בס"ד

המרכז האקדמי לב

הפקולטה להנדסה ומדעי המחשב

החוג להנדסת חשמל ואלקטרוניקה

פרויקט גמר באלקטרוניקה

מימוש FPGA של אלגוריתם MD6 Hash
הטמעה וקוד הפעלה

FPGA Implementation of MD6 Hash Algorithm
Implementation and Executable Code

מגיש: אביאל בירדואקר

מנחה: מר אורי שטרו

ט' טבת תשפ"ד

ירושלים

תודות

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

לד"ר דוד מרטין מרסלו – על הסיוע יוצא הדופן ושיתוף הפעולה לאורך כל הפרויקט. אנחנו אסירי תודה.

המרכז האקדמי לב

תקציר

אלגוריתם MD6 הוא ממשפחת פונקציות הגיבוב בתורת ההצפנה. פונקציית גיבוב, הידועה גם בשם פונקציה חד-כיוונית, מקבלת הודעה אקראית באורך אקראי ומייצרת הודעה מגובבת באורך קבוע. איכות פונקציית הגיבוב נמדדת, בין היתר, ע"פ רמת ה"עירבול" שלה. הוי אומר, נתוני הקלט יעברו פרוצדורה כזו שיהיה כמה שפחות קשר בין הקלט לפלט ושסך כל הפלטים יתפזרו באופן שווה ככל האפשר על פני טווח הפלט. ניתן למצוא מגוון יישומים המבוססים על עקרונות הגיבוב, כגון הפקת "טביעת אצבע" או "חתימה" באורך קבוע עבור מידע דיגיטלי באורך משתנה, או למשל הזנת סיסמא לצורך פתיחת נעילה. אותה סיסמא מומרת לקוד אשר מושווה בתוך המחשב לצורך זיהוי הסיסמא שהוזנה בתחילה. בכל הדוגמאות הללו אין שום מטרה לפענח בחזרה את המידע. אלגוריתם MD6 (פותח על ידי פרופסור Ron Rivest מ-MIT, מומחה בעל עולמי בתחום ההצפנה, וצוות מומחי הצפנה בראשותו) הוגש בתחרות בינלאומית להגדרת אלגוריתם גיבוב מדור שלישי (SHA-3) וזאת כחלק מהמטרה לחזק את אלגוריתמי הגיבוב כנגד פריצות מתוחכמות אשר עלולות להגיע. במסגרת הפרויקט, האלגוריתם ממומש בחומרה ובתוכנה, לצורך השוואה והמחשת היעילות של המימוש בחומרה.

לאלגוריתם MD6 יש כמה מצבי פעולה שבהם הוא משתמש במספר משתנה של פונקציות דחיסה לצורך גיבוב המידע. אי לכך, השימוש בחומרה יכול להיות מאוד יעיל בשל העובדה שניתן לחשב בחומרה כמה פונקציות דחיסה במקביל, או בשביל להתאים בצורה ספציפית את אופן הפעולה של האלגוריתם המתאים לנו, וכמובן מהירות החישוב בחומרה לעומת בתוכנה.

במסגרת הפרויקט מומש אלגוריתם MD6 בשפת החומרה – Verilog. הפרויקט כלל כתיבת האלגוריתם לפי מסמכי המפתחים של האלגוריתם, נבדק ומומש על מערכת פיתוח הכוללת רכיב FPGA. מומשה פונקציית הדחיסה הראשית של אלגוריתם MD6 בצורה יעילה על לוח ה- Basys3 FPGA של Digilent Inc., המבוסס על Xilinx Artix-7 FPGA. קצב עיבוד המימוש בחומרה יותר גבוה לעומת המימוש בתוכנה.

יישום 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.

המרכז האקדמי לב

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 find, 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.

המרכז האקדמי לב

תוכן עניינים

המרכז האקדמי לב

רשימת איורים

רשימת טבלאות

המרכז האקדמי לב

1 מבוא

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

המטרה העיקרית של פרויקט זה היא להעריך את הביצועים של יישום ה-MD6 hash בחומרה לעומת בתוכנה, ולגלות את נקודות החוזק והחולשה שלהם. על ידי הבנת הפשרות בין שני היישומים הללו, אנו שואפים לתרום ידע רב ערך לתחום ההצפנה ואבטחת המידע.

רכיבי FPGAs מתוכננים לבצע עיבוד מקביל, המאפשרים חישובי גיבוב מרובים להתרחש בו-זמנית. בניגוד לגיבוב מבוסס תוכנה, שבו פעולות מבוצעות ברצף של מעבד. FPGAs יכולים לעבד נתחי נתונים מרובים במקביל. מקבילות זו מאיצה משמעותית את תהליך הגיבוב.

ההתמקדות של הפרויקט ביישום חומרה משמעותית במיוחד בהתחשב במשאבים המוגבלים בסביבות חומרה. ניתוח הביצועים של ה-MD6 באופטימיזציה של ניצול המשאבים, יאפשר פעולות קריפטוגרפיות יעילות יותר במכשירים בעלי יכולות עיבוד מוגבלות. ניתן לכוונן יישומי חומרה כך שינצלו ביעילות רכיבים לוגיים וזיכרון, תוך הבטחה שפעולת הגיבוב חסכונית במשאבים.

המחקר ההשוואתי בין יישומי חומרה ותוכנה הוא המפתח למתן הבנה מקיפה של הפשרות הכרוכות בכך. ניתוח זה ילמד לגבי היישום המתאים ביותר בהתבסס על מקרי שימוש ספציפיים ומגבלות משאבים. הניתוח ההשוואתי בין יישומי חומרה ותוכנה משמש מדריך למפתחים וארכיטקטי מערכות בבחירת הגישה המתאימה ביותר בהתבסס על הדרישות הספציפיות שלהם. הדרכה זו חיונית לפיתוח מערכות מאובטחות ויעילות בתחומים שונים.

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

פרויקט זה ממלא תפקיד מרכזי בקידום ידע קריפטוגרפי, אופטימיזציה של ניצול המשאבים ומתן הדרכה מעשית לעיבוד נתונים מאובטח. ההטמעה והניתוח ההשוואתי של ה-MD6 בפלטפורמות חומרה ותוכנה תורמים הן להבנה המדעית של פונקציות ה-hash הצפנה והן ליישומים המעשיים שלהן, במטרה סופית לשפר את אבטחת המידע בעולם יותר ויותר דיגיטלי.

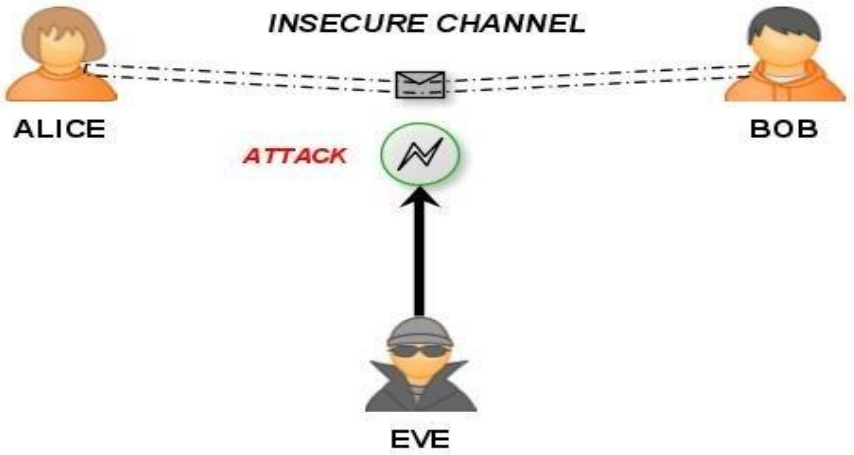
2 רקע תיאורטי

2.1 מבוא לקריפטוגרפיה [1]

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

- סודיות: מתייחסת לכללים והנחיות מסוימים המבוצעים בדרך כלל במסגרת הסכמי סודיות המבטיחים שהמידע מוגבל לאנשים או למקומות מסוימים.
- שלמות: הנתונים מתייחסת לשמירה והבטחה שהנתונים יישארו מדויקים ועקביים לאורך כל מחזור החיים שלו.
- אימות: הוא תהליך לוודא שפיסת הנתונים שנתבע על ידי המשתמש שייכת אליו.
- אי-הכחשה: מתייחסת ליכולת לוודא שאדם או צד הקשורים לחוזה או תקשורת אינם יכולים להכחיש את האותנטיות של חתימתם על המסמך שלהם או על שליחת הודעה.

ניקח לדוגמא 2 משתתפים לאחת קוראים אליס (השולחת) ולשני בוב (המקבל). כעת, אליס רוצה לשלוח הודעה לבוב דרך ערוץ מאובטח. התהליך הוא כדלקמן. הודעת השולח, או לפעמים נקראת Plaintext, מומרת לצורה בלתי ניתנת לקריאה באמצעות מפתח - k. הטקסט המתקבל נקרא ה-Ciphertext. תהליך זה ידוע בשם הצפנה. בזמן הקבלה, ה-Ciphertext מומר בחזרה לטקסט הפשוט באמצעות אותו מפתח - k, כך שניתן לקרוא אותו על ידי המקלט. תהליך זה ידוע בשם פענוח.



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

ישנם מספר סוגים של קריפטוגרפיה, כל סוג עם תכונות 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 המקבל משתמשים באותו מפתח, אותו יש לשמור בסוד כדי לשמור על אבטחת התקשורת.

- 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: פונקציית גיבוב היא אלגוריתם מתמטי הממיר נתונים בכל גודל לפלט בגודל קבוע. לעתים קרובות נעשה שימוש בפונקציות גיבוב כדי לאמת את שלמות הנתונים ולהבטיח שלא טיפלו בהם.

אלגוריתם MD6 נמצא תחת קטגוריית פונקציות גיבוב קריפטוגרפית, ולכן נרצה להרחיב על פונקציית הגיבוב בפרק הבא.

2.2 פונקציית גיבוב – HASH [2]

פונקציית גיבוב היא פונקציה חד-כיוונית הממירה קלט באורך כלשהו לפלט באורך קבוע וידוע מראש .פונקציית גיבוב מתוכננת כך שכל שינוי בקלט יגרום לשינוי משמעותי בפלט. בדרך זו ניתן להתמודד עם בעיית הבטחת שלמות מסרים גדולים, על ידי השוואת הערך המגובב שלהם במקום להשוותם ישירות. בשל היותו קטן משמעותית, קל יותר להגן על הערך המגובב מאשר על המסר המקורי.

פונקציות גיבוב קריפטוגרפיות הן מאבני הבסיס של ההצפנה המודרנית ומשמשות כחתימות דיגיטליות, קודי אימות, שמירת סיסמאות ומחולל מספרים פסידו-אקראיים. ביישומים שאינם קריפטוגרפים הן משמשות לעיתים כמזהה ייחודי של קובץ לצורך בדיקת שלמותו או נכונותו וכן לזיהוי קבצים זהים.

פונקציית גיבוב קריפטוגרפית בטוחה מקיימת את התנאים הבאים:

- Pre-Image Resistance:
  מאפיין זה אומר שיהיה קשה מבחינה חישובית להפוך פונקציית Hash.
  במילים אחרות, אם פונקציית גיבוב H יצרה ערך גיבוב z, אז זה אמור להיות תהליך קשה למצוא ערך קלט x שייתן z.

  מאפיין זה מגן מפני תוקף שיש לו רק ערך hash והוא מנסה למצוא את הקלט.

- Second Pre-Image Resistance:
  מאפיין זה אומר בהינתן קלט והגיבוב שלו, זה אמור להיות קשה למצוא קלט אחר עם אותו גיבוב.

  במילים אחרות, אם פונקציית גיבוב H עבור קלט x מייצרת ערך גיבוב $H(x)$, אז זה אמור להיות קשה למצוא כל ערך קלט אחר y כך ש- $H(y) = H(x)$.

  מאפיין זה של פונקציית גיבוב מגן מפני תוקף שיש לו ערך קלט והגיבוב שלו, ורוצה להחליף ערך שונה כערך לגיטימי במקום ערך הקלט המקורי.

- Collision Resistance:
מאפיין זה אומר שזה אמור להיות קשה למצוא שני כניסות שונות בכל אורך שמביאות לאותו גיבוב. מאפיין זה מכונה גם פונקציית גיבוב ללא התנגשות.

במילים אחרות, עבור פונקציית גיבוב H קשה למצוא שתי כניסות שונות x ו-y כך ש-$H(x) = H(y)$.

מכיוון שפונקציית גיבוב היא דחיסת פונקציה עם אורך גיבוב קבוע, לא ייתכן שלפונקציית גיבוב לא יהיו התנגשויות. מאפיין זה רק מאשר שקשה למצוא את ההתנגשויות הללו.

מאפיין זה מקשה מאוד על תוקף למצוא שני ערכי קלט עם אותו גיבוב. כמו כן, אם פונקציית הגיבוב עמידה בפני המאפיין הזה היא עמידה גם לשני 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 לתחרות תקן הגיבוב SHA-3 של NIST [4].

האלגוריתם פועל באמצעות מבנה Merkle-Damgård [5], כלומר הוא מעבד נתוני קלט בבלוקים ודוחס כל בלוק לפלט בגודל קבוע. אלגוריתם MD6 כולל גודל בלוק גמיש, המאפשר לו לעבד ביעילות נתונים בגדלים משתנים. הוא כולל גם מספר תכונות חדשניות, כגון פונקציות דחיסה מקבילה ואלגוריתם עדכון הודעות מבוסס עץ בינארי. כל מילה באלגוריתם מוגדרת כ-64 סיביות.

2.3.1 הקלט והפלט של ה-MD6

הקלט ל-MD6 הן כדלקמן:

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

כניסות החובה היחידות הן ההודעה M שיש לגיבוב ואורך ההודעה המגובבת d. לכניסות האופציונליות יש ערכי ברירת מחדל אם לא מסופק ערך כלשהו.

הפלט של MD6 הוא H - מחרוזת סיביות באורך של d סיביות.

2.3.1.1 ההודעה – M

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

2.3.1.2 אורך ההודעה המגובבת – d

הקלט השני ל-MD6 הוא אורך הסיביות d של ההודעה המגובבת אשר ערכיו יכולים לנוע בין $0 < d \le 512$.

d חייב להיות ידוע בתחילת חישוב הגיבוב, מכיוון שהוא לא רק קובע את אורך הפלט ה-MD6 הסופי, אלא גם משפיע על חישוב ה-MD6 בכל פעולת ביניים

אורכי ה-d כפי שנדרש מ-SHA-3 הם: $224, 256, 384, 512$.

2.3.1.3 המפתח – K

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

המפתח הוא קלט אופציונלי, ברירת המחדל כאשר המשתמש בוחר לא להכניס מפתח הוא $K = 0$ ואורכו $k = 0$ סיביות.

2.3.1.4 פרמטר בקרת המצב – L

הקלט הבא ל-MD6 הוא פרמטר בקרת המצב L, המשמש לבחירת אחד ממצבי הפעולה של ה-MD6 אשר יפורטו להלן בפרק 2.3.3. אורך פרמטר בקרת המצב L הוא $0 \leq L \leq 64$.

פרמטר בקרת המצב L הוא אופציונלי, ברירת המחדל כאשר המשתמש 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 כמוסבר להלן בפרק 2.3.4. אורך מספר הסיבובים הוא $0 < r < 168$.

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

2.3.2 קבועים ב-MD6

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

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

2.3.2.1 וקטור הקבועים Q

וקטור הקבועים Q זהו וקטור אשר מכיל 15 מילים של 64 סיביות. הוקטור משמש כחלק מבלוק הנתונים אשר נכנס לכל פונקציית דחיסה באלגוריתם MD6 כמוסבר לעיל בפרק 2.3.4.

ערכי ה-Q מוצגים בטבלה 1.

טבלה 1: וקטורי הקבועים - Q

0	0x7311c2812425cfa0	**1**	0x6432286434aac8e7	**2**	0xb60450e9ef68b7c1
3	0xe8fb23908d9f06f1	**4**	0xdd2e76cba691e5bf	**5**	0x0cd0d63b2c30bc41
6	0x1f8ccf6823058f8a	**7**	0x54e5ed5b88e3775d	**8**	0x4ad12aae0a6d6031
9	0x3e7f16bb88222e0d	**10**	0x8af8671d3fb50c2c	**11**	0x995ad1178bd25c31
12	0xc878c1dd04c4b633	**13**	0x3b72066c7a1552ac	**14**	0x0d6f3522631effcb

2.3.2.2 וקטור הקבועים S

וקטור הקבועים S זהו וקטור אשר מכיל 168 מילים של 64 סיביות. הוקטור משמש כחלק מחישוב פונקציה הדחיסה כאשר בכל סיבוב משתמשים במילה אחרת ב-S לפי מספר הסיבוב כמוסבר לעיל בפרק 2.3.4.

ערכי ה-S מוצגים בטבלה.

טבלה 2: וקטורי הקבועים - S

0	0x0123456789abcdef	**1**	0x0347cace1376567e	**2**	0x058e571c26c8eadc
3	0x0a1cec3869911f38	**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	0x95dc0c5dcfdede5a	**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	0xfab2e097b7b92366	**34**	0xfab2e097b7b92366	**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	0xd5624c14db1fdba2	**52**	0xfbc4d829b63a7ce5	**53**	0x848970524854b56b
54	0x0913a0a490adeff7	**55**	0x1336c1c9217e104e	**56**	0x357d431362d8209c
57	0x5bebc427e5b041b8	**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**	0x08a8befe4342cb56
123	0x1151ff7c86855dac	**124**	0x33b23cf9090ff6f8	**125**	0x54747973121a2b50
126	0xf8f8b2e724345da0	**127**	0x81e1e74f6c4cf6e1	**128**	0x02c20c9ffc9d2b63
129	0x078419bedd3f5de6	**130**	0x0c0833fdbe5bf66c	**131**	0x1810657a58b62af8
132	0x20308af4b1485f50	**133**	0x607197694290f1a0	**134**	0xa0f2acd3852122e0
135	0x61f5d9260e634761	**136**	0xa2fa724c18e7c9e2	**137**	0x67e4a69831ea5a65
138	0xacc9cfb043f4feea	**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 ההזזה r&l

וקטורי הקבועים r&l הם וקטורי הזזה לימין (r) ולשמאל (l) אשר כל אחד מכיל 16 מספרים הקסדצימלים, הוקטורים משמשים לחלק מחישוב פונקציית הדחיסה, כאשר בכל סיבוב מחשבים 16 מילים חדשות כמוסבר להלן בפרק 2.3.4 לכל מילה משתמשים בערך אחר של הזזה לפי המיקום בוקטור.

ערכי ברירת המחדל מוצגים בטבלה 3.

טבלה 3: וקטורי ההזזה r&l

$(i-n)\%16$	**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 כחלק מחישוב פונקציית הדחיסה, כאשר הקבועים בוחרים על אלו מילים מתוך בלוק הנתונים פונקציית הדחיסה תעשה את פעולות חישוב.

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

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

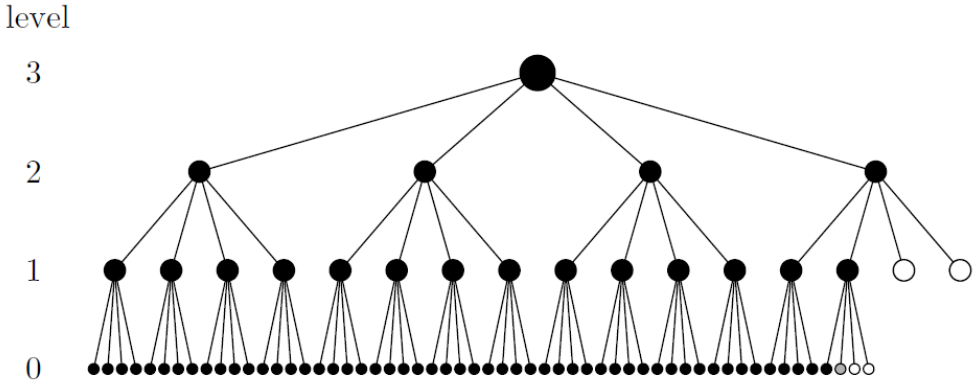
2.3.3 מצבי הפעולה של ה-MD6

באלגוריתם MD6 יש 3 מצבי פעולה:
1) מקבילי – עץ היררכי
2) טורי – מבוסס על מבנה Merkle-Damgård
3) היברידי – שילוב של מקבילי וטורי

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

- <u>מצב פעולה מקבילי:</u>
במצב מקבילי אשר גם קרוי מצב הפעולה הסטנדרטי, האלגוריתם מחלק את המידע המתקבל לבלוקים בגודל 16 מילים - כלומר 1024 סיביות בלוק המידע זה נקרא בלוק B, וכל 4 בלוקים נכנסים לפונקציית דחיסה אחת (נרחיב עליה <u>בפרק 2.3.4</u>) אשר הפלט שלה זהו בלוק בגודל 16 מילים, זאת אומרת שמספר הבלוקים מתחלק בכל רמה ב-4. לכן מספר הרמות המקסימלי האפשרי הוא 27 שכן $\log_4\left(\frac{2^{64}}{1024}\right) = 27$.

ערך בחירת המחדל של פרמטר בקרת המצב הוא $L = 64$ אשר מבטיח שאופן הפעולה תהיה בצורה המקבילה שהיא אופן הפעולה סטנדרטית.



איור 2.2 – מצב הפעולה המקבילי [3].

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

כל קבוצה של 4 בלוקים מרמה 0 נכנסים לפונקציית דחיסה ברמה 1 אשר מוציאה כפלט 16 מילים, והם העיגולים השחורים המופיעים ברמה 2 וכן הלאה בכל הרמות, במקרה הזה אפשר לראות שנשאר בלוק אחד של נתונים ברמה 3 ולכן זוהי הרמה האחרונה.

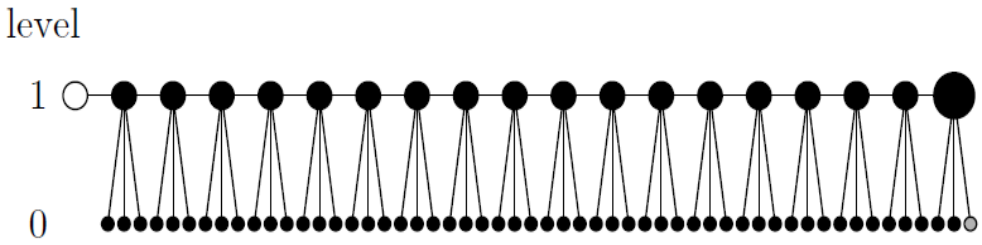
- <u>מצב פעולה טורי:</u>
במצב פעולה טורי, האלגוריתם כמו במצב הפעולה המקבילי מחלק את ההודעה לבלוקים של 16 מילים, אבל לעומת מצב הפעולה המקבילי, מצב הפעולה הטורי עובד עם שתי רמות בלבד.

כאשר ברמה 0 מופיעים כל בלוקי המידע המחולקים ל-16 מילים, וברמה 1 מופיעים כל פונקציות הדחיסה הנצרכות לצורך הגיבוב ובקצה הכי שמאלי וקטור איתחול של 16 מילים של אפסים – "initialization vector" או בקיצור IV.

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

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

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



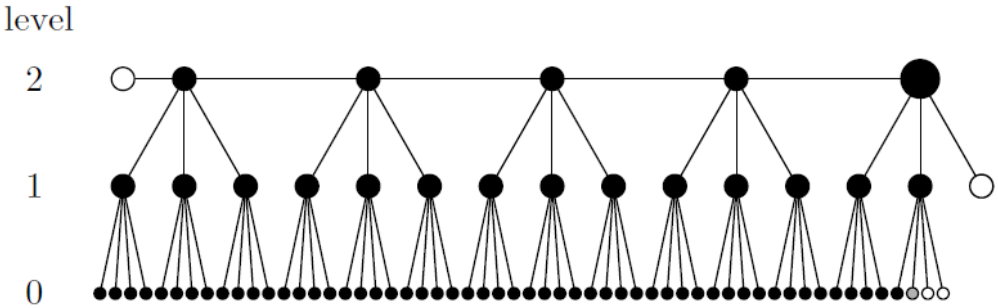
איור 2.3 – מצב הפעולה הטורי [3].

- <u>מצב פעולה היברידי:</u>

מצב הפעולה ההיברידי מאחד בין מצבי הפעולה המקבילי והטורי. במידה ומצד אחד, ערך ה-L שנקבע גדול מ-0, אך מצד שני, קטן ממספר הדרגות הנצרך 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 תוצאת הגיבוב.

לשם ההמחשה, במידה ו-$L = 1$, 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 כמתואר באיור 2.4.



איור 2.4 – מצב הפעולה ההיברידי [3].

2.3.4 פונקציית הדחיסה

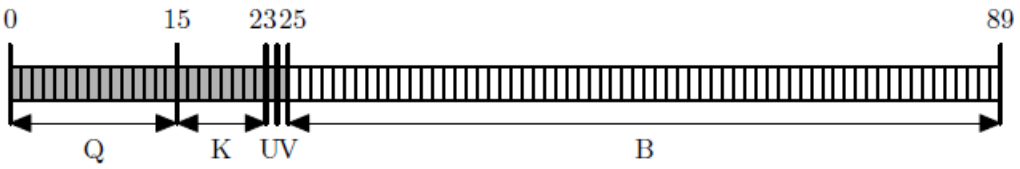
פונקציית הדחיסה, היא פונקציית החישוב המרכזית של אלגוריתם MD6, כל פונקציית דחיסה מקבלת 64 מילים של 64 סיביות שהם 4096 סיביות של הודעה, ומוציאה גיבוב של 16 מילים של 64 סיביות שהם 1024 סיביות מידע. בפרק זה נפרט על מבנה הפונקציה ואופן פעולתה.

2.3.4.1 מבנה פונקציית הדחיסה

קלט פונקציה הדחיסה הוא בלוק מידע B בעל 64 מילים, והוא נטמע בתוך בלוק נתונים N בעל 89 מילים אשר מורכב מכמה חלקים כמוצג באיור 2.5.

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

על הכניסות Q, K, B פירטנו בפרקים 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 שלה במצב הפעולה באלגוריתם ע"י index ו-level, כאשר level זה מספר השלב ו-index זה המיקום באותו שלב או במילים אחרות הערך הסידורי של פונקציית הדחיסה הנוכחית, 7 הבתים הראשונים של U מכילים את ערך ה-index וה-byte שנשאר מכיל את ערך ה-level, כמופיע באיור 2.6.

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

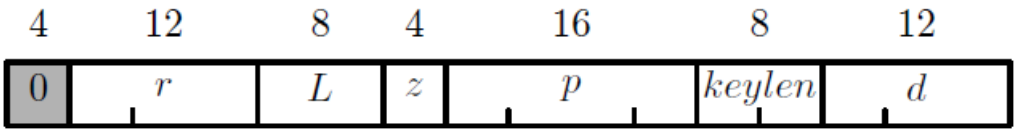
2.3.4.1.2 מילת הבקרה V

מילת הבקרה V מורכבת מ-6 חלקים שונים כמופיע באיור 2.7.

נפרט:

- d – אורך ההודעה המגובבת, 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 האחרונה אחרת שווה ל-0, באורך 4 סיביות

- L – פרמטר בקרת המצב, באורך של 8 סיביות
- r – מספר הסיבובים, באורך של 12 סיביות
- 4 סיביות של אפסים



איור 2.7 – פריסת מילת הבקרה V [3].

2.3.4.2 פעולת החישוב של פונקציית הדחיסה

כמו שהוסבר בפרק הקודם, פונקציית הדחיסה מעבדת את בלוק הנתונים N, ומפעילה את לולאת החישוב המוצגת במשוואה 1.

(1)

$$n = 89, t = 16$$

$$\text{for j in range}(0, r)$$

$$\text{for i in range}(n + j \cdot t, n + (j + 1) \cdot t)$$

$$\{$$

$$x = S_j \oplus A_{i-n} \oplus A_{i-t_0}$$

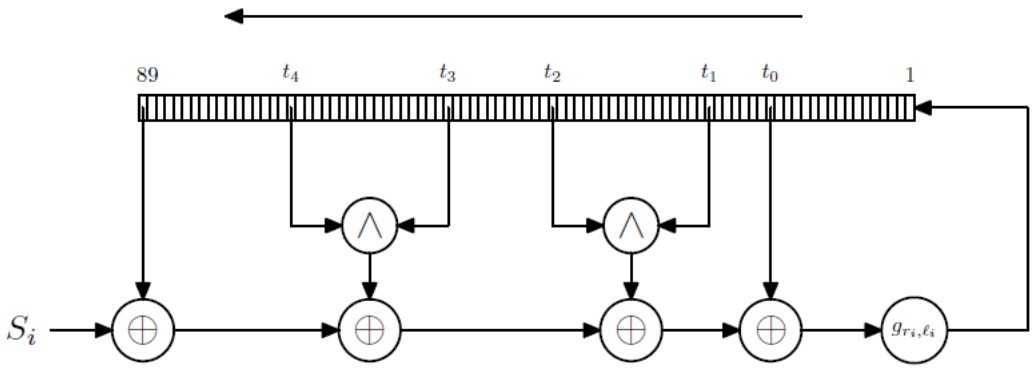
$$x = x \oplus (A_{i-t_1} \,\&\, A_{i-t_2}) \oplus (A_{i-t_3} \,\&\, A_{i-t_4})$$

$$x = x \oplus (x \gg r_{(i-n)\%16})$$

$$A_i = x \oplus (x \ll l_{(i-n)\%16})$$

$$\}$$

אפשר להציג את לולאת החישוב כאוגר הזזה בעל משוב לא לינארי כמוצג באיור 2.8.



איור 2.8 – לולאת החישוב מוצגת כאוגר הזזה לא לינארי [3].

בכל סיבוב, הפונקציה מחשבת 16 מילים חדשות, המילה המחושבת הראשונה תתווסף לבלוק הנתונים N כך שכעת 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 מקבועי עמדות ההקשה t.

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

בשורה השלישית עושים XOR עם התוצאה של השורה השנייה ועם אותה תוצאה אשר מוזזת ימינה לפי וקטורי ההזזה r.

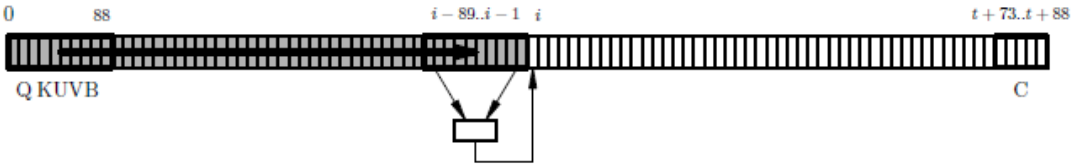
בשורה הרביעית עושים XOR עם התוצאה של השורה השלישית ועם אותה תוצאה אשר מוזזת שמאלה לפי וקטורי ההזזה l.

התוצאה מוכנסת לבלוק הנתונים במיקום של ה-index – i.

קבועי עמדות ההקשה t כמו שהסברנו לעיל נמצאים בטווח הנ"ל $16 < t_{0-4} < 89$ ולכן:

- כל סיבוב בחישוב משתמש רק ב-89 המילים ב-index הכי גבוה של בלוק הנתונים (89 המילים האחרונות), כמוצג באיור 2.9, מה שמאפשר חיסכון בזיכרון כאשר בכל סיבוב נוסיף את ה-16 המילים החדשות ונוריד את 16 המילים הראשונות מבלוק הנתונים.

- משום שאין 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- r מחזורי שעון.

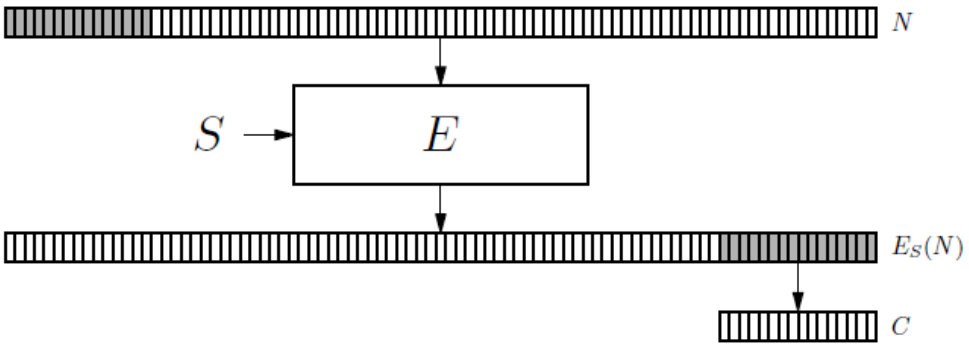


איור 2.9 – אופן הפעולה של לולאת החישוב [3].

2.3.4.3 פלט פונקציית הדחיסה

פלט פונקציית הדחיסה זהו 16 המילים האחרונות שחישבנו בלולאת החישוב של פונקציית הדחיסה כמוצג באיור 2.10

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



איור 2.10 – פונקציית הדחיסה f נראית כפעולת הצפנה ואחריה פעולת חיתוך [3].

3 פיתוח ושיטות

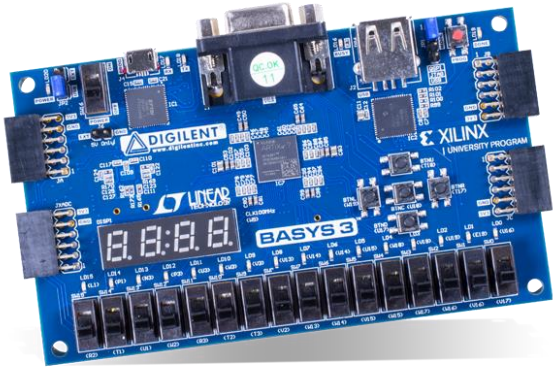
במסגרת הפרויקט, מומש אלגוריתם MD6 אשר מכיל פונקציה דחיסה אחת בלבד אשר פועלת לפי מצב הפעולה הסטנדרטי, עקב מחסור במשאבי רכיב החומרה כפי שיוסבר להלן.

קישור למימוש מצבי הפעולה (הנמצאים בתיקיית הפרויקט) נמצא בנספח א.

פרק זה מחולק באופן הבא:

- ערכת הפיתוח ורכיב ה-FPGA
- מעטפת להעברת נתונים
- הטמעה על רכיב ה-FPGA
- יצירת קוד תוכנה עם תוסף ממשק משתמש

3.1 ערכת הפיתוח ורכיב ה-FPGA [8]



איור 3.1 – ערכת הפיתוח Basys-3 [11].

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

טבלה 5 מציגה את המאפיינים העיקריים של ערכת הפיתוח.

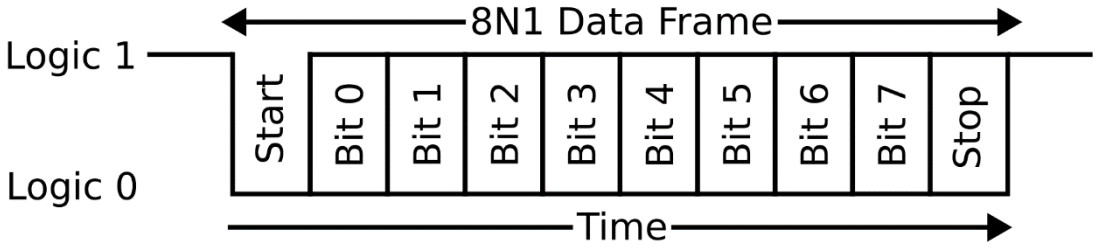
טבלה 5: מאפייני ערכת הפיתוח

תכונה	תיאור
רכיב ה-FPGA	Artix-7 XC7A35T-1CPG236C
ממשקי כניסות ויציאות	• USB-UART לתכנות ותקשורת טורית • USB-UART Bridge • יציאת VGA של 12 סיביות • USB HID Host לעכברים מקלדות וזיכרונות
זיכרון	זיכרון 32Mb Serial Flash
תצוגות	תצוגה אחת בת 4 ספרות בת 7 פלחים
מתגים ונוריות	• 16 מתגים • 16 נורות לד • 5 כפתורים
שעונים	מתנד קריסטל אחד של 100MHz
יציאות הרחבה	• Pmod עבור אותות XADC

	• 3 יציאות Pmod
אמצעיים	• 33,280 תאים לוגיים ב-5200 חלקיים (כל חלק מכיל ארבעה LUTs עם 6 כניסות ו-8 FFs) • 1800Kb של זיכרון RAM בלוק מהיר

3.2 מעטפת להעברת נתונים

בכדי לגבב את ההודעה, תחילה יש "להכשיר את הקרקע" ולהכין את המעטפת שתנהל את המידע שמגיע מהמחשב ללוח ומשודר בחזרה. לשם כך מימשנו פרוטוקול תקשורת UART מסוג 8N1 [9]
(8 סיביות, ללא סיבית זוגיות, ועם סיבית עצירה אחד) כמוצג באיור 3.2. הסיבה שבחרנו בפרוטוקול זה היא משום שאנו משתמשים בערכת הפיתוח ה-Basys3 אשר קיים בה רכיב UART מובנה בכניסת ה-micro USB [8].

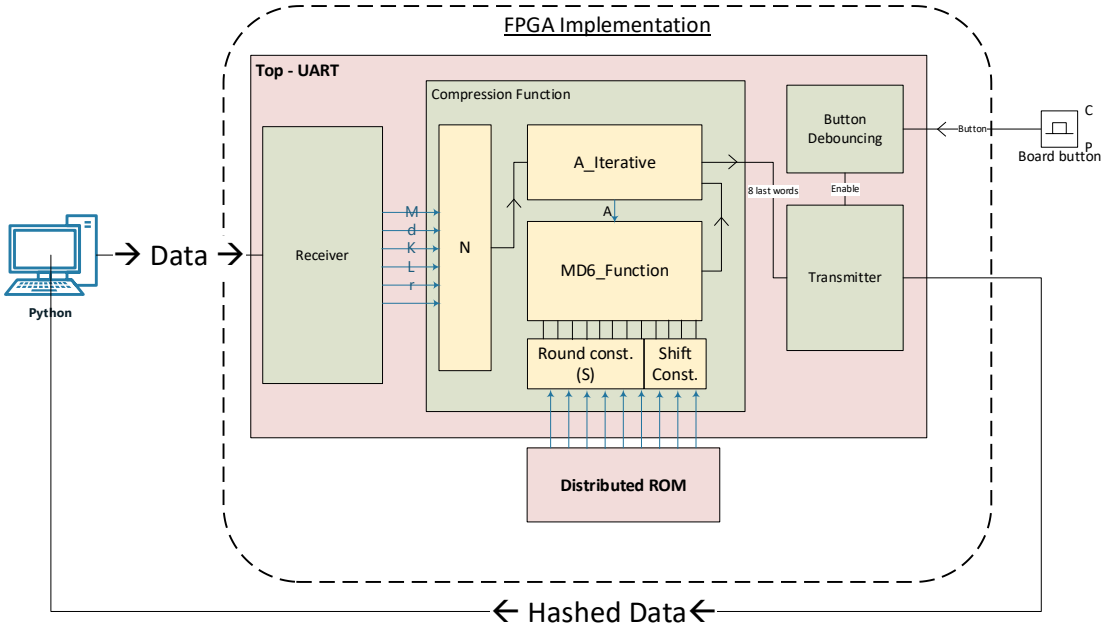


איור 3.2 – מבנה שליחת המידע בתקשורת UART מסוג 8N1 [12].

המידע המתקבל מהמשתמש יכול להיכתב בייצוג בינארי, הקסדצימלי או ASCII.
באמצעות פרוטוקול ה-UART נשלח מידע מסוג byte בלבד. נדרש להמיר את המידע המתקבל מהמשתמש, למידע מסוג byte ורק אז להעביר דרך ה-UART.

בנוסף, כדי להקל על החומרה מחישובים מיותרים (רוטציות, ריפוד באפסים, חישוב אורכי וקטור...), המידע מעובד מראש, כך שהוא מרופד באפסים בגדלים קבועים כדי לקטלג את סוג המידע למקום המתאים באלגוריתם וכמו כן, תוכך כדי שהמידע מתקבל דרך ה-UART, הוא מסודר ב-Big-Endian כך שאין צורך לעשות רוטציה כלשהי. חוץ מהמידע המתקבל מהמשתמש, נוסף מידע לצורך בקרה ובכך הושג חיסכון בחישובים בחומרה שמבזבזים משאבים קריטיים.

באיור 3.3 ניתן לראות דיאגרמת בלוקים של התכנון כולל מעטפת העברת המידע.



איור 3.3 – דיאגרמת בלוקים של התכנן כולל המעטפת.

3.3 הטמעה על רכיב ה-FPGA

הטמעת המימוש על הרכיב כוללת את החלקים הבאים:

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

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

3.3.1 Constraint Specification – מפרט אילוצים

מפרט האילוצים מציין אילוצים שונים כדי להנחות את תהליך הסינתזה והיישום .
אילוצים אלה כוללים אילוצי שעון (לדוגמה, תדר שעון, תחומי שעון), אילוצי קלט/פלט (לדוגמה, הקצאות פינים) ומגבלות תזמון (למשל, זמני הגדרה והחזקה).

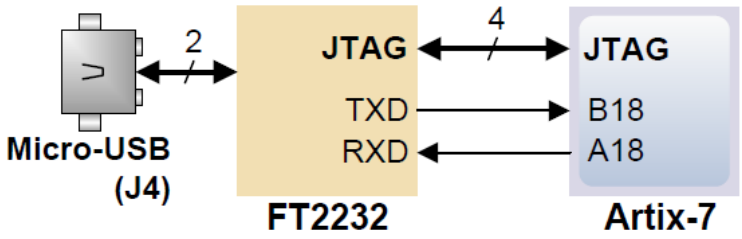
מפרט האילוצים נכתב בקובץ XDC (Xilinx Design Constraints) המבוסס טקסט ומשמש לצורך הגדרת האילוצים של התכנון.

להלן פירוט האילוצים:

- Clocks:
  הוגדר אילוץ זמן מחזור של 10ns או במילים אחרות תדר של 100MHz ב-duty cycle של 50%.
- Input & Output delay:
  הוגדר שבכניסת המידע ובחזרתו מהרכיב לא יהיה זמן עיכוב (לא נצרך כי כל המידע נכנס לאותו המקום אחד אחרי השני).
- Configuration settings:
  הוגדר שהמתח שבו יפעלו פיני תצורת ה-FPGA הוא CFGBVS VCCO והוגדר 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 חיבורי היציאות לפינים:

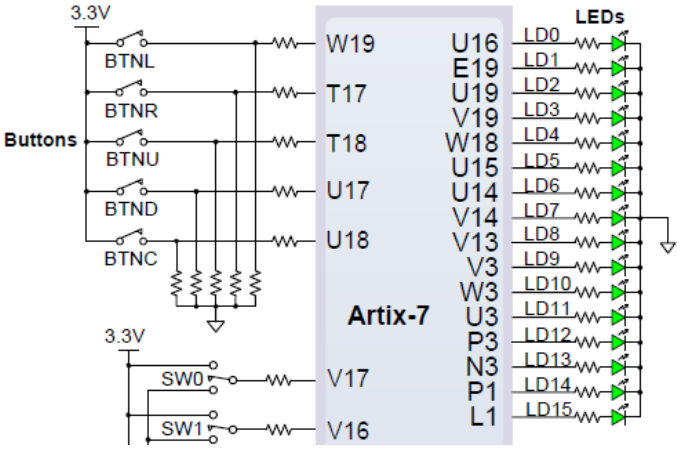
- השעון : clk - מחובר לפין W5 אשר שם מחובר המתנד של ה-Basys3.
- תקשורת ה-UART:
  TxD – מחובר לפין A18
  RxD – מחובר לפין B18
  בפינים אלו מחובר רכיב ה-UART של ערכת הפיתוח ה-Basys3 כמוצג באיור 3.23



איור 3.4 – חיבורי תקשורת ה-UART בערכת הפיתוח.

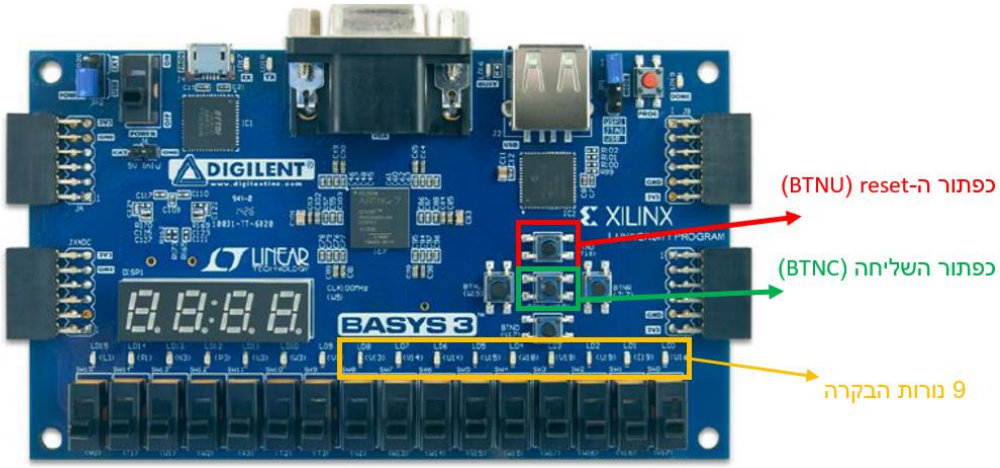
- כפתורי הלחיצה:
  reset – מחובר לפין T18 אשר מחובר לכפתור הלחיצה BTNU.
  Button tx – מחובר לפין U18 אשר מחובר לכפתור הלחיצה BTNC.
  כמוצג באיור 3.24.

- נוריות:
  Done M – מחובר לפין U16 אשר מחובר לנורה LD0.
  Done d – מחובר לפין E19 אשר מחובר לנורה LD1.
  Done K – מחובר לפין U19 אשר מחובר לנורה LD2.
  Done L – מחובר לפין V19 אשר מחובר לנורה LD3.
  Done r – מחובר לפין W18 אשר מחובר לנורה LD4.
  Done keylen – מחובר לפין U15 אשר מחובר לנורה LD5.
  Done padding – מחובר לפין U14 אשר מחובר לנורה LD6.
  Done rx – מחובר לפין V14 אשר מחובר לנורה LD7.
  Done MD6 – מחובר לפין V13 אשר מחובר לנורה LD8.
  כמוצג באיור 3.24.



איור 3.5 – חיבור הנוריות והלחצנים לפינים בערכת הפיתוח.

את מיקום כפתורי הלחיצה והנוריות על ערכת הפיתוח אפשר לראות באיור 3.25.



איור 3.6 – מיקום הכפתורים והנוריות על ערכת הפיתוח.

3.3.2 Synthesis– סינתזה

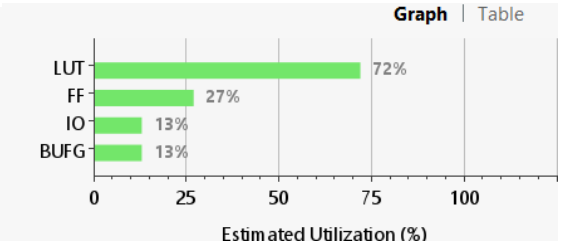
סינתזה ממלאת תפקיד מהותי בהמרת תיאור חומרה ברמה גבוהה (הכתוב בשפת Verilog) ל-netlist שהוא ייצוג ברמה נמוכה של התכנון שניתן להשתמש בו לצורך אופטימיזציה והטמעה.

התוכנה לוקחת את קוד ה-RTL ומתרגמת אותו לייצוג netlist מבני של התכנון. לאחר מכן מבצעת טכניקות אופטימיזציה שונות כדי לשפר את ביצועי התכנון, ניצול השטח וצריכת החשמל, ע"י הפחתת אוגרים מיותרים וייעול החיוט בין החלקים השונים. לאחר מכן היא ממפה את התכנון המסונתז על פני רכיב ה-FPGA שבו השתמשנו.

כפי שהוסבר בפרק 3.3.2 בחלקי קוד RTL והסימולציה של הדו"ח המצורפים בנספח א. התכנון ההתחלתי (רק ה-cf) לאחר העברתו בסינתזה השתמש ביחס ל-spec של ה-Basys3, בסביבות 400% שימוש ב-FFs ו-37,000% שימוש ב-LUTs מרכיבי החומרה הקיימים ברכיב ה-FPGA, כמוצג באיור 3.10. לאחר המעבר לתכנון האיטרטיבי, התכנון לאחר סינתזה עבר שינוי משמעותי מבחינת כלי החומרה בו הוא משתמש כמוצג באיור 3.26.

Graph | **Table**

Resource	Estimation	Available	Utilization %
LUT	14875	20800	71.51
FF	11204	41600	26.93
IO	14	106	13.21
BUFG	4	32	12.50



איור 3.7 – כלי החומרה המנוצילם לאחר סינתזה.

התוכנה לוקחת בחשבון את האילוצים המתקיימים בתכנון ונותנת משוב לעשות שינויים כדי להתאים את התכנון כנדרש.

בנוסף במקרה של שגיאות או כאשר התוכנה מזהה בעיות היכולות לגרום לכשלים בתכנון או דברים הניתנים לייעול, התוכנה שולחת משוב בדמות "שגיאה" במקרה של שגיאות ו"אזהרות" במקרה של בעיות או דברים הניתנים לשיפור.

בתכנון שלנו, יש 205 אזהרות המתקבלות בתהליך הסינתזה. נפרטם ונסביר את הסיבה שהן מופיעות.

- אזהרה: [Synth 8-7129]

כמות אזהרות: 100

הסבר:

האזהרה מציינת שכלי הסינתזה זיהה שייתכן שחלק מהאותות בתכנון עברו אופטימיזציה, אך הם לא הוסרו. אזהרה זו נוצרת בדרך כלל כאשר יש אותות בתכנון שאינם בשימוש או שאין להם השפעה על הפונקציונליות של התכנון.

בתכנון שלנו:

הסיבה לאזהרה היא כי פונקציית הדחיסה בנויה כך שהיא לא משתמשת בכל המידע, כיוון שהמידע שבשימוש תלוי בקבועים $t_1 - t_4$. משום שקבועים אלו באלגוריתם לא חייבים להיות בערכים שנתנו להם והם בעלי שינוי, אז המידע שלא בשימוש בתכנון הנוכחי נצרך.

- אזהרה: [Synth 8-3917]

כמות אזהרות: 100

הסבר:

האזהרה מציינת שכלי הסינתזה זיהה מצב שעלול לגרום למחלוקת או להתנהגות לא מוגדרת במהלך פעולת התכנון. אזהרה זו נוצרת בדרך כלל כאשר יש התנגשות פוטנציאלית בין שני אותות או יותר בתכנון.

בתכנון שלנו:

הסיבה לאזהרה היא כי נכנס ערך 0 לאותות בתחילת התכנון, ואין התנגשויות עם אותות אחרים. כמובן שהאיפוס של האותות נצרך כדי להשתמש בתכנון באופן רב פעמי.

- אזהרה: [Synth 8-689]

כמות אזהרות: 1

הסבר:

האזהרה מציינת שכלי הסינתזה זיהה שיש אי התאמה בין רוחב אות או חיבור יציאה בתכנון לבין רוחב היציאה המתאימה במופע מודול.

בתכנון שלנו:

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

- אזהרה: [Synth 8-6014]

כמות אזהרות: 3

הסבר:

האזהרה מציינת שכלי הסינתזה זיהה אלמנט רציף בתכנון שאינו בשימוש והוסר.

בתכנון שלנו:

האלמנטים המדוברים נחוצים לאתחול התוכנית לשימוש חוזר בכל סוגי המצבים בקוד.

- אזהרה: [Synth 8-327]

כמות אזהרות: 1

הסבר:

האזהרה מציינת שכלי הסינתזה זיהה משתנה בתכנון שלא מוקצה לו ערך בכל מצב אפשרי, ויצר latch לאחסון הערך של המשתנה.

בתכנון שלנו:

ה-D נצרך כדי לקבל את החלק היוצא של המידע המוצפן, לכן הוא מוגדר רק לערכים ספציפיים כי הוא מוגדר למספר אפשרויות מוגבל.

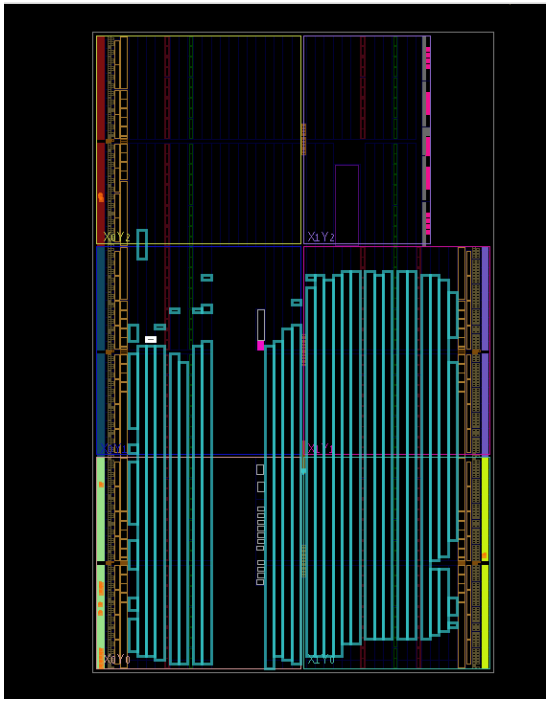
3.3.3 Place & Route – מיקום וחיוט

מיקום וחיוט - במהלך שלב זה כלי התוכנה לוקח את ה-netlist שנוצר בשלב הסינתזה וממפה פיזית את הלוגיקה הדיגיטלית על המשאבים הפיזיים של ה-FPGA.

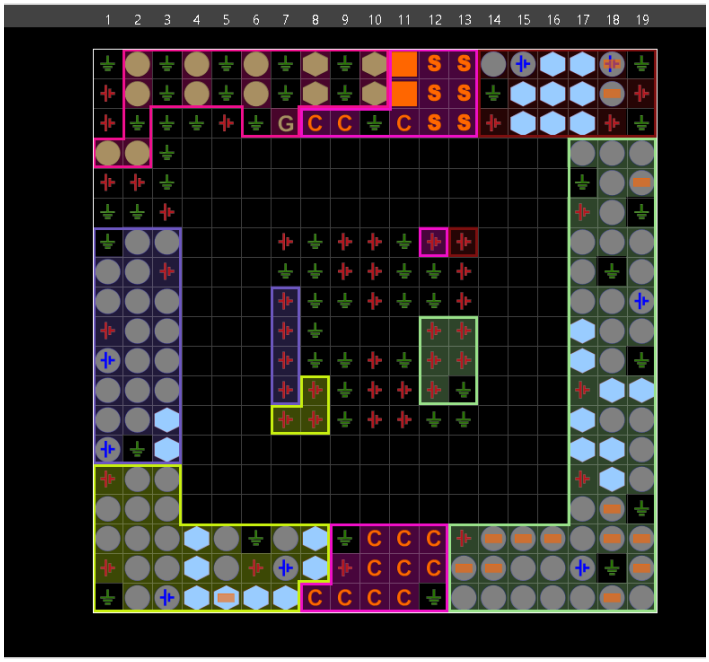
בתחילה התוכנה קובעת היכן כל אלמנט לוגי צריך להיות ממוקם ברכיב FPGA. זה כרוך בבחירת תאים לוגיים ספציפיים ודלגלגים כדי למקסם את הביצועים ולמזער את עיכובי החיוט. המטרה היא לייעל את המיקום עבור גורמים כמו התפשטות האותות, ניצול השטח ועמידה במגבלות תזמון.

לאחר מיקום האלמנטים הלוגיים, התוכנה קובעת כיצד לחבר אותם באמצעות משאבי החיוט הזמינים, כגון חיבורים, חוטים ומתגים הניתנים לתכנות ב-FPGA. חיוט יעיל חיוני

כדי למזער עיכובים ולעמוד בדרישות התזמון. באיור 2.27 אפשר לראות את התכנון ממוקם על רכיב ה-FPGA , ובאיור 2.28 רואים את מפת הכניסות והיציאות של הרכיב.



איור 3.8 – מיקום רכיבי החומרה המשומשים לאלגוריתם.

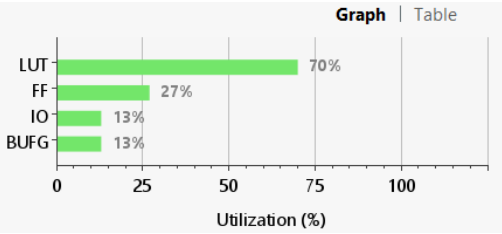


איור 3.9 – מפת הכניסות והיציאות של הרכיב.

כלי התוכנה עוקב אחר השימוש במשאבי FPGA כמו LUTs, דלגלגים, מרבבים וקשרים הדדיים כדי להבטיח ניצול יעיל. מטרתו היא למזער בזבוז משאבים ברכיב. השימוש במשאבים הסופי על הרכיב מוצג באיור 3.29.

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	14559	20800	70.00
FF	11204	41600	26.93
IO	14	106	13.21
BUFG	4	32	12.50



איור 3.10 – כלי החומרה המנוצלים לאחר מיקום וחיוט.

בנוסף, כלי התוכנה מבצע את ניתוח התזמון של התכנון כדי להבטיח שהוא עומד במגבלות הזמן ובאילוצים 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 3.11 – ניתוח התזמון של התכנון.

3.3.4 Bitstream generation and Device programming

לאחר שהמיקום והחיוט מצליחים, כלי התוכנה מייצר את קובץ ה-Bitstream שמגדיר את ה-FPGA עם הלוגיקה המתוכננת, כך שיהיה ניתן להטמיע את התכנון על רכיב ה-FPGA.

3.4 יצירת קוד תוכנה לקישור בין החומרה למשתמש

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

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

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

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

- <u>קבלת המידע מהמשתמש:</u>
  כמוסבר <u>בפרק 2.3</u> באלגוריתם MD6 יש מידע קלט שהוא אופציונלי (המשתמש בוחר אם להכניס אותו). לכן התוכנה תשאל את המשתמש איזה מידע אופציונלי הוא בוחר להכניס ואיזה לא, כאשר המשתמש בוחר לא להכניס מידע אופציונלי התוכנה תכניס למידע האופציונלי את ערכי ברירת המחדל.

בנוסף ההודעה 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 ההגדרה", המשתמש מכניס כל מידע ומידע בגודל

- <u>יצירת מידע עזר:</u>
לאחר קבלת המידע התוכנה יוצרת 3 מיידעי עזר אשר מועברים בנוסף לרכיב החומרה:
  - keylen – אורך המפתח ב-byte.
  - Padding M – אורך ריפוד המידע להודעה.
  - Index M – מספר פונקציות הדחיסה.

- <u>המרת המידע לפורמט byte:</u>
העברת המידע דרך התקשורת הטורית UART ע"י ספריית serial בשפת Python מתאפשר רק כאשר המידע המועבר הוא בפורמט byte לכן כל המידע עובר המרה לפורמט זה.

- <u>ריפוד המידע באפסים וסידורו:</u>
כדי שכל מידע ומידע יועבר למיקומו הנכון ברכיב החומרה, כל מידע מרופד באפסים עד כדי גודלו המקסימלי.
  - 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 מההודעה והמפתח, כל מידע עובר היפוך כך שה-byte הראשון מועבר לסוף וכן הלאה, כדי להתאים את שלחית המידע לאלגוריתם.

- <u>שרשור כל המידע:</u>
לשם שליחת המידע דרך תקשורת הטורית, כל המידע משורשר אחד אחרי השני לשליחה בצורה יעילה ומהירה.

- <u>שליחת וקבלת המידע דרך התקשורת הטורית:</u>
כדי לשלוח את המידע דרך התקשורת הטורית, התוכנה תבקש מהמשתמש להכניס את כניסת ה-COM אליה מחובר ערכת הפיתוח.

לאחר הכנסת מספר ה-COM, התוכנה שולחת את המידע המשורשר ע"י שימוש בספריית serial בשפת Python, ע"י שימוש בפקודה ser.write() ומקבלת את המגובב חזרה מהמשתמש ע"י פקודת ser.read().

4 תוצאות

בפרק זה מוצגים הפעלת התכנון, הסימולציות ותוצאות הבדיקות שנועדו לאימות התכנון.

4.1 הפעלת אלגוריתם MD6

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

בהפעלת האלגוריתם יש שני חלקים מרכזיים:

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

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

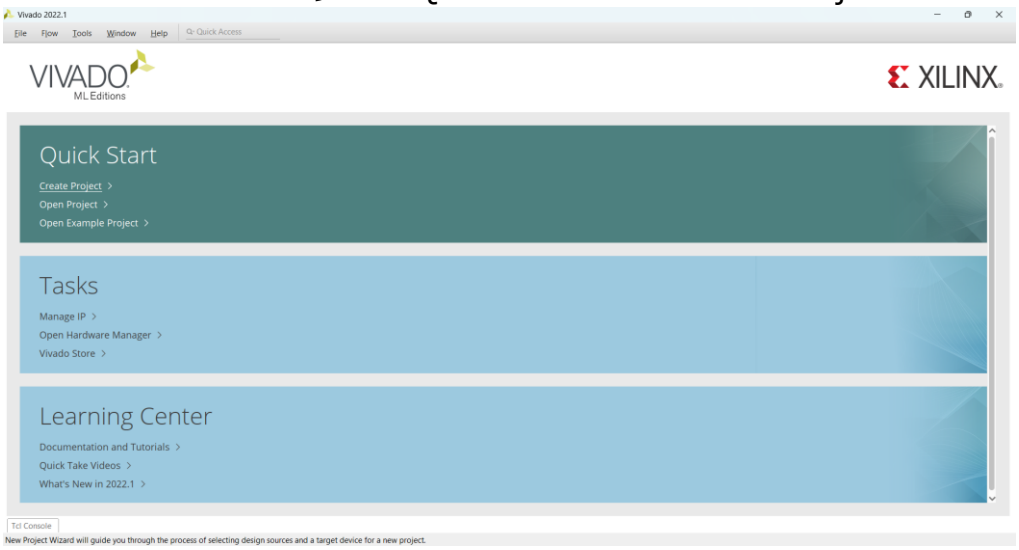
- תיקיית הקבצים של הפרויקט.
- ערכת הפיתוח Basys3.
- כלי התוכנה VIVADO.

קישור לתיקיית קבצי הפרויקט ומדריך בוידאו להפעלת האלגוריתם מצורפים בנספח א'

4.1.1 תהליך הטמעת אלגוריתם MD6 על רכיב ה-FPGA

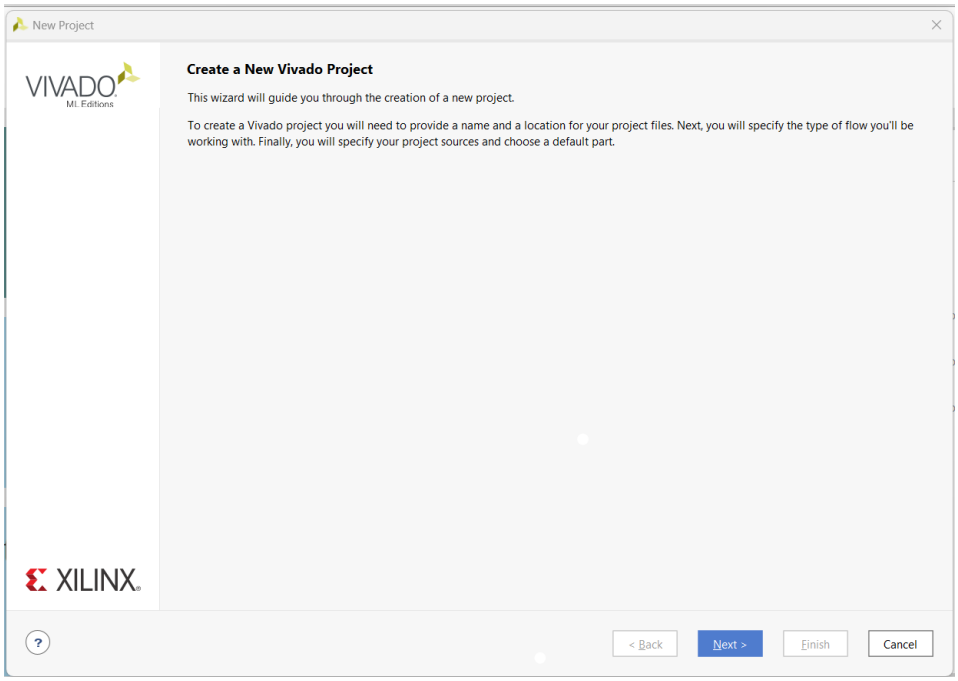
פרק זה מפרט את תהליך הטמעת אלגוריתם MD6 על רכיב ה-FPGA.

- תחילה יש להפעיל את כלי התוכנה VIVADO.
- יש לבחור Create Project שנמצא ב-Quick Start כמופיע באיור 4.1



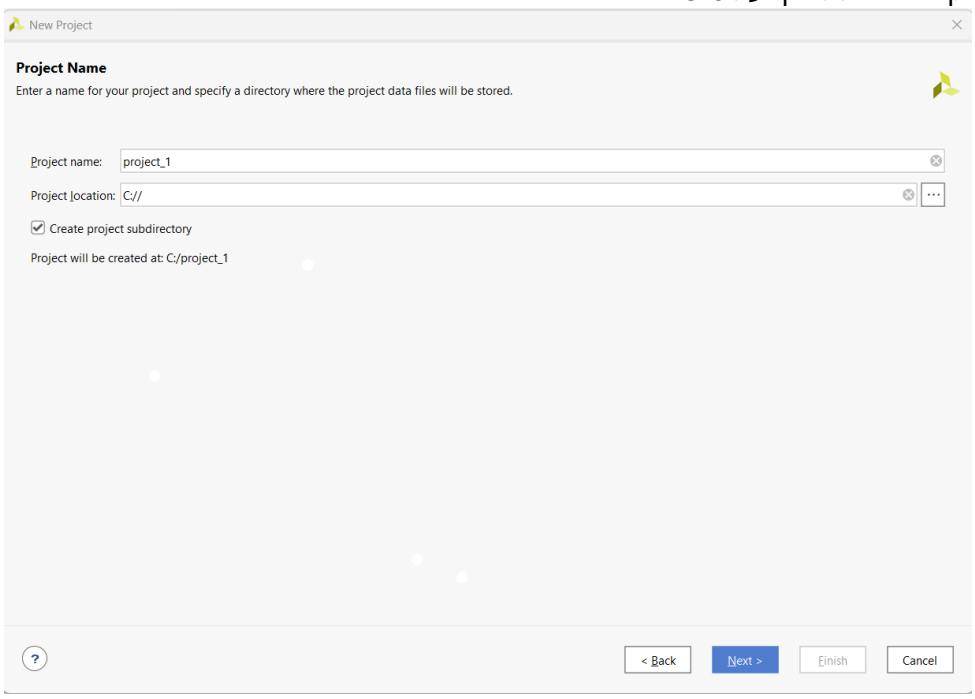
איור 4.1 – חלון הפתיחה של כלי התוכנה VIVADO.

- בחלון ה-Create a New Vivado Project המופיע באיור 4.2, יש ללחוץ על Next.



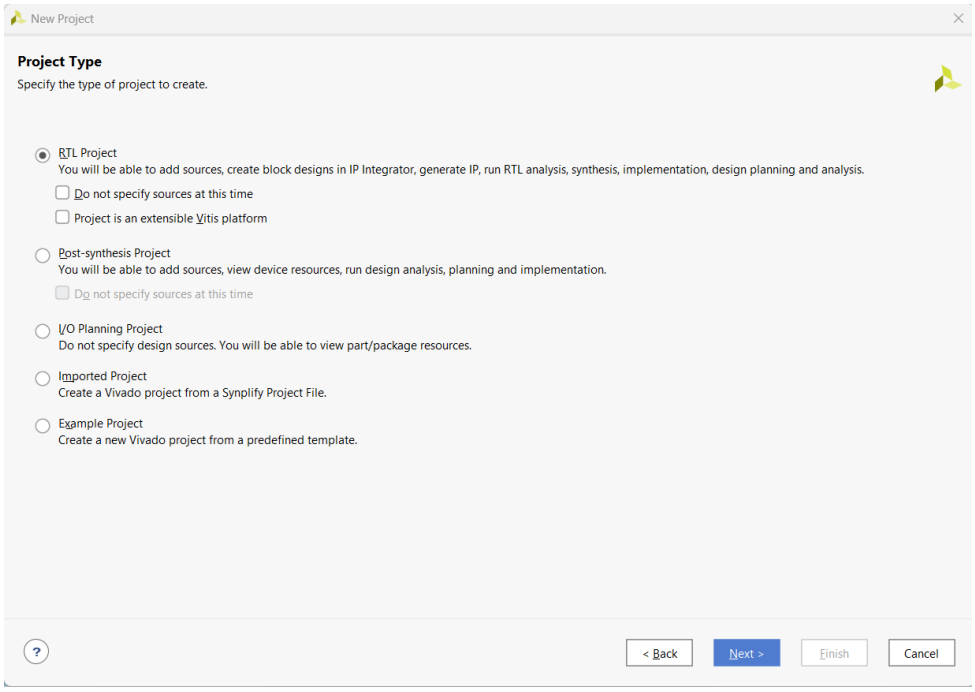
איור 4.2 – חלון ה-Create a New Vivado Project.

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



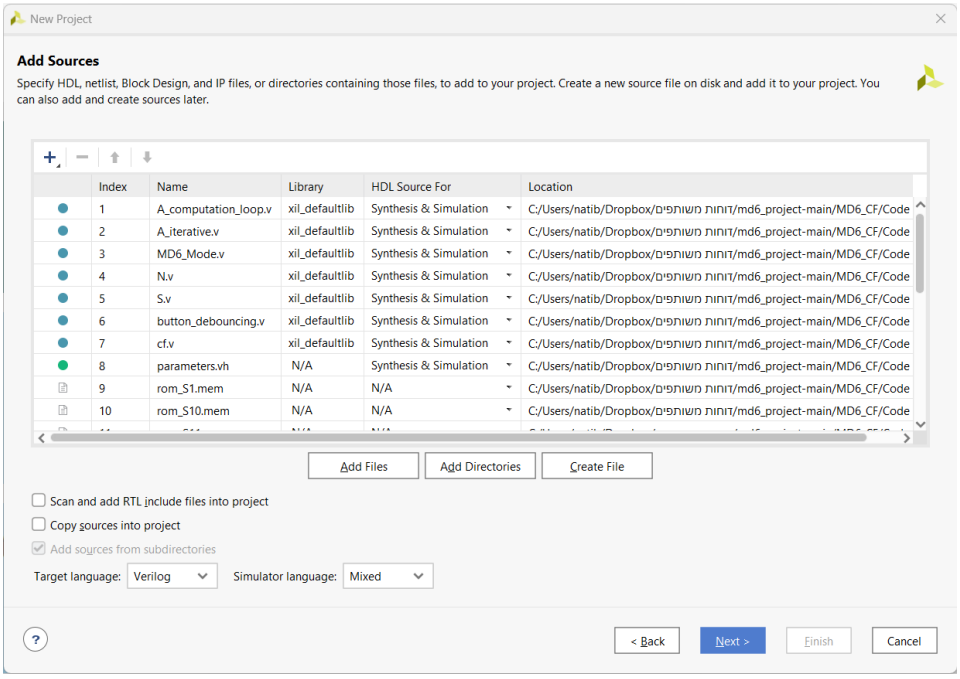
איור 4.3 – חלון ה-Project Name.

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



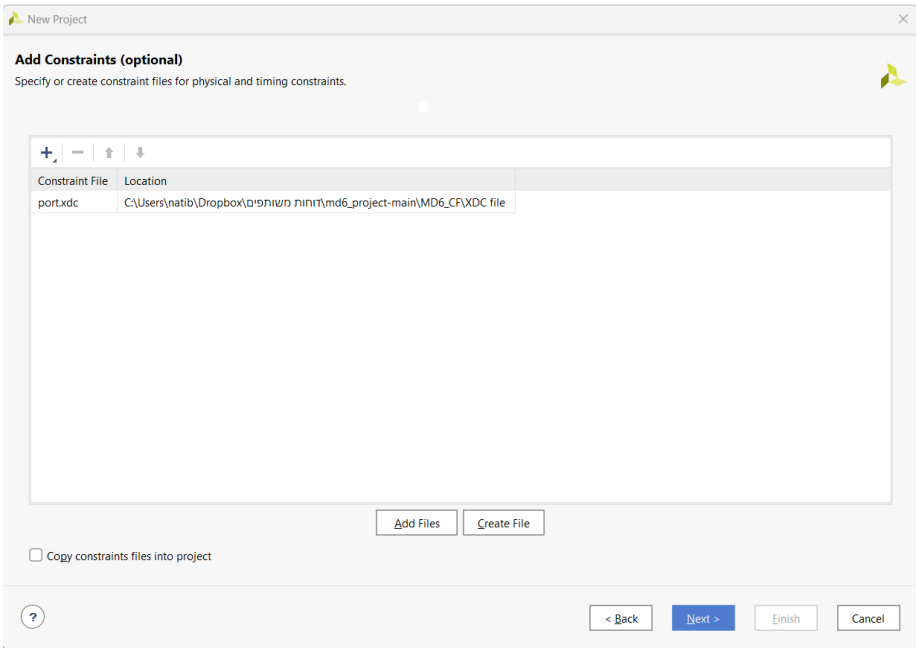
איור 4.4 – חלון ה-Project Type.

- 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.



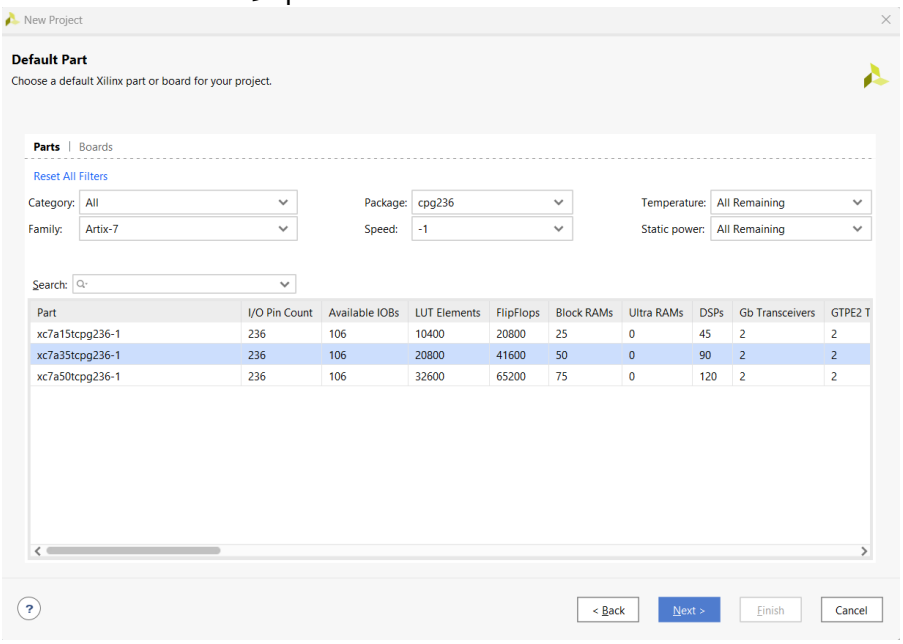
איור 4.5 – חלון ה-Add Sources.

- 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.



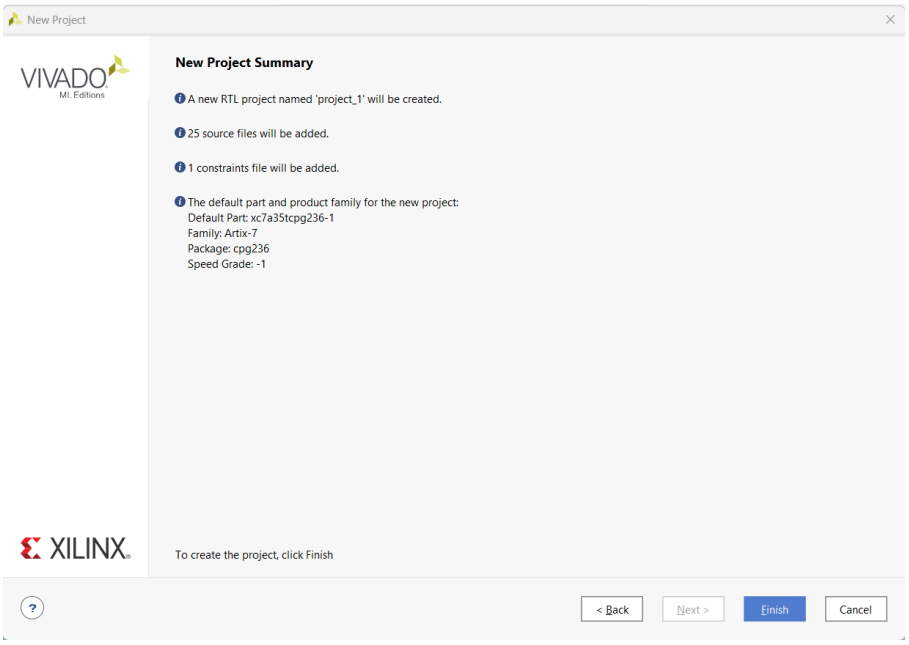
איור 4.6 – חלון ה-Add Constraints.

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



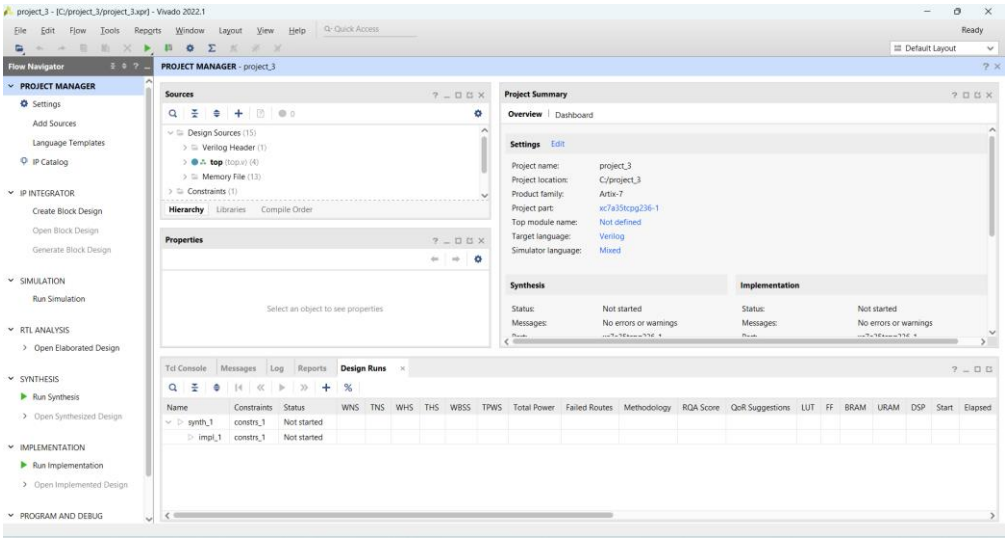
איור 4.7 – חלון ה-Default Part.

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



איור 4.8 – חלון ה-New Project Summary.

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

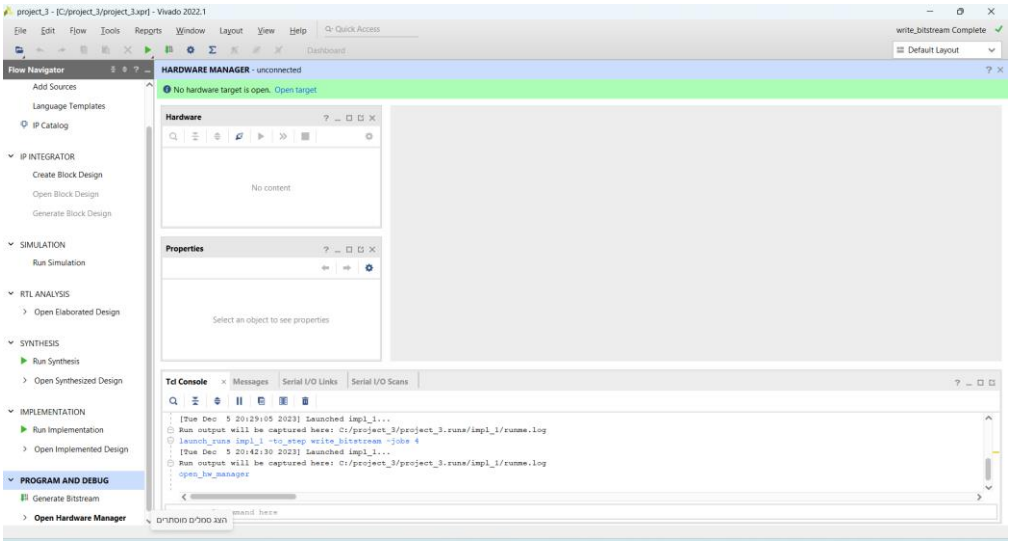


איור 4.9 – החלון הראשי של ניהול הפרויקט.

- יש להפעיל תחילה את הסינתזה ע"י לחיצה על Run Synthesis, שנמצא תחת הכותרת SYNTHESIS ב-PROJECT MANAGER.
- אחרי סיום תהליך הסינתזה, יש להפעיל את תהליך ה"מיקום וחיווט" ע"י לחיצה על Run Implementation, שנמצא תחת הכותרת IMPLEMENTATION ב-PROJECT MANAGER.
- אחרי סיום המיקום והחיווט, יש לייצר את קובץ ה-Bitstream ע"י לחיצה על Generate Bitstream, שנמצא תחת הכותרת PROGRAM AND DEBUG ב-PROJECT MANAGER.
- אחרי סיום ייצור קובץ ה-Bitstream, כדי להתחיל את תהליך ההטמעה על רכיב החומרה יש לפתוח את ה-Hardware Manager ע"י לחיצה על Open Hardware

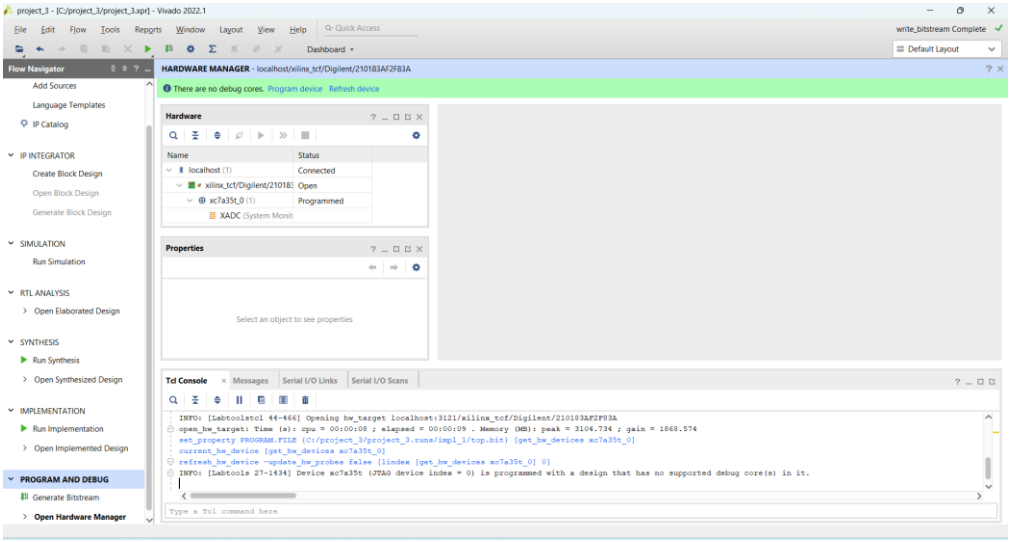
Manager, שנמצא תחת הכותרת PROGRAM AND DEBUG ב- PROJECT MANAGER.

- לאחר פתיחת חלון ה-HARDWARE MANAGER המופיע באיור 4.10, כדי להתחבר עם כלי החומרה בו מוטמע התכנון, יש ללחוץ על Open target הנמצא תחת כותרת החלון (בשורה הירוקה), ואז בחלון הקטן שיפתח יש ללחוץ על Auto Connect.



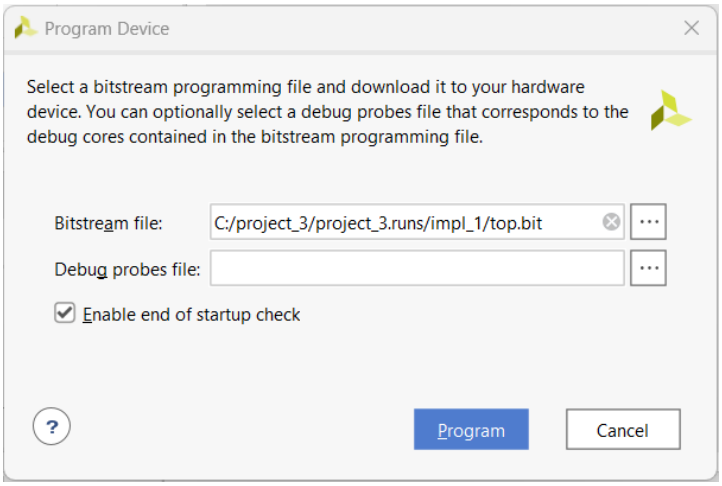
איור 4.10 – חלון ה-HARDWARE MANAGER.

- לאחר שכלי התוכנה מצא את הרכיב המיועד להטמעה, יש ללחוץ על Program device הנמצא תחת כותרת החלון כמופיע באיור 4.11.



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

- לאחר הלחיצה על Program device יפתח חלון בו מופיע קובץ Bitstream המיועד להטמעה על רכיב החומרה כמופיע באיור 4.12. יש ללחוץ על Program לצורך להטמעה.



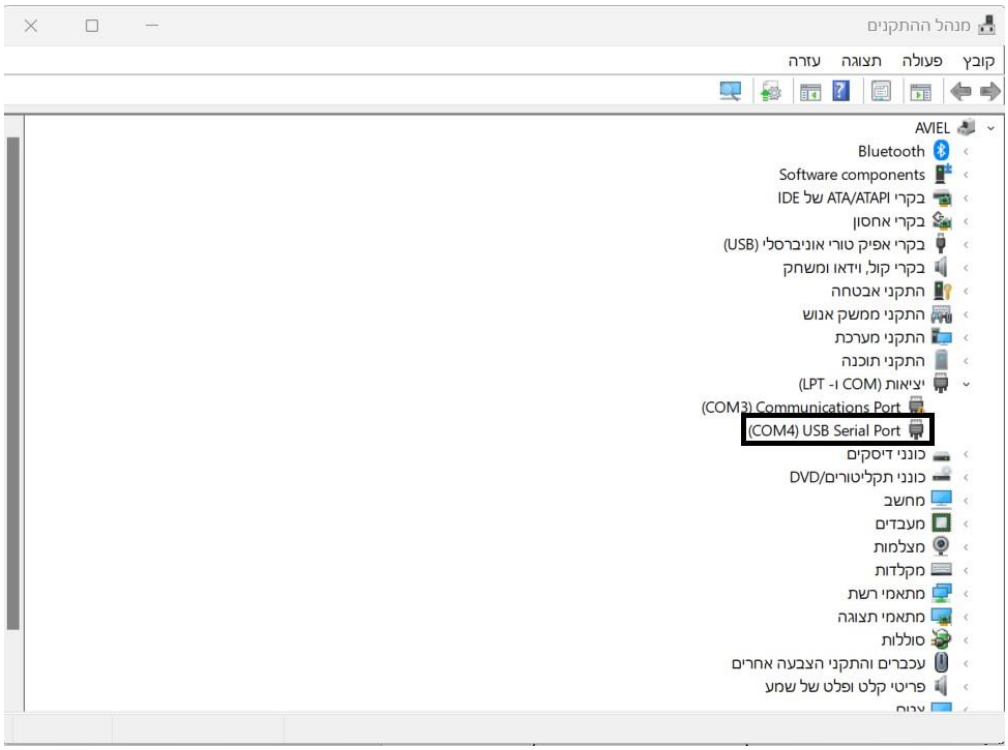
איור 4.12 – חלון ה-Program Device להטמעת התכנון.

- כעת תכנון אלגוריתם MD6 הוטמע בלוח החומרה וניתן לגבב דרכו מידע.

4.1.2 תהליך הפעלת גיבוב המידע על רכיב ה-FPGA

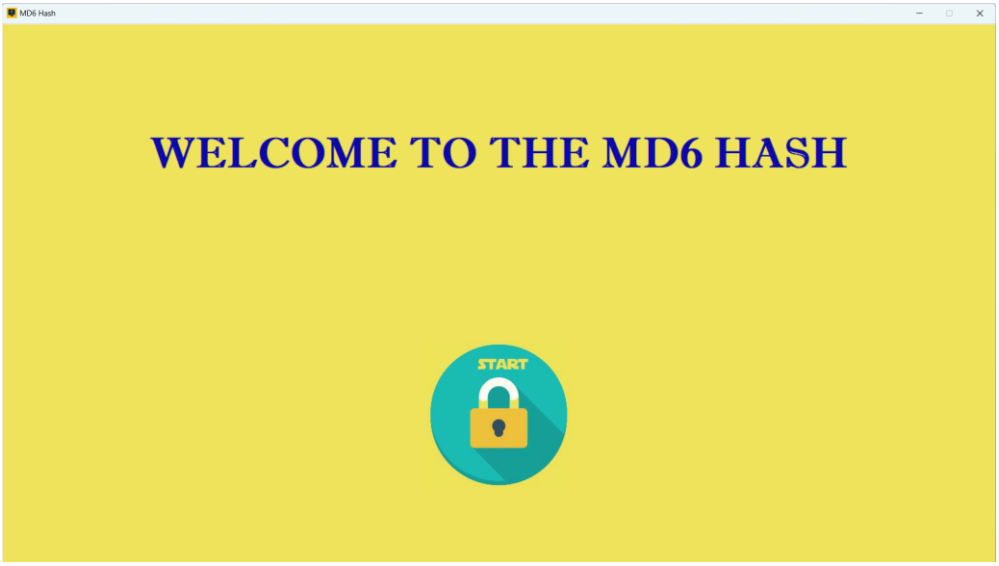
פרק זה מפרט את תהליך גיבוב אלגוריתם MD6 על רכיב ה-FPGA.

- קודם החלת הגיבוב יש לבדוק את מספר יציאת ה-COM בו מחובר רכיב החומרה. ניתן לבדוק זאת במנהל ההתקנים, כמוצג באיור 4.13 .



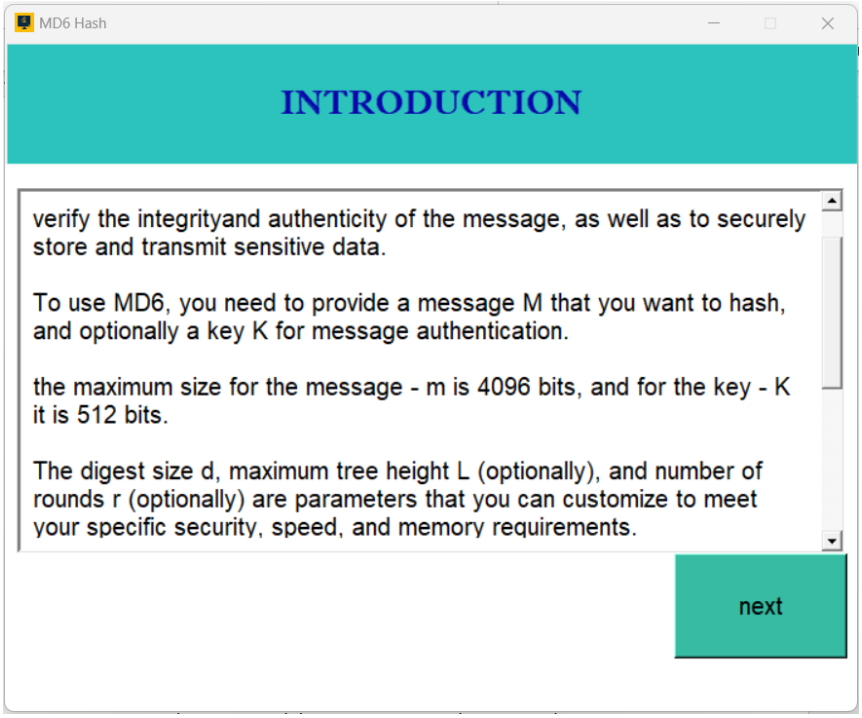
איור 4.13 – מציאת מספר ה-COM במנהל ההתקנים.

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



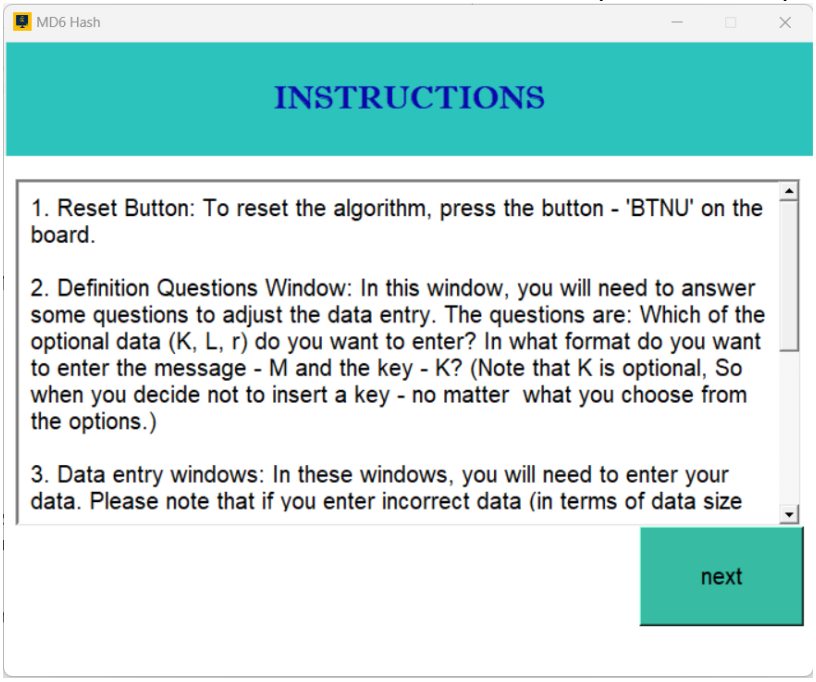
איור 4.14 – חלון ההתחלה של תוכנית הגיבוב.

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



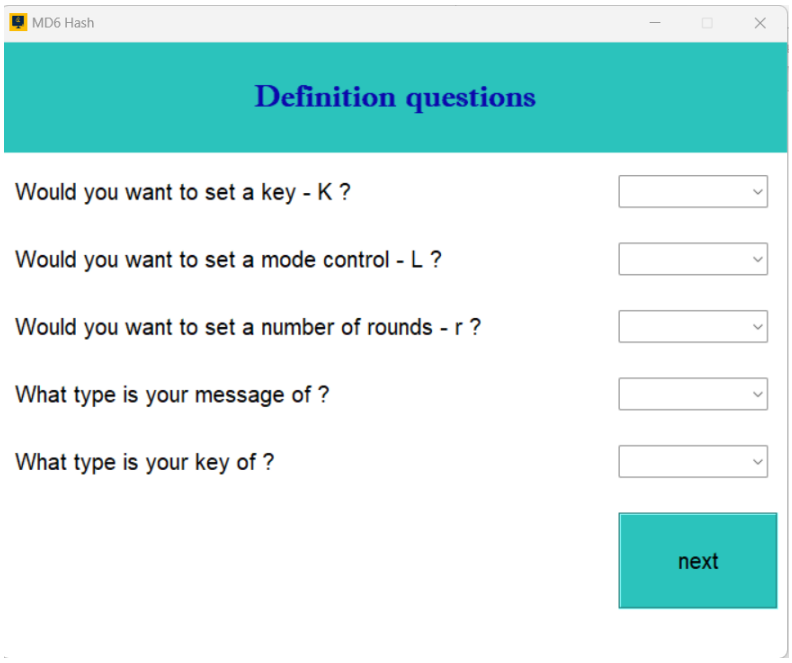
איור 4.15 – חלון ה-INTRODUCTION (המבוא).

- לאחר לחיצה על כפתור ה-next, החלון הבא הוא חלון ה-INSTRUCTIONS בו מפורט בקצרה על תהליך הגיבוב בתוכנית זו כמופיע באיור 4.16.



איור 4.16 – חלון ה-INSTRUCTION (ההוראות).

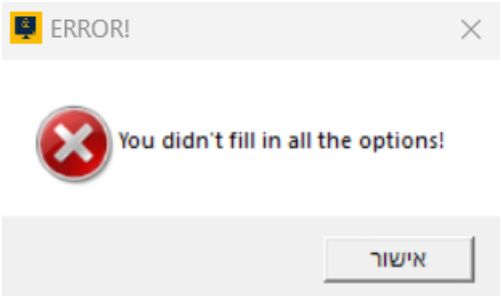
- לאחר לחיצה על כפתור ה-next, החלון הבא הוא חלון ה-Definition questions, כמוצג באיור 4.17, שבו נבדק אם נדרש מידע אופציונלי ובאיזה פורמט להכניס את ההודעה והמפתח.



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

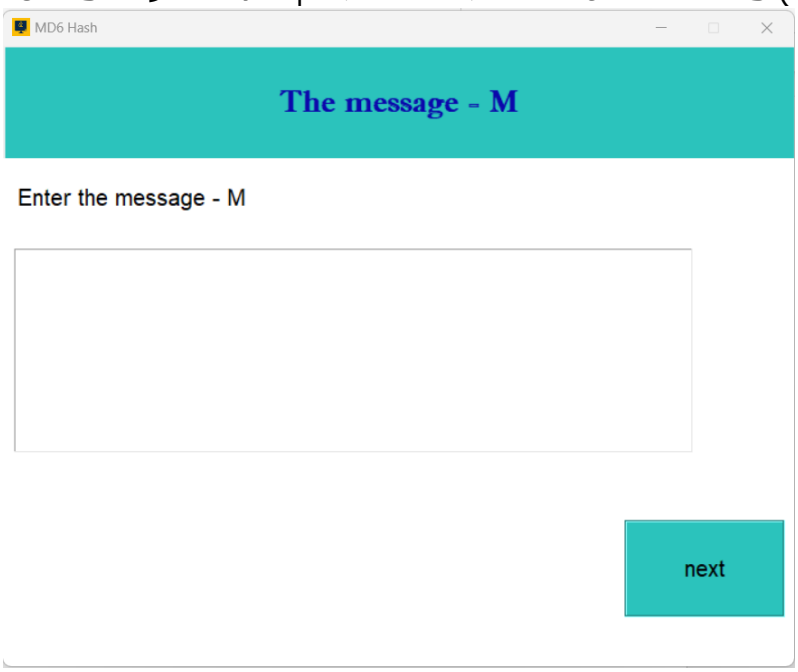
- לפני לחיצה על כפתור ה-Next, יש למלא את התשובות לכל שאלות ההגדרה, אחרת ישלח חלון שגיאה, כמוצג באיור 4.18.

הודעת שגיאה כזו תשלח בכל פעם שלא יבחרו האופציות האפשריות בחלון מסוים.



איור 4.18 – חלון השגיאה על אי מילוי תאים.

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



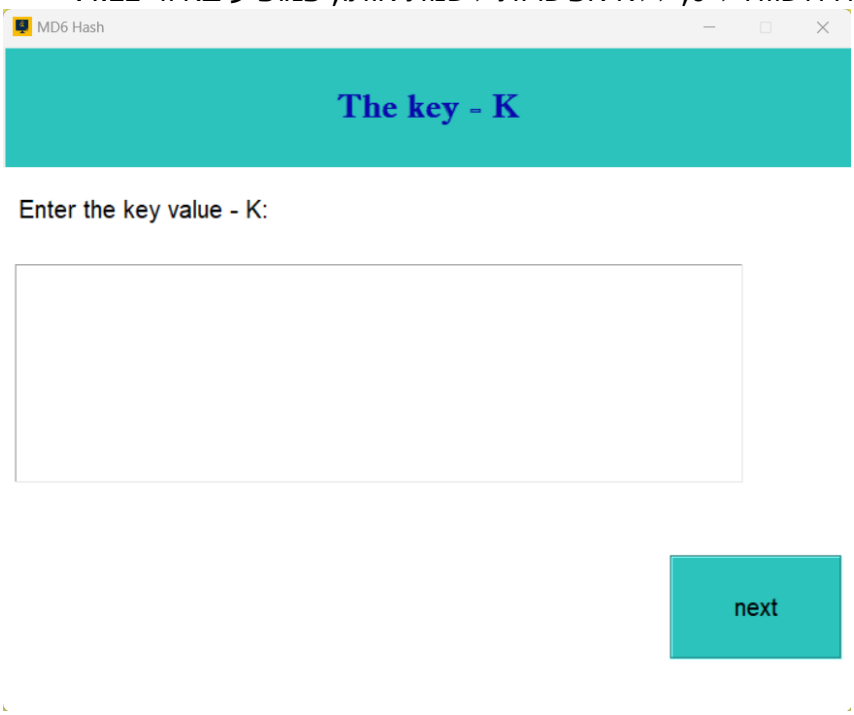
איור 4.19 – חלון ה-Message (ההודעה).

- יש לשים לב – אם ההודעה איננה בגודל הנכון או איננה בפורמט הנכון, התוכנית תשלח הודעת שגיאה כמוצג באיור 4.20.

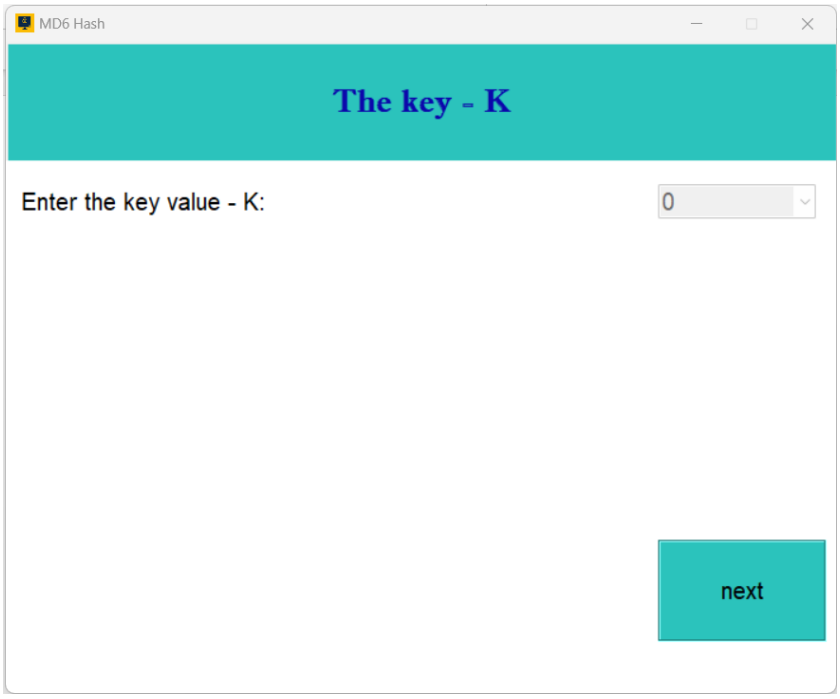
הודעת שגיאה כזו תשלח בכל פעם שלא נבחרות האופציות האפשריות בחלון מסוים.

איור 4.20 – חלון השגיאה על הכנסת מידע לא תואם.

- לאחר מילוי ההודעה ולחיצה על לחצן ה-next, יפתח חלון ה-Key. במידה ויסופק מפתח, יפתח חלון שבו יוכל לספק מפתח, כמו חלון ה-Message, כמופיע באיור 4.21. ולא, יפתח חלון שמראה את ערך ברירת המחדל של המפתח השווה ל-0, ללא אפשרות לשנות אותו, כמופיע באיור 4.22.



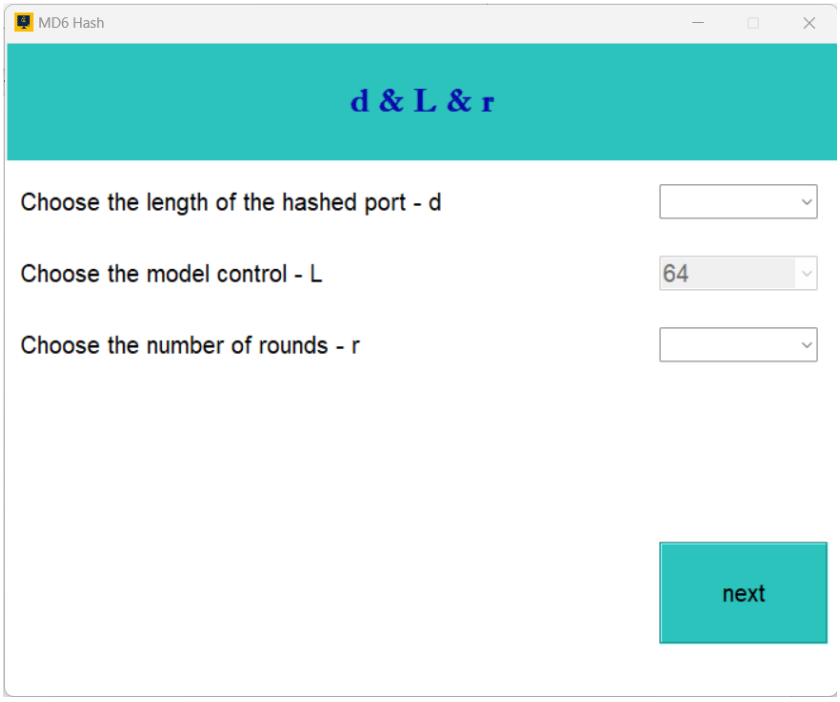
איור 4.21 – חלון ה-Key (המפתח) עם תיבת הכנסת המידע.



איור 4.22 – חלון ה-Key (המפתח) עם ערך ברירת המחדל.

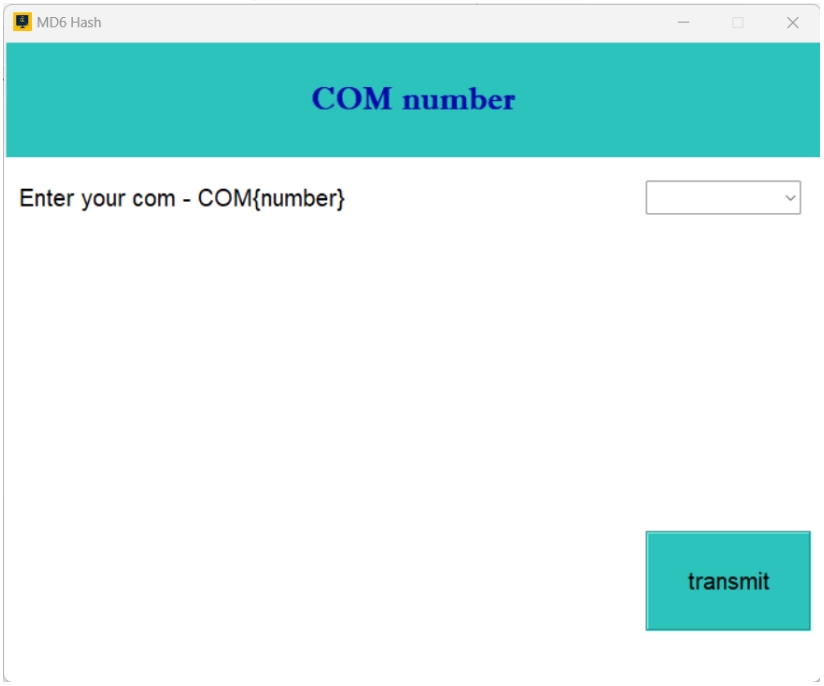
- לאחר לחיצה על לחצן ה-next, יפתח חלון ה-r & L & d, כמוצג באיור 4.23, בו יש לספק את אורך המידע המגובב, פרמטר בקרת המצב ומספר הסיבובים.

פרמטר בקרת המצב ומספר הסיבובים הם כניסות אופציונליות, לכן במידה ולא יסופקו, יופיעו ערך ברירת המחדל שלהם, ללא אפשרות שינוי. לדוגמא, כפי שניתן לראות באיור 4.23, לא סופק פרמטר בקרת המצב, אשר על כן מופיע ערך ברירת המחדל שלו, ללא אפשרות שינוי.



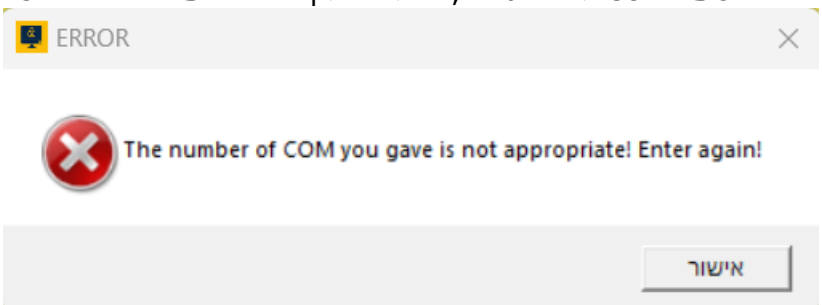
איור 4.23 – חלון ה-d & L & r.

- לאחר הכנסת המידע ולחיצה על לחצן ה-next, יפתח חלון ה-COM, בו יש לספק את מספר ה-COM אליו מחובר רכיב החומרה, כמוצג באיור 4.24.



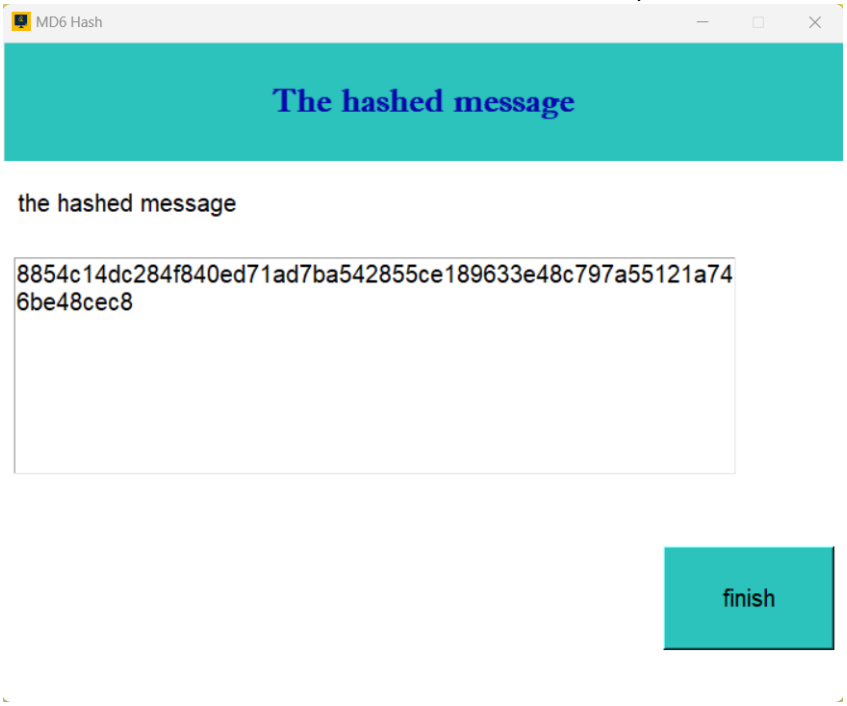
איור 4.24 – חלון ה-COM number.

- כאשר נבחר מספר COM לא מתאים, ישלח חלון שגיאה כמוצג באיור 4.25.



איור 4.25 – חלון השגיאה על הכנסת מספר COM שגוי.

- לאחר הכנסת מספר ה-COM ולחיצת על כפתור ה-transmit, המידע מועבר לגיבוב ברכיב החומרה.
- ה-Basys3 כולל נורות LED המאפשרות לתת אינדיקציה לתהליכים שמתרחשים בלוח. נורות LD0 - LD8 מורות על חלקים של מידע שנקלט, כך שהדלקה של כלל הנורות מורה על סיום קליטת המידע. לאחר מכן ניתן ללחוץ על לחצן השליחה BTNC אשר שולח את המידע המגובב חזרה לתוכנית. את מיקום הנורות והלחצנים ניתן לראות באיור 3.25.
- לאחר לחיצה על לחצן השליחה, יפתח חלון - the hashed message אשר מציג את ההודעה המגובבת, כמוצג באיור 4.26.



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

- לסגירת התוכנית יש ללחוץ על לחצן ה-finish.

4.2 אימות תכנון החומרה על ערכת הפיתוח

4.2.1 תהליך אימות תכנון החומרה על ערכת הפיתוח

כדי לאמת את התכנון על רכיב החומרה, קודדה תוכנה בשפת Python בשם Test vector שתפקידה ליצור וקטורי בדיקה. התוכנה יוצרת מידע אקראי (תואם לקריטריונים של האלגוריתם) שעובר גיבוב בתכנון התוכנה ובתכנון החומרה. לאחר מכן 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. ובכן תהליך זה נועד לאמת את נכונות הגיבוב. כמו כן, ניתן להשוות את הביצועים של כל התכנון.

שימוש בקוד ה-Python להפעלת התוכנית דורש להתקין Python ועוד ספריות נוספות שנצרכות בשביל לקמפל את הקוד. לשם כך נוצר קובץ exe כך שלא נצרך להתקין שום דבר נוסף.

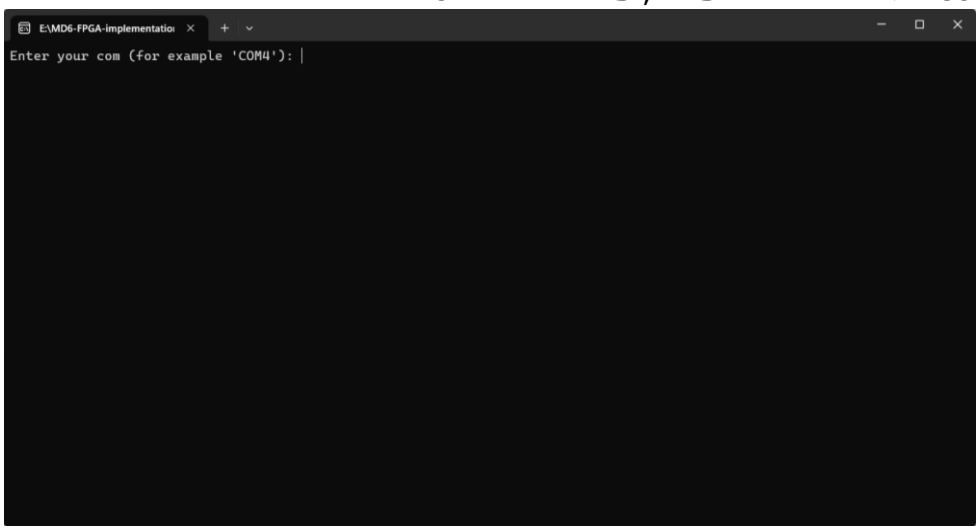
4.2.2 הפעלת אימות תכנון החומרה על ערכת הפיתוח

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

- תיקיית הקבצים של הפרויקט.
- כלי התוכנה VIVADO.

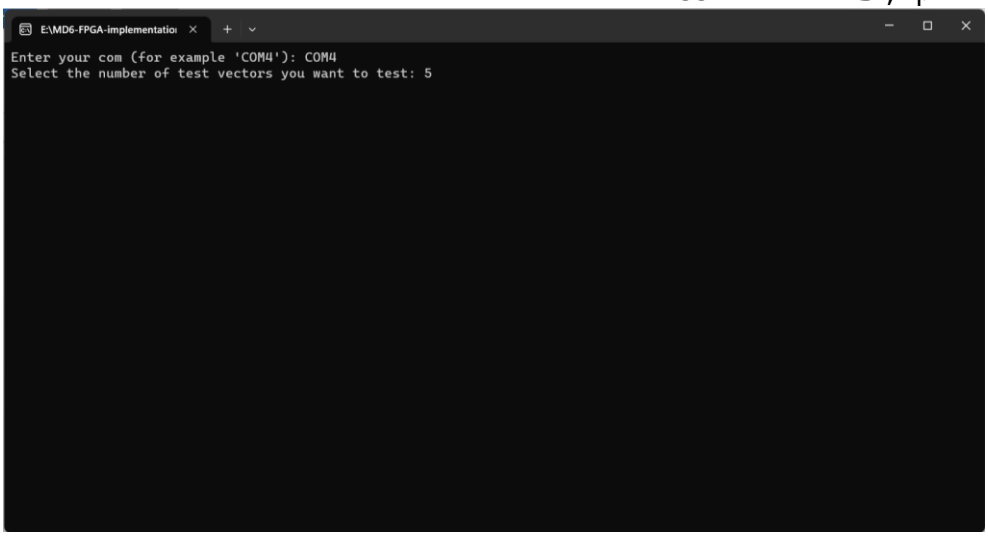
קישור לתיקיית קבצי הפרויקט להפעלת האלגוריתם מצורפים בנספח א.

- בתחילה יש להטמיע את התכנון על רכיב החומרה. ניתן למצוא את התהליך המפורט בפרק 4.1.1.
- בנוסף, יש לבדוק את מספר יציאת ה-COM בו מחובר רכיב החומרה במנהל ההתקנים, כמוצג באיור 4.13.
- יש להפעיל את קובץ הפעלת אימות תכנון החומרה על ערכת הפיתוח Test_vector.exe הנמצא ב- md6_project-main\Software\Test Vectors Software\GUI App בספריית הפרויקט.
- לאחר הרצת קובץ ההפעלה יפתח חלון והתוכנה תבקש להכניס את כניסת ה-COM אליה מחובר המכשיר, כמוצג באיור 4.32.



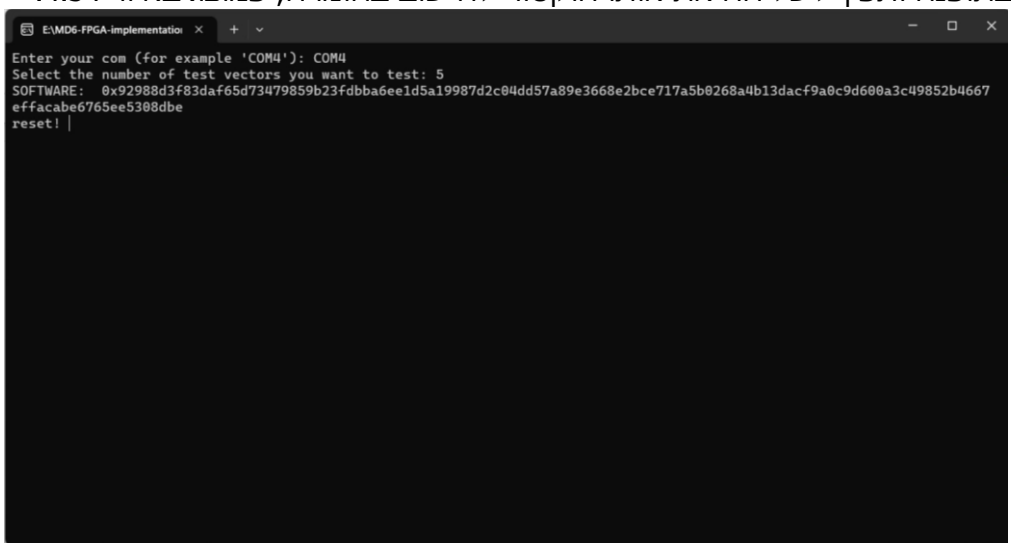
איור 4.27 – הכנסת מספר כניסת ה-COM.

- אחרי הכנסת מספר ה-COM, התוכנה תבקש להכניס את מספר וקטורי הבדיקה, כמוצג באיור 4.33.



איור 4.28 – הכנסת מספר וקטורי הבדיקה.

- לאחר הכנסת מספר וקטורי הבדיקה, התוכנה תחשב את הוקטור הראשון בתוכנה ותכין לשליחה את אותו הוקטור לחישוב בחומרה, כמוצג באיור 4.34.



איור 4.29 – תצוגת הוקטור הראשון המגובב בתוכנה.

- בשביל לקבל את גיבוב החומרה בתכנון, יש ללחוץ תחילה על כפתור ה-reset (BTNU) ולאחריו enter. כאשר כל 9 הנורות יידלקו, יש ללחוץ על כפתור השליחה (BTNC) ויתקבל המידע המגובב. ניתן לראות את מיקומם באיור 3.25. בנוסף, בשורה מתחת מתקבל גיבוב של התכנון בתוכנה, כמוצג באיור 4.35.

```
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:  0x0d0e5b3c068453fe5096ce887da02c1e6db5d78681c6cf90a5491f0625dc9f2006891e09957740174b8ede2a2766e78e
reset!
```

איור 4.30 – תצוגת וקטור החומרה שהתקבל.

- לאחר סיום קבלת כל הוקטורים, המסך ייסגר והתוכנה תייצר קובץ csv שמשווה את הוקטורים בתוכנה ובחומרה וקובע אם הם שווים (true או false), כמוצג באיור 4.36.

	A	B	C	D	E	F	G
1	Vector	Software	Hardware	Comparison Result			
2	vector-1	0x92988d3	0x92988d3	TRUE			
3	vector-2	0x0d0e5b3	0x0d0e5b3	TRUE			
4	vector-3	0x6119d4a	0x6119d4a	TRUE			
5	vector-4	0x36013f7	0x36013f7	TRUE			
6	vector-5	0x547eae8	0x547eae8	TRUE			
7							
8							
9							
10							
11							
12							
13							

איור 4.31 – קובץ csv להשוואה בין התוצאות.

5 דיונים

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 יעילות תכנון האלגוריתם בחומרה ובתוכנה.

5.1 השוואה לספרות

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.

קלט האלגוריתם לפי הדוגמא הראשונה בדו"ח האלגוריתם של Ronald Linn Rivest מופיע באיור 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 מופיעות באיור 5.2.

```
8854c14dc284f840ed71ad7ba542855ce189633e48c797a55121a746be48cec8 -Mabc

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

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

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

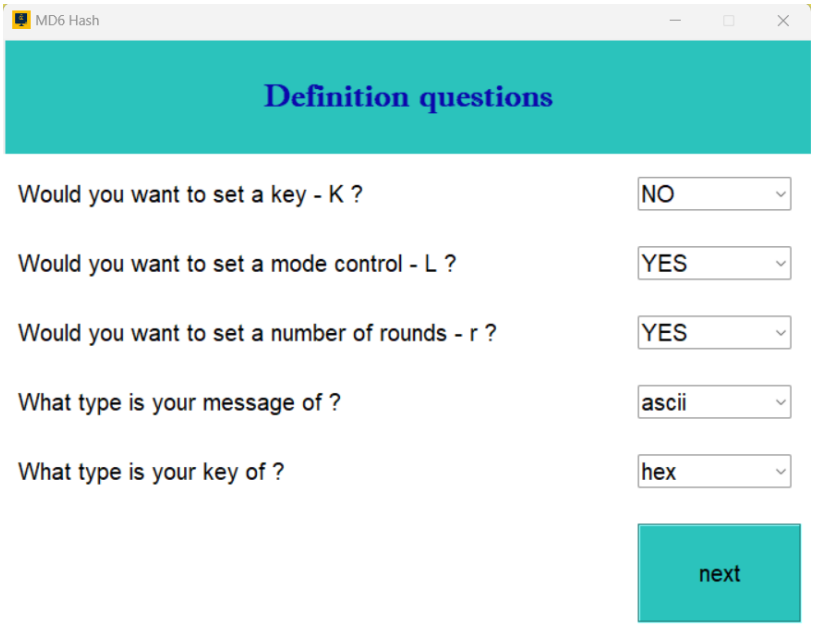
טבלה 6: המידע הנכנס לאלגוריתם בסימולציה

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

5.1.1 השוואת תוצאות אימות תכנון החומרה על ערכת הפיתוח

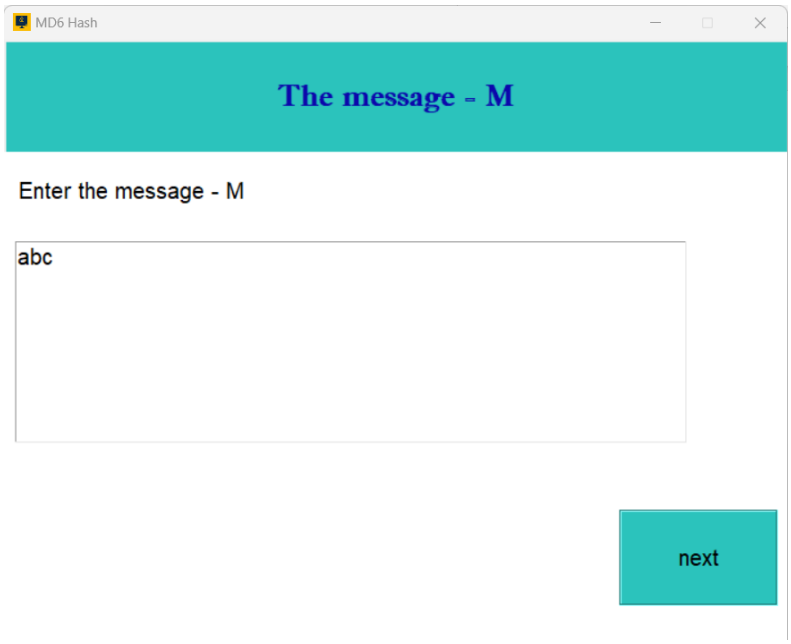
כדי להשוות בחומרה יש להשתמש בקובץ ההפעלה של האלגוריתם MD6_CF.exe ויש להכניס את המידע בצורה הבאה:

- תחילה יש לענות על שאלות ההגדרה בצורה נכונה כך שיתאימו לדוגמא הראשונה בדו"ח האלגוריתם, כמוצג באיור 5.5. יש לשים לב שלמרות שבדוגמא אין מפתח, בכל זאת יש לבחור סוג פורמט.



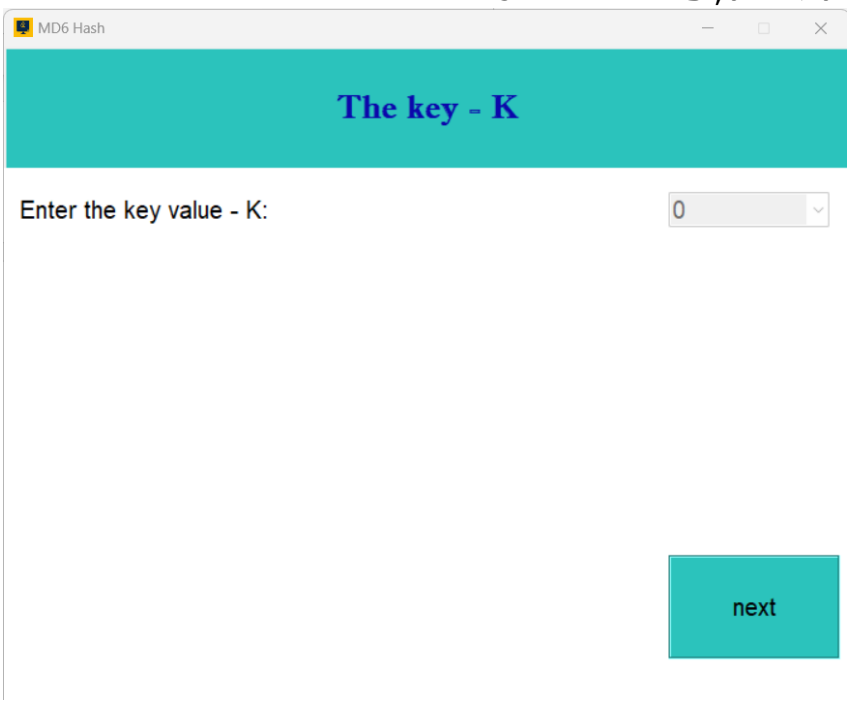
איור 5.3 – התאמת שאלות ההגדרה לדוגמא הראשונה.

- לאחר מכן, יש להכניס את ההודעה בפורמט ASCII כמבוקש, כמופיע באיור 5.6.



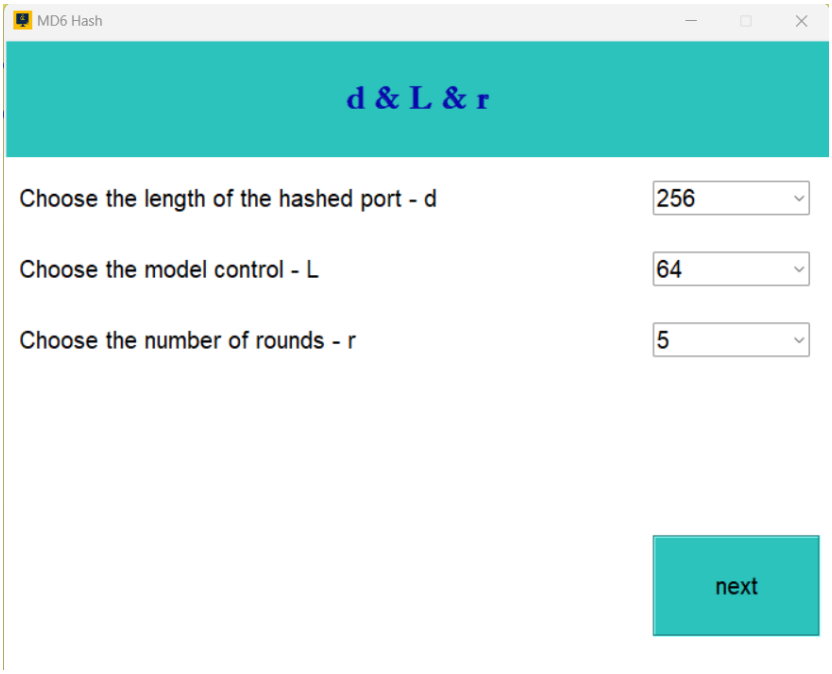
איור 5.4 – התאמת ההודעה לדוגמא הראשונה.

- משום שבדוגמא אין מפתח, בחלון המפתח יופיע ערך ברירת המחדל ללא אפשרות לשנותו, כמוצג באיור 5.7.



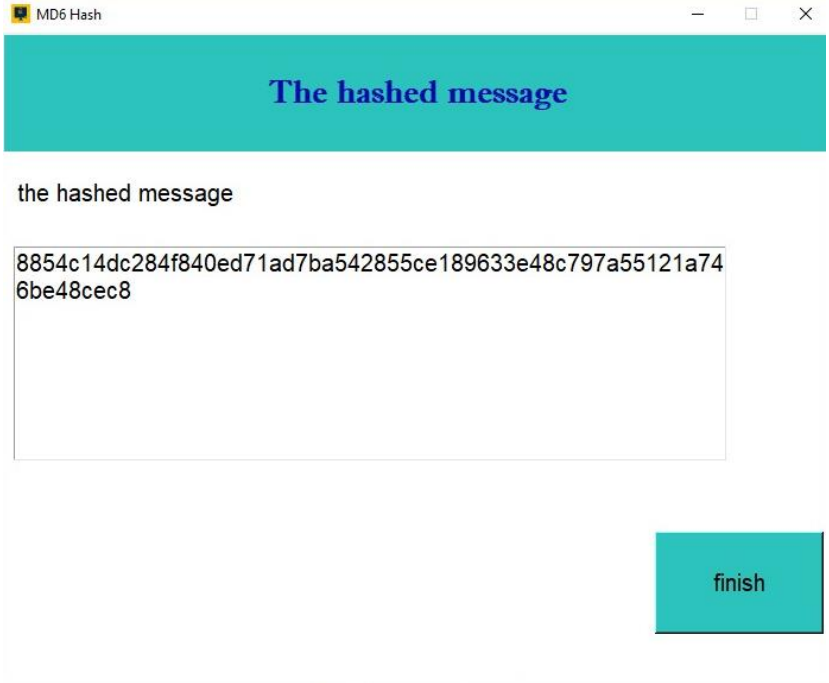
איור 5.5 – התאמת המפתח לדוגמא הראשונה.

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



איור 5.6 – התאמת d & L & r לדוגמא הראשונה.

- לאחר התאמת מידע הקלט לדוגמא הראשונה, כפי שניתן לראות באיור 5.9, אכן התקבלה אותה תוצאת גיבוב.



איור 5.7 – תוצאת הגיבוב של הדוגמא הראשונה בחומרה.

סרטון המראה את הרצת הדוגמא הראשונה של דו"ח האלגוריתם ניתן למצוא בנספח א.

5.2 השוואה לתוכנה

כדי להראות את יעילות עיבוד המידע בתכנון, נשווה את תכנון האלגוריתם בחומרה לתכנון בתוכנה.

במהלך הפרויקט נוצר תכנון שנוצר בתוכנה בשפת Python לאימות נכונות תוצאות הגיבוב. אך בדיקת יעילות 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 ויותר שייך 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 משוואה 2:

$$\max \text{Freq} = \frac{1}{Crystal\ Oscillator - WNS} \quad (2)$$

כאשר:
Crystal Oscillator – זמן המחזור של מתנד הקריסטל של רכיב החומרה השווה ל-10ns, כמפורט בטבלה 5 בפרק 3.1.

WNS (Worst Negative Slack) – מייצג את הנתיב הקריטי, שהוא הנתיב עם מרווח התזמון הנמוך ביותר. הנתיב הקריטי קובע את התדירות המקסימלית שבה מעגל דיגיטלי יכול לפעול תוך עמידה במגבלות התזמון.

חישוב ה-WNS נעשה בשלב המיקום והחיווט של התכנון ומוצג באיור 3.11 בפרק 3.3.3.

לפי נתונים אלו התדר המקסימלי יוצא: $\max \text{Freq} = \frac{1}{10ns-0.171ns} = 101.74MHz$.

לאחר מציאת התדר המקסימלי יש לחשב את קצב עיבוד המידע (throughput) של האלגוריתם לפי משוואה 3.

$$Throughput = \frac{num\ of\ bits}{num\ of\ rounds} \cdot max\ freq \tag{3}$$

תחילה יש לחשב את קצב עיבוד המידע המינימלי, בו נכללים מספר הביטים של ההודעה לגיבוב בלבד ומספר הסיבובים של האלגוריתם בלבד ללא התקשורת הטורית. יש לקבוע את גודל המידע המקסימלי ואת מספר הסיבובים המקסימלי כדי להראות את הבדלי קצב עיבוד המידע ב-*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:

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

לאחר חישוב העיבוד המינימלי, יש לחשב את קצב טיפול המידע אשר כולל את המידע, מידע עזר והמידע המגובב. בנוסף לכך, יש להתחשב בסיבובי השעון של התקשורת 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 התקשורת הטורית – 19234 סיבובים.

מכאן, קצב טיפול המידע שהתקבל הינו:

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

- <u>חישוב קצב עיבוד המידע בתוכנה:</u>

כדי לחשב את קצב עיבוד המידע בתוכנה באלגוריתם MD6 בשפת C, יש להשתמש בספריית sys/time.h כדי לחשב את הפרש הזמנים מתחילת הגיבוב עד סופו, כאשר התוצאה המתקבלת כוללת את מספר הסיבובים ואת תדר ההרצה של התוכנית.

קצב עיבוד המידע שהתקבל הוא:

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

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

קצב עיבוד המידע בתכנון זה אכן לא מספיק מצדיק את המעבר למימוש האלגוריתם בחומרה. אך במידה וימומש האלגוריתם ע"פ תכנון זה עם יותר פונקציות דחיסה, הפרשי קצב העיבוד יגדלו הרבה יותר. בזמן שבחומרה פונקציות הדחיסה עובדות במקביל וכתוצאה מכך מספר הסיבובים 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 השוואה בין מימוש חומרתי למימוש תוכנתי וזיהוי דרך שבה ניתן לשפר את יעילות התכנון על ידי ניתוח נקודות החוזק והחולשה שלהם.

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

במהלך הפרויקט עלו כמה אתגרים כגון כתיבת פרוטוקול התקשורת לצורך העברת מידע לחומרה, משאבי חומרה מוגבלים בלוח ובעקבות כך ייעול של התכנון. האתגרים בהם נתקלנו שימשו חוויות למידה יקרות ערך והיו מכריעים בשכלול התכנון.

מתוך הכרה במגבלות, כגון כמות מינימלית של משאבי הלוח ובהתאם לכך חוסר היכולת לממש שילוב של מצבי פעולה ומספר רב יותר של פונקציות דחיסה ,מומש מצב פעולה שכיח יותר, כדוגמת מצב פעולה מקבילי. יש לקחת בחשבון מגבלות אלו באיטרציות עתידיות של הפרויקט.

מטרות הפרויקט הושגו ברמה גבוהה תוך שאיפה לתרום ידע רב ערך לתחום ההצפנה ואבטחת המידע. הפרויקט תורם בהיותו מספק דרך יחודית למימוש אלגוריתם MD6 בחומרה בצורה יעילה וכמו כן תורם בהבנת היתרון של מימוש חומרתי על פני מימוש תוכנתי. הממצאים ממלאים פער מכריע בהבנת ההשלכות המעשיות של יעילות קצב העברת המידע בחומרה.

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

עם סיום הפרויקט הזה, מודגש הפוטנציאל של יישום אלגוריתם גיבוב בחומרה בעיצוב עתיד תחום ההצפנה ואבטחת המידע. התוצאות החיוביות שנצפו לא רק מאשרות את היעדים הראשוניים אלא גם מעוררות מחויבות מתמשכת לפתרונות חדשניים המשפרים את אבטחת המידע.

המרכז האקדמי לב

מקורות מידע ומאמרים

[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.Il. 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)

המרכז האקדמי לב

נספח א – קישורים למסמכי וקבצי הפרויקט

- קישור לתיקיית קבצי הפרויקט:
  aviel207/Final_Project (github.com)
- קישור לקובץ ה-README של תיקיית הפרויקט:
  Final_Project/README.md at main · aviel207/Final_Project (github.com)
- קישור לדו"ח הפרויקט המלא:
- קישור לחלק א' של דו"ח הפרויקט:
- קישור לחלק ב' של דו"ח הפרויקט:
- קישור לסרטון של הפעלת אלגוריתם MD6:
  Full Demonstration Of Implementing The CF MD6 On FPGA (youtube.com)

המרכז האקדמי לב

נספח ב – פירוט תוכן תיקיית הפרויקט

- תיקיית "Documents " – תיקיית המסמכים מכילה את דו"ח האלגוריתם שהוגש כאחת מהצעות האלגוריתמים לתחרות תקן הגיבוב SHA-3 של NIST, דו"חות הפרויקט, דו"ח הפרויקט שהוגש לתחרות מטעם AMD-Xilinx ודפי המידע של ערכת הפיתוח ה-basys3 .

- תיקיית "MD6_CF_hardware " – תיקיית קבצי התכנון של האלגוריתם הממומש בחומרה המכיל את קבצי המקור, קבצי ה-mem, קובץ ההידור, קובץ ה-XDC, וקבצי ה-test bench.

- תיקיית "MD6_CF_prototype " – תיקייה המכילה את התכנון הראשוני של פונקציית הדחיסה.

- תיקיית "MD6_CF_PAR_MODE " – תיקייה המכילה מימוש של אלגוריתם MD6 בחומרה במצב פעולה מקבילי עם שתי פונקציות דחיסה.

- תיקיית "MD6_Operating_Modes" – תיקייה המכילה את תכנון מצבי הפעולה של האלגוריתם.

- תיקיית "MD6_CF_software " – תיקייה המכילה את המימוש בתוכנה של אלגוריתם MD6 בעל פונקציה דחיסה אחת.

- תיקיית "Executable_files " – תיקייה המכילה את קבצי ההפעלה של המימוש בחומרה, הפעלת אימות תכנון החומרה על ערכת הפיתוח ואימות תכנון החומרה על סימולציית ModelSim.

נספח ג – הגשה לתחרות

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

התחרות היא תחרות עולמית בה מתחרים סטודנטים מאוניברסיטאות שונות מכל העולם 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 המנצחים והפיינליסטים:

2023 Results Gallery - XOHW (openhw.eu)