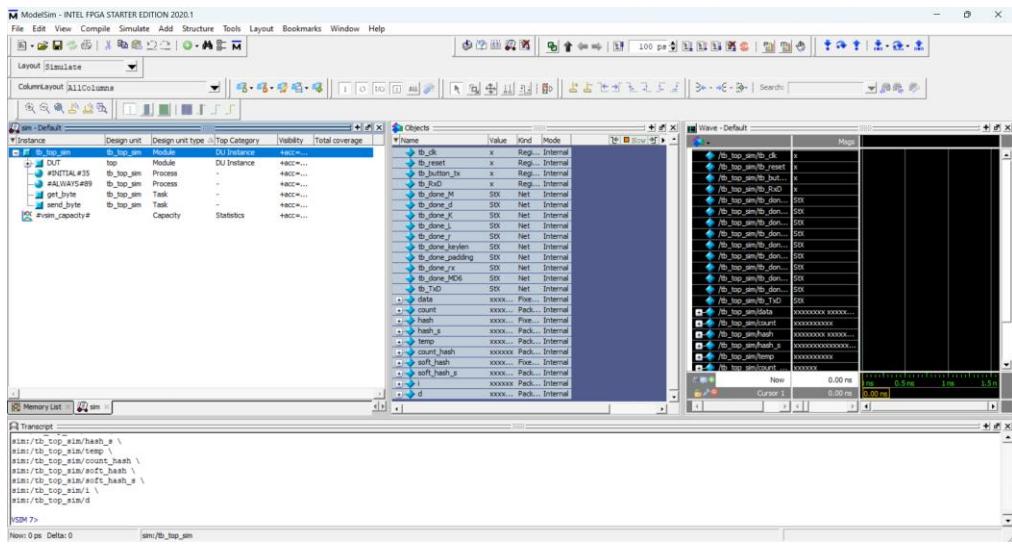
- לאחר שהקימפול של קבצי ה-Source עבר בהצלחה, יש לкомפל את קובץ ה-**:test bench**

vlog tb_top_sim.v

לאחר שקימפול קובץ ה-testbench עבר בהצלחה, יש להפעיל את הסימולציה ע"י הפקודה הבאה:

vsim tb_top_sim

- לאחר הפעלת פקודה ה-vsim יפתח החלון הראשי של הסימולציה כמווצג באיור 4.30



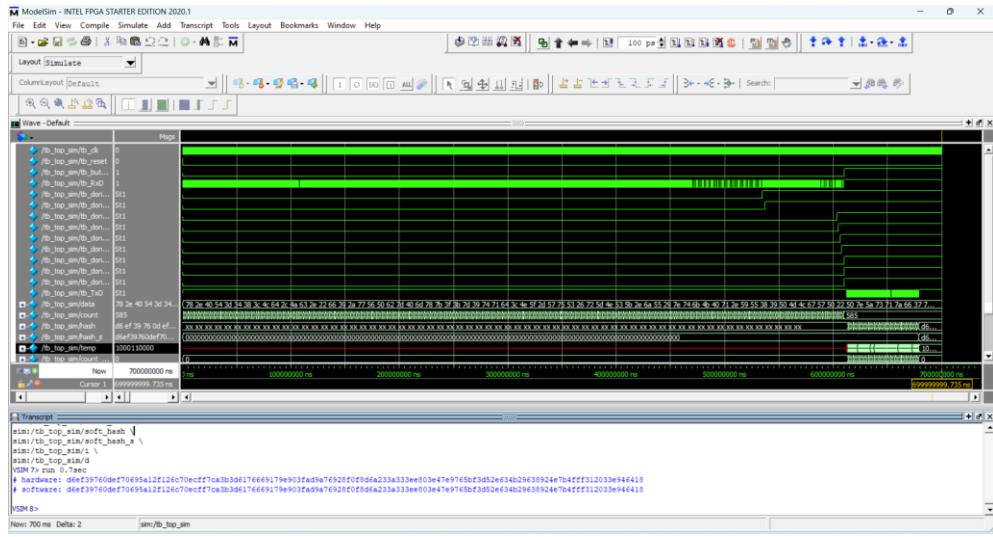
איור 4.30 – החלון הראשי של הסימולציה בכליה התוכנה ModelSim.

כדי לראות את צורת הגלים של האותות, יש לבחור בחלון ה-Object את האותות הרצויים ולאחר מכן יש ללחוץ מקש ימני ולבחר Add Wave (או בקיצור **w**. (Ctrl + w).

- להרצת הסימולציה יש לכתוב בשורת Transcript את הפקודה run ואת זמן הריצה הרצוי.
זמן הריצה הנדרש לסיום כל הסימולציה הינו לכל היוטר 0.7sec. אי לכך יש להריץ את הפקודה הבאה:

run 0.7sec

אחרי סיום הרצת הסימולציה יתקבלו במסמך ה-Transcript את תוצאות הגיבוב בחומרה ובתוכנה אחד אחרי השני כמפורט באירור 4.31.



איור 4.31 – תוצאות הגיבוב ביחד עם צורת הגלים של האותות הסופי.

4.3 אימות תכnon החומרה על ערכות הפיתוח

4.3.1 תהליכי אימוט תכנון החומרה על ערכת הפיתוח

כדי לאמת את התוכן על רכיב החומרה, קודה תוכנה בשפת Python בשם `Test vector` שתפקידיה ליצור וכתורו בדיקה. התוכנה יוצרת מידע אקראי (תואם לקריטריונים



של האלגוריתם) שעובר גיבוב בתוכנו התוכנה ובתוכנו החומרה. לאחר מכן מכון התוכנה משווה בין המידע המגובב ובודקת שהם שווים. תהליך זה מתבצע עבור כל וקטורי הבדיקה שבחר המשתמש להכניס, ולאחר מכן היא מייצרת קובץ אקסל שמכיל את המידע המגובב וההשוואה. ובכך תהליך זה נועד לאמת את נכונות הגיבוב. כמו כן, ניתן לשווות את הביצועים של כל התוכנות.

שימוש בקוד ה-[Python](#) להפעלת התוכנית דרוש להתקין Python Python ועוד ספריות נוספות שנדרכות לצורך קומפקטת הקוד. לשם כך נוצר קובץ exe כך שלא נדרש להתקין שום דבר נוספת.

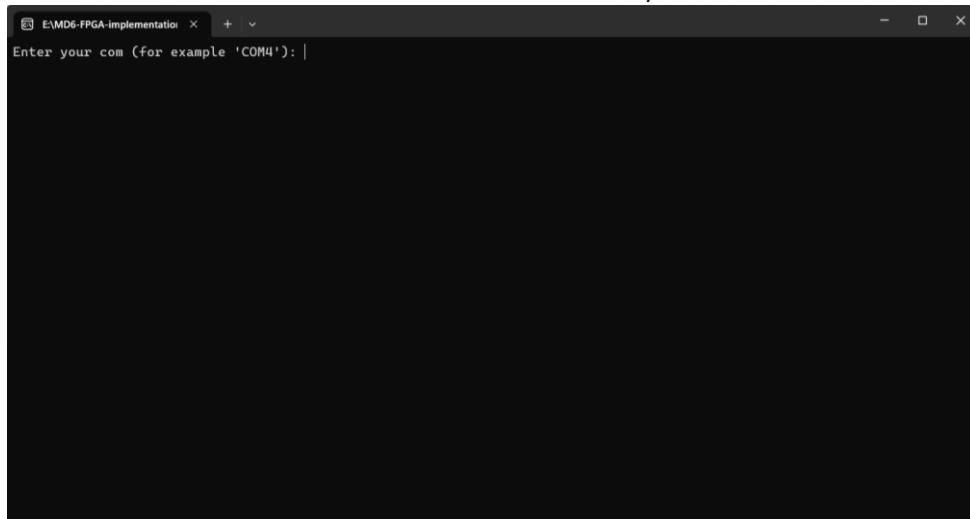
4.3.2 הפעלת אימות תכנון החומרה על ערכת הפיתוח

פרק זה מפרט את תהליך הפעלת אימות תכנון החומרה על ערכת הפיתוח כדי שייהי ניתן להפעיל את האלגוריתם, יש לוודא שהדברים הבאים נמצאים בהישג יד:

- תקינות הקבצים שלuproject.
- כל התוכנה VIVADO.

קישור לתיקיית קבציuproject להפעלת האלגוריתם מצורפים [בנספח א](#).

- בתחילת הפעלה יש להטעין את התוכנו על רכיב החומרה. ניתן למצוא את התהליך המפורט [בפרק 4.1.1](#).
- בנוסף, יש לבדוק את מספר יציאת COM בו מחובר רכיב החומרה במנהל ההתקנים, כמו צג באIOR 4.13.
- יש להפעיל את קובץ הפעלת אימות תכנון החומרה על ערכת הפיתוח md6_project-main\Software\Test_Vectors.exe App GUI בספרייתuproject.
- לאחר הריצת קובץ הפעלה יפתח חלון והתוכנה תבקש להכניס את כניסת ה- COM אליה מחובר המכשיר, כמו צג באIOR 4.32.



איור 4.32 – הכנסת מספר כניסת COM.

- אחרי הכנסת מספר COM, התוכנה תבקש להכניס את מספר וקטורי הבדיקה, כמו צג באIOR 4.33.



```
E:\MD6-FPGA-implementation x + 
Enter your com (for example 'COM4'): COM4
Select the number of test vectors you want to test: 5
```

איור 4.33 – הכנסת מסטר וקטורי הבדיקה.

- לאחר הכנסת מסטר וקטורי הבדיקה, התוכנה תחשב את הוקטור הראשוני בתוכנה ותזכיר לשילחה את אותו הוקטור לחישוב בחומרה, כמו צג באיור 4.34.

```
E:\MD6-FPGA-implementation x + 
Enter your com (for example 'COM4'): COM4
Select the number of test vectors you want to test: 5
SOFTWARE: 0x92988d3f83daf65d73479859b23fdbba6ee1d5a19987d2c04dd57a89e3668e2bce717a5b0268a4b13dacf9a0c9d600a3c49852b4667
effacabe6765ee5308dbe
reset! |
```

איור 4.34 – תצוגת הוקטור הראשוני המגובב בתוכנה.

- בשביל לקבל את גיבוב החומרה בתכונן, יש ללחוץ תחילה על כפתורו ה-`reset` (BTNU) ולאחריו `enter`. כאשר כל 9 הנורות יידלקו, יש ללחוץ על כפתורו שליליה (BTNC) ויתקבל המידע המגובב. ניתן לראות את מיקומם באיור 3.25. בנוסף, בשורה מתהווה מתקובל גיבוב של התכונן בתוכנה, כמו צג באיור 4.35.



```

E:\MD5-FPGA-implementation x + x
Enter your com (for example 'COM4'): COM4
Select the number of test vectors you want to test: 5
SOFTWARE: 0x92988d3f83daf65d73479859b23fdbba6ee1d5a19987d2c04dd57a89e3668e2bce717a5b0268a4b13dacf9a0c9d600a3c49852b4667
effacabe6765ee5308dbe
reset!
HARDWARE: 0x92988d3f83daf65d73479859b23fdbba6ee1d5a19987d2c04dd57a89e3668e2bce717a5b0268a4b13dacf9a0c9d600a3c49852b4667
effacabe6765ee5308dbe
SOFTWARE: 0xd0e5b3c068453fe5096ce887da02c1e6db5d78681c6cf90a5491f0625dc9f2006891e09957740174b8ede2a2766e78e
reset!

```

איור 4.35 – תצוגת וקטור החומרה שהתקבל.

- לאחר סיום קבלת כל הוקטורים, המסר יסגר והתוכנה תיצור קובץ csv שימושו את הוקטורים בתוכנה ובחומרה וקובע אם הם שווים (false או true), כמו צג באיור 4.36.

| | A | B | C | D | E | F | G |
|----|----------|---|----------|-------------------|---|---|---|
| 1 | Vector | Software | Hardware | Comparison Result | | | |
| 2 | vector-1 | 0x92988d3f83daf65d73479859b23fdbba6ee1d5a19987d2c04dd57a89e3668e2bce717a5b0268a4b13dacf9a0c9d600a3c49852b4667 | | TRUE | | | |
| 3 | vector-2 | 0xd0e5b3c068453fe5096ce887da02c1e6db5d78681c6cf90a5491f0625dc9f2006891e09957740174b8ede2a2766e78e | | TRUE | | | |
| 4 | vector-3 | 0x6119d4a0x6119d4a | | TRUE | | | |
| 5 | vector-4 | 0x36013f70x36013f7 | | TRUE | | | |
| 6 | vector-5 | 0x547eae80x547eae8 | | TRUE | | | |
| 7 | | | | | | | |
| 8 | | | | | | | |
| 9 | | | | | | | |
| 10 | | | | | | | |
| 11 | | | | | | | |
| 12 | | | | | | | |
| 13 | | | | | | | |

איור 4.36 – קובץ csv להשוואה בין התוצאות.



5 דינמים

בפרק זה נשווה בין תכנון אלגוריתם MD6 שתוכנן במסגרת הפרויקט, לבין התוצאות בדו"ח האלגוריתם של יוצר האלגוריתם Ronald Linn Rivest [3].
יתר על כן, נשווה בין ייעילות תכנון האלגוריתם בחומרה ובתוכנה.

5.1 השוואת ספרות

בפרק זה נשווה את תכנון הפרויקט לדו"ח אלגוריתם MD6 של Ronald Linn Rivest [3].
משום שהתכנון מכיל את אלגוריתם MD6 עם פונקציית דחיסה היחידה, מתוך שלושת הדוגמאות הניתנות בדו"ח, רק אחת מתאימה לפונקציה דחיסה אחת והיא הדוגמא הראשונה.

Ronald Linn Rivest קלט האלגוריתם לפי הדוגמא הראשונה בדו"ח האלגוריתם של מופיע באIOR 5.1.

```
> md6sum -r5 -I -Mabc
-r5
-- Mon Aug 04 18:28:03 2008
-- d =      256 (digest length in bits)
-- L =       64 (number of parallel passes)
-- r =        5 (number of rounds)
-- K =      '' (key)
-- k =        0 (key length in bytes)
```

איור 5.1 – מידע הקלט של הדוגמא הראשונה [3].

תוצאת הגיבוב של האלגוריתם לפי הדוגמא הראשונה בדו"ח מופיעה באIOR 5.2.

8854c14dc284f840ed71ad7ba542855ce189633e48c797a55121a746be48cec8 -Mabc

The final hash value is 0x8854c14d...cec8 .

איור 5.2 – תוצאת הגיבוב של הדוגמא הראשונה [3].

כדי לסכם בΖΟΡΑ יותר ברורה מידע הקלט ותוצאת הגיבוב מוצגים בטבלה 23.

טבלה 23: המידע הנכנס לאלגוריתם בסימולציה.

| המידע | סוג המידע |
|--|-------------------------|
| Abc | ההודעה – M |
| None | המפתח – K |
| 256 | אורך ההודעה המוגבהת – d |
| 64 | מצב הפעולה – L |
| 5 | מספר הסיבובים – r |
| 8854c14dc284f8 40ed71ad7ba542 855ce189633e48 c797a55121a746 be48cec8 | ההודעה המוגבהת – H |



5.1.1 השוואת תוצאות אימות תכנון החומרה בסימולציה ב-ModelSim

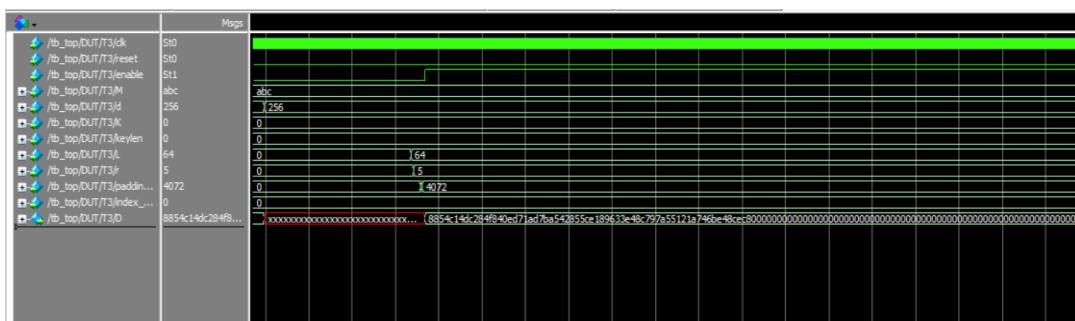
קובץ ה-.vtb המשמש לבדיקת הדוגמא הראשונה הוא קובץ ה-.vtb_top אשר נמצא ב-[MD6_project-main\MD6_CF\Test Bench](#)

קובץ ה-.vtb_top עוטף את DUT (Device Under Test) ומעביר את המידע מסודר ומורופד באפסים כמו שהוא צריך להיות מועבר ברכיב חומרה.

בנוסף, משומם שהמידע מועבר באמצעות תקשורת טורית – UART, ה-uart test bench תוכנן כך שהמידע נכנס סיבית אחר סיבית כאשר כל byte עוטף בסיבית התחלת וסיבית עצירה, כמו פיעם באירור 5.1.

```
//1: "a"
#104166 tb_RxD = 0;
#104166 tb_RxD = 1;
#104166 tb_RxD = 0;
#104166 tb_RxD = 0;
#104166 tb_RxD = 0;
#104166 tb_RxD = 1;
#104166 tb_RxD = 1;
#104166 tb_RxD = 0;
#104166 tb_RxD = 1;
```

איור 5.3 – תוצאת הגיבוב של הדוגמא הראשונה [3].



איור 5.4 – חלון צורת הגלים של הקלט והפלט של הדוגמא הראשונה.

באיור 5.2 מופיע חלון-wave של כלי תוכנת הסימולציה ModelSim אשר מציג את המידע שהתקבל ע"י תקשורת-UART ואת ההודעה המגובבת לפי הדוגמא הראשונה של האלגוריתם. ההודעה וההוודעה המגובבת מוצגים בפורמט הקסדצימלי והשאר בפורמט דצימלי כדי להראות בצורה ברורה את נוכנות המידע.

כפי שניתן לראות מהאיור, התקבלה תוצאה גיבוב נכונה המתאימה בדיקן לדוח'Ronald Linn Rivest

הפעלת סימולציה ב-ModelSim מפורטת צעד אחר צעד [פרק 4.1.2](#)



5.1.2 השוואת תוצאות אימות תכnuן החומרה על ערכות הפתרונות

כדי להשוות בחומרה יש להשתמש בקובץ הפעלה של האלגוריתם MD6.exe ויש להכניס את המידע בצורה הבאה:

- תחילה יש לענות על שאלות ההגדרה בצורה נכונה כך שיתאימו לדוגמא הראשונה בדוח האלגוריתם, כמו באיור 5.5. יש לשים לב שלמורות שבדוגמה אין מפתח, בכל זאת יש לבחור סוג פורטט.

The screenshot shows the 'MD6 Hash' application window. The title bar says 'MD6 Hash'. The main area has a teal header with the text 'Definition questions'. Below it are five questions with dropdown menus:

- 'Would you want to set a key - K ?' with 'NO' selected.
- 'Would you want to set a mode control - L ?' with 'YES' selected.
- 'Would you want to set a number of rounds - r ?' with 'YES' selected.
- 'What type is your message of ?' with 'ascii' selected.
- 'What type is your key of ?' with 'hex' selected.

 At the bottom right is a teal button labeled 'next'.

איור 5.5 – התאמת שאלות ההגדרה לדוגמא הראשונה.

- לאחר מכן, יש להכניס את ההודעה בפורטט ASCII כMOVOKSH, כמו באיור 5.6.

The screenshot shows the 'MD6 Hash' application window. The title bar says 'MD6 Hash'. The main area has a teal header with the text 'The message - M'. Below it is a text input field containing 'abc'. At the bottom right is a teal button labeled 'next'.

איור 5.6 – התאמת ההודעה לדוגמא הראשונה.



- משום שבדוגמה אין מפתח, בחלון המפתח יופיע ערך ברירת המחדל ללא אפשרות לשנותו, כמו צג באיור 5.7.

The key - K

Enter the key value - K: 0

next

. איור 5.7 – התאמת המפתח לדוגמא הראשונה.

- גם בחלון ה- r & L & d , יש לבחור את קלט המידע הנכון לפי הדוגמא הראשונה, כמו צג באיור 5.8.

d & L & r

Choose the length of the hashed port - d 256

Choose the model control - L 64

Choose the number of rounds - r 5

next

. איור 5.8 – התאמת r & L & d לדוגמא הראשונה.



- לאחר התאמת מידע הקלט לדוגמא הראשונה, כפי שניתן לראות באIOR 5.9, akan התקבלה אותה תוצאה גיבוב.

IOR 5.9 – תוצאה הגיבוב של הדוגמא הראשונה בחומרה.

סרטון המראה את הריצת הדוגמא הראשונה של דוח האלגוריתם ניתן למצוא [בנספח א.](#)

5.2 השוואה לתוכנה

כדי להראות את יכולות עיבוד המידע בתוכנו, נשווה את תכנון האלגוריתם בחומרה לתכנון בתוכנה.

במהלך הפרויקט נוצר תכנון שנוצר בתוכנה בשפת Python לאימונות וכוננות תוצאות הבדיקה. אך בדיקת יכולות זו פחות אפקטיבית משום ששפת Python הינה שפה עילית ולכן היא מלכתחילה יותר איטית וקצב העיבוד שלה יהיה נמוך מהרגיל. תכנון אלגוריתם בשפת C יותר קרוב לשפת מכונה וייתר שיר להשוות אותו לתכנון החומרה.

נפרט אודות חישוב קצב עיבוד המידע בחומרה ובתוכנה עבור מספר הסיבובים המקסימלי בתכנון:

- חישוב קצב עיבוד המידע בחומרה:**

כדי לחשב את קצב עיבוד המידע צריך לחשב קודם את התדר המקסימלי של רכיב החומרה בתכנון לפי [משואה 2](#):

(2)

$$\text{max Freq} = \frac{1}{\text{Crystal Oscillator} - WNS}$$

כאשר:

זמן המחזoor של מתנד הקристל של רכיב החומרה השווה ל- 10ns , כמפורט בטבלה 5 [פרק 3.1](#).



(Worst Negative Slack) WNS – מיצג את הנטייב הקריטי, שהוא הנטייב עם מרוחת התזמון הנמוך ביותר. הנטייב הקריטי קובע את התקדים המקסימלית שבה מעגל דיגיטלי יכול לפעול תוך עמידה במוגבלות התזמון.

чисוב ה-WNS נעשה בשלב המיקום והחוiot של התכנון ומוצג באIOR 3.30.

$$\text{max Freq} = \frac{1}{10ns - 0.171ns} = 101.74MHz$$

לאחר מציאת התקדר המקסימלי יש ליחס את קצב עיבוד המידע (throughput) של האלגוריתם לפי [משוואה 3](#).

(3)

$$\text{Throughput} = \frac{\text{num of bits}}{\text{num of rounds}} \cdot \text{max freq}$$

תחליה יש ליחס את קצב עיבוד המידע המינימלי, בו נכללים מספר הביטים של ההודעה לגיבוב בלבד ומספר הסיבובים של האלגוריתם בלבד ללא התקשרות הטורית. יש לקבוע את גודל המידע המקסימלי ואת מספר הסיבובים המקסימלי כדי להראות את הבדלי קצב עיבוד המידע *Worst Case*.

- מספר הסיביות המקסימלי של ההודעה שאפשר להכניס בתכנון – 4096 סיביות.
- מספר הסיבובים המקסימלי של התכנון – 168 סיבובים.
- מספר הסיבובים להפעלת פונקציית הדחיסה של האלגוריתם – 3 סיבובים.

מכאן קצב עיבוד המידע שהתקבל הינו:

$$\text{min - throughput} = \frac{4096}{168 + 3} \cdot 101.74MHz \cong 2.437 Gbit/sec$$

לאחר חישוב העיבוד המינימלי, יש ליחס את קצב טיפול המידע אשר כולל את המידע, מידע עזר והמידע המגובב. בנוסף לכך, יש להתחשב בסיבובי השעון של התקשרות הטורית.

- מספר הסיביות המקסימלי המועבר לחומרה – 5192 סיביות.
- מספר הסיבובים המקסימלי של התכנון – 168 סיבובים.
- מספר הסיבובים להפעלת פונקציית הדחיסה של האלגוריתם – 3 סיבובים.
- מספר הסיבובים של התקשרות הטורית – 19234 סיבובים.

מכאן, קצב טיפול המידע שהתקבל הינו:

$$\text{throughput} = \frac{5192}{168 + 3 + 19234} \cdot 101.74MHz \cong 27.227 Mbit/sec$$

чисוב קצב עיבוד המידע בתוכנה:

כדי ליחס את קצב עיבוד המידע בתוכנה באלגוריתם MD6 בשפת C, יש להשתמש בספרייה `sys/time.h` כדי ליחס את הפרש הזמן מתחילה הגיבוב עד סוף, כאשר התוצאה המתבקשת כוללת את מספר הסיבובים ואת תדר הריצה של התוכנית.

קצב עיבוד המידע שהתקבל הוא:

$$\text{min - throughput} = \frac{4096}{22.389m} \cong 0.183 \text{ Mbit/sec}$$

תוצאות ההשוואה מלמדות שקצב עיבוד המידע בחומרה, ללא תוספת חישובי העזר, גדול ב-4 סדרי גודל מקצב העיבוד בתוכנה. בהתחשב בתוספת חישובי העזר והתקשורת הטרוית, קצב העיבוד גדול ב-2 סדרי גודל.

קצב עיבוד המידע בתכנון זה אכן לא מספיק לבדוק את המעבר למימוש האלגוריתם בחומרה. אך במידה ומומש האלגוריתם ע"פ תכנון זה עם יותר פונקציות דחיסה, הפרשי קצב העיבוד יגדלו הרבה יותר. בזמן שבוחמרא פונקציות הדחיסה עובדות במקביל וכתוצאה לכך מספר היסובים נשאר זהה, בתוכנה זמן העבודה יכפיל את עצמו עבור כל פונקציית דחיסה. מכאן שהיעילות של האלגוריתם בחומרה רק תגדל.

6 מסקנות וסיכום

המטרות העיקריות של הפרויקט היו מימוש אלגוריתם MD6 בחומרה, הערכת הביצועים תוך השוואת בין מימוש חומרתי למימוש תוכני ויזהו דרך שבה ניתן לשפר את יעילות התכנון על ידי ניתוח נקודות החזק והחולשה שלהם.

מימוש תכנון האלגוריתם MD6 עבר בהצלחה תוך ניצול מרבי של משאבי לוח החומרה. נתקבלו תוצאות של קצב עיבוד מיידי בחומרה אשר היו בכמה סדרי גודל הרבה יותר טובות ביחס לקצב העיבוד בתכנון התוכנית.

במהלך הפרויקט עלו כמה אתגרים כגון כתיבת פרוטוקול התקשרות לצורך העברת מידע לחומרה, משאבי חומרה מוגבלים בלוח ובעקבות כך יכול של התכנון. האתגרים בהם נתקלנו שימשו חוות דעת יקרות ערך והוא מכריים בשכלול התכנון.

מתוך הכרה במוגבלות, כגון כמות מינימלית של משאבי הלוח ובהתאם לכך חוסר יכולת למשמש שילוב של מצבי פעולה ומספר רב יותר של פונקציות דחיסה, מומש מצב פעולה שכיח יותר, כדוגמת מצב פעולה מקביל. יש לנקח בחשבון מגבלות אלו באטרציות עתידיות של הפרויקט.

מטרות הפרויקט הושגו ברמה גבוהה תוך שאיפה לתרום ידע רב בתחום ההצפנה ואבטחת המידע. הפרויקט תורם בהיותו מספק דרך ייחודית למימוש אלגוריתם MD6 בחומרה בצורה ייעילה וכן תורם בהבנת היתרונות של מימוש חומרתי על פני מימוש תוכני. הממצאים מלאים פער מכריים בהבנת ההשלכות המעשיות של יעילות קצב העברת המידע בחומרה.

מחקר עתידי עשוי להעמיק בהשפעות של מימוש חומרתי לעומת מימוש תוכני מבחינת מהירות, זמן וניצול שטח בלוח ובנוסף לחקור תוכנות או אינטגרציות נוספות כדי לשפר עוד יותר את אותם פרמטרים.

עם סיום הפרויקט זהה, מודגש הפוטנציאלי של יישום אלגוריתם גיבוב בחומרה בעיצוב עתיד בתחום ההצפנה ואבטחת המידע. התוצאות החשובות שנמצפו לא רק מאשרות את היעדים הראשוניים אלא גם מעוררות מחויבות מתמשכת לפתרונות חדשניים המשפרים את אבטחת המידע.

מקורות מידע ומארמים

- [1] Follow, G. (2018, November 2). Cryptography introduction. GeeksforGeeks. [Cryptography Introduction - GeeksforGeeks](#)
- [2] Cryptography Hash functions. (n.d.). Tutorialspoint.com. Retrieved December 16, 2023, from [Cryptography Hash functions \(tutorialspoint.com\)](#)
- [3] Rivest, R. L., Agre, B., Bailey, D. V., Crutchfield, C., Dodis, Y., Elliott, K., Khan, F. A., Krishnamurthy, J., Lin, Y., Reyzin, L., Shen, E., Sukha, J., Sutherland, D., Tromer, E., & Yin, Y. L. (n.d.). The MD6 hash function A proposal to NIST for SHA-3. Mit.edu. Retrieved December 16, 2023, from [RABCx08.pdf \(mit.edu\)](#)
- [4] Dworkin, M. J. (2015). SHA-3 standard: Permutation-based hash and extendable-output functions. National Institute of Standards and Technology. [Federal Information \(nist.gov\)](#)
- [5] Merkle, R. C. (n.d.). Ralphmerkle.com. Retrieved December 16, 2023, from [Certified1979.pdf \(ralphmerkle.com\)](#)
- [6] Whatley, C. A., & Hambly, J. (2023). Salt: Scotland's newest oldest industry. John Donald Short Run Press. [What does Salt mean for passwords? | Security Encyclopedia \(hypr.com\)](#)
- [7] Secret key. (n.d.). Hypr.com. Retrieved December 16, 2023, from [What is a Secret Key? | Security Encyclopedia \(hypr.com\)](#)
- [8] Basys 3TM FPGA Board Reference Manual. (2016). Digilent.com. [basys3:basys3_rm.pdf \(digilent.com\)](#)
- [9] UART basics. (n.d.). ECE353: Introduction to Microprocessor Systems. Retrieved December 16, 2023, from [UART Basics – ECE353: Introduction to Microprocessor Systems – UW–Madison \(wisc.edu\)](#)
- [10] (N.d.-b). Researchgate.net. Retrieved December 11, 2023, from [Communication Between Alice and Bob intercepted by Eve. Here channel is... | Download Scientific Diagram \(researchgate.net\)](#)
- [11] (N.d.). Mouser.Co.II. Retrieved December 11, 2023, from [102050644.png \(600×436\) \(mouser.co.il\)](#)
- [12] Customisable design - UART. (n.d.). Tinytapeout.com. Retrieved December 11, 2023, from [Customisable Design - UART :: Documentation in English \(tinytapeout.com\)](#)



נספח א – קישורים למסמכים וקבצי הפרויקט

- **קישור לתקיית קבצי הפרויקט:**
[aviel207/Final Project \(github.com\)](https://github.com/aviel207/Final_Project)
- **קישור לקובץ README של תקיית הפרויקט:**
[Final Project/README.md at main · aviel207/Final Project \(github.com\)](https://github.com/aviel207/Final_Project/blob/main/README.md)
- **קישור לדוח הפרויקט המלא:**
[Final Project/Documents/Project reports/FPGA Implementation of MD6 Hash.pdf at main · aviel207/Final Project \(github.com\)](https://github.com/aviel207/Final_Project/blob/main/Documents/Project%20reports/FPGA%20Implementation%20of%20MD6%20Hash.pdf)
- **קישור לחילק קוד RTL וסימולציה של דוח הפרויקט:**
[Final Project/Documents/Project reports/FPGA Implementation of MD6 Hash - RTL Code and Simulation.pdf at main · aviel207/Final Project \(github.com\)](https://github.com/aviel207/Final_Project/blob/main/Documents/Project%20reports/FPGA%20Implementation%20of%20MD6%20Hash%20-%20RTL%20Code%20and%20Simulation.pdf)
- **קישור לחילק הטמעה וקוד הפעלה של דוח הפרויקט:**
[Final Project/Documents/Project reports/FPGA Implementation of MD6 Hash - Implementation and Executable Code.pdf at main · aviel207/Final Project \(github.com\)](https://github.com/aviel207/Final_Project/blob/main/Documents/Project%20reports/FPGA%20Implementation%20of%20MD6%20Hash%20-%20Implementation%20and%20Executable%20Code.pdf)
- **קישור לסרטון של הפעלת אלגוריתם MD6:**
[Full Demonstration Of Implementing The CF MD6 On FPGA \(youtube.com\)](https://www.youtube.com/watch?v=JyfXzvBjwIY)

נספח ב – פירוט תוכן תקיות הפרויקט

- **תיקיות "Documents"** – תקיות המסמכים מכילה את דוח האלגוריתם שהוגש כאחת מהצעות האלגוריתמים לתחינות תקן הגיבוב SHA-3 של NIST, דוחות הפרויקט, דוח הפרויקט שהוגש לתחינות מטעם AMD-Xilinx ודף המידע של ערכת הפיתוח ה-3 basys .
- **תיקיות "MD6_CF.hardware"** – תקיות קבצי התוכנון של האלגוריתם המומוש בחרומרה המכיל את קבצי המקור, קבצי-h-mem, קובץ ההידור, קובץ-h-XDC, וקבצי-h-test bench.
- **תיקיות "MD6_CF_prototype"** – תקיה המכילה את התוכנון הראשוני של פונקציית הדחיסה.
- **תיקיות "MD6_CF_PAR_MODE"** – תקיה המכילה מימוש של אלגוריתם MD6 בחרומרה במצב פועלה מקבילי עם שתי פונקציות דחיסה.
- **תיקיות "MD6_Operating_Modes"** – תקיה המכילה את תכנון מצבים הפעיל של האלגוריתם.
- **תיקיות "MD6_CF_software"** – תקיה המכילה את המימוש בתוכנה של אלגוריתם MD6 בעל פונקציה דחיסה אחת.
- **תיקיות "Executable_files"** – תקיה המכילה את קבצי הפעלה של המימוש בחומרה, הפעלת אימונות תכנון החומרה על ערכת הפיתוח ואמונות תכנון החומרה על סימולציית ModelSim .



נספח ג – הגשה לתחרות

המימוש בחומרה של אלגוריתם MD6, הוגש לתחרות "Open Hardware Competition" של חברת Xilinx – AMD.

התחרות היא תחרות בין-לאומית בה מתרחרים סטודנטים מאוניברסיטאות שונות ברחבי אירופה ומדינות סמוכות (כגון ישראל) אשר עשו פרויקט על רכיב FPGA של החברה.

המימוש בחומרה של אלגוריתם MD6 שמומש במסגרת פרויקט זה הגיע לשלב הגמר של התחרות

קישור לתוצאות התחרות בה מוצגים המנצח והפיינלייטים:

[2023 Results Gallery - XOHW \(openhw.eu\)](https://2023.Result.Gallery.XOHW.openhw.eu)