בס"ד

המרכז האקדמי לב

הפקולטה להנדסה ומדעי המחשב

החוג להנדסת חשמל ואלקטרוניקה

פרויקט גמר באלקטרוניקה

מימוש FPGA של אלגוריתם MD6 Hash

קוד RTL וסימולציה

FPGA Implementation of the MD6 Hash Algorithm

RTL Code and Simulation

מגיש: אופק שרעבי

מנחה: מר אורי שטרו

ט' טבת תשפ"ד

ירושלים

תודות

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

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, רכז הפרויקטים – על שיתוף הפעולה לאורך כל הפרויקט ועל הבחינה וההתעניינות. אנחנו אסירי תודה.

לאלכס קליין על הקצאת מחשב עוצמתי 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 שנתנו בידינו את הכלים לעמוד במשימה בכבוד בפרויקט זה ובכלל בתואר כולו.

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 את כתיבת האלגוריתם לפי מסמכי המפתחים של האלגוריתם וכמו כן בדיקה ומימוש על מערכת פיתוח הכוללת רכיב FPGA. מומשה פונקציית דחיסה ראשית של ה-MD6 בצורה יעילה על לוח ה-Basys3 של חברת Digilent Inc., המבוסס על Xilinx Artix-7 FPGA. קצב עיבוד המימוש בחומרה יותר מהיר 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

המרכז האקדמי לב

רשימת טבלאות

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 FPGA מתוכננים לבצע עיבוד מקביל, המאפשרים חישובי גיבוב מרובים להתרחש בו-זמנית. בניגוד לגיבוב מבוסס תוכנה, שבו פעולות מבוצעות ברצף של מעבד. רכיבי FPGA 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 (HSM) 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.

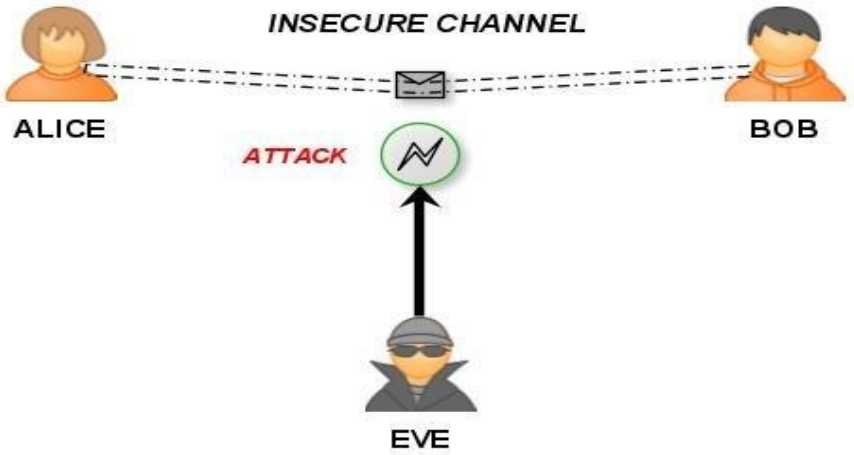
2 רקע תיאורטי

2.1 מבוא לקריפטוגרפיה

קריפטוגרפיה [1] 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 [2].

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

פונקציית גיבוב [3] 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:
  בהינתן פלט של פונקציית גיבוב, תהליך מציאת ערך הקלט המתאים הינו קשה.

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

- Second Pre-Image Resistance:
  בהינתן קלט כלשהו, קשה למצוא קלט אחר המוביל לאותו פלט של פונקציית הגיבוב.

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

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

- <u>Collision Resistance:</u>
  קשה מבחינה חישובית למצוא שני קלטים שונים שיובילו לאותו פלט של פונקציית הגיבוב.

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

  ראוי לציין כי פונקציית גיבוב "דוחסת" מידע באורך מסוים למידע מגובב באורך קטן יותר. אי לכך, לא ייתכן שלפונקציית גיבוב לא יהיו התנגשויות. מאפיין זה רק מאשש את העובדה שקשה למצוא התנגשויות אלו.

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

2.3 אלגוריתם MD6

אלגוריתם MD6 [4] 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 [5].

האלגוריתם פועל באמצעות מבנה Merkle-Damgård [6], 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 $0 \leq m < 2^{64}$ 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 $0 < d \leq 512$.

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 $0 \le L \le 64$.

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

טבלה 2 – המשך: וקטורי הקבועים – S.

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

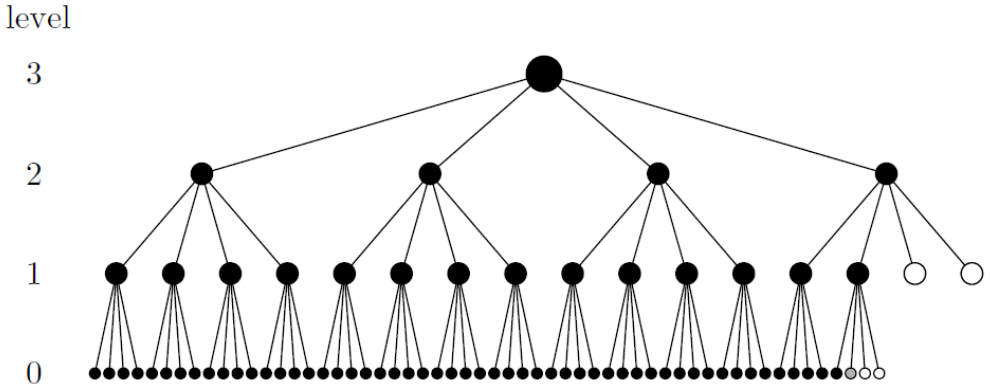
2.3.3 מצבי הפעולה של ה-MD6

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. כל 4 בלוקים מתאגדים לבלוק אחד ונכנסים לפונקציית דחיסה אחת (עליה מורחב בפרק 2.3.4) 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 יהיה בצורה המקבילה שהוא אופן הפעולה הסטנדרטי.



איור 2.2 – מצב הפעולה המקבילי [4].

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.

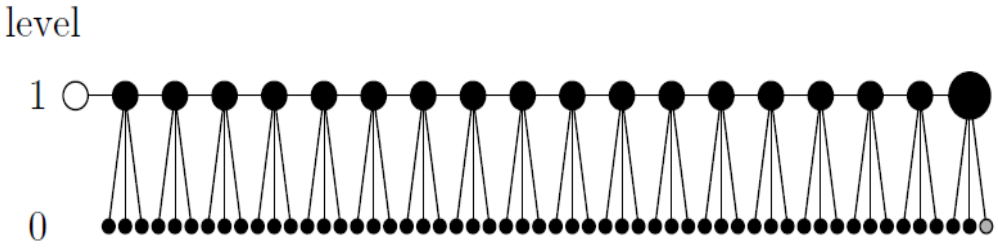
- <u>מצב פעולה טורי:</u>
במצב פעולה טורי, האלגוריתם כמו במצב הפעולה המקבילי מחלק את ההודעה לבלוקים של 16 מילים, אבל לעומת מצב הפעולה המקבילי, מצב הפעולה הטורי עובד עם שתי רמות בלבד.

כאשר ברמה 0 מופיעים כל בלוקי המידע המחולקים ל-16 מילים, וברמה 1 מופיעים כל פונקציות הדחיסה הנצרכות לצורך הגיבוב ובקצה הכי שמאלי וקטור אתחול 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 $L = 0$.



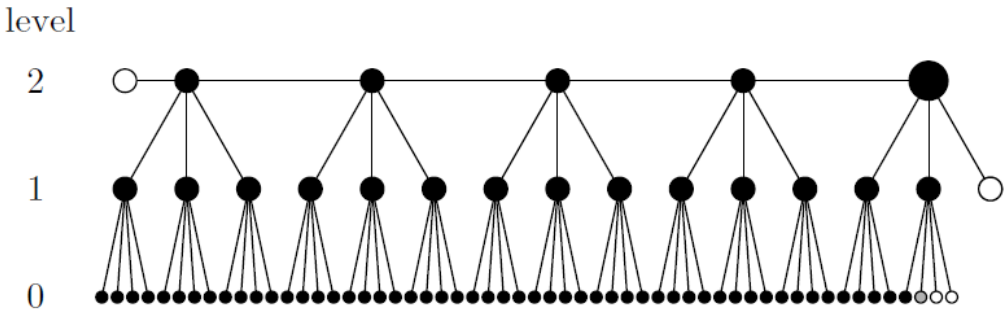
איור 2.3 – מצב הפעולה הטורי [4].

- <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 [4].

2.3.4 פונקציית הדחיסה

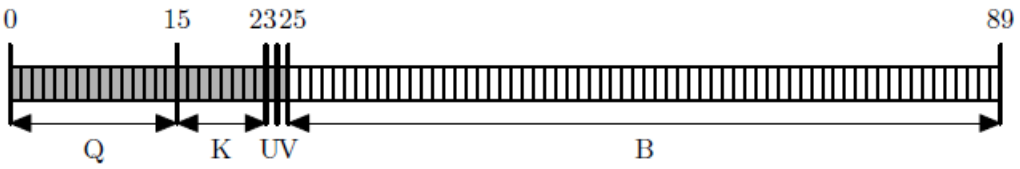
פונקציית הדחיסה, היא פונקציית החישוב המרכזית של אלגוריתם MD6, כל פונקציית דחיסה מקבלת 64 מילים של 64 סיביות שהם 4096 סיביות של הודעה, ומוציאה גיבוב של 16 מילים של 64 סיביות שהם 1024 סיביות מידע. בפרק זה נפרט על מבנה הפונקציה ואופן פעולתה.

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 [4].

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 [4].

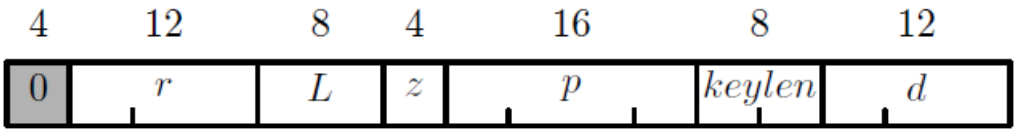
2.3.4.1.2 מילת הבקרה V

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 [4].

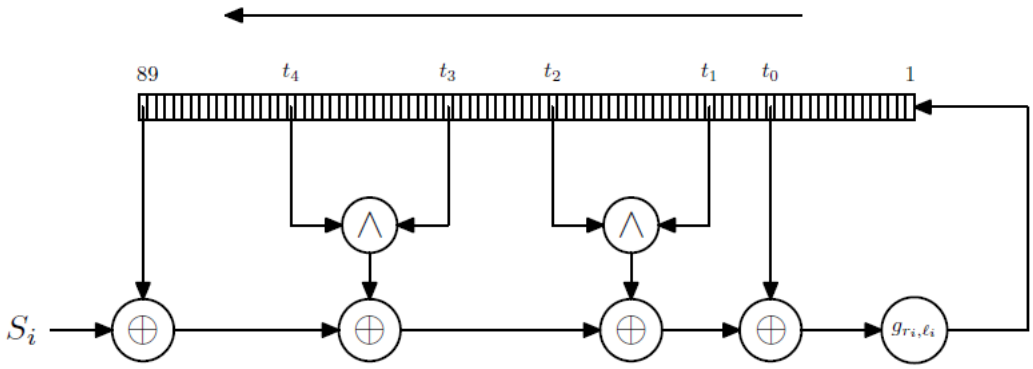
2.3.4.2 פעולת החישוב של פונקציית הדחיסה

כמו שהוסבר בפרק הקודם, פונקציית הדחיסה מעבדת את בלוק הנתונים N, ומפעילה את לולאת החישוב המוצגת בטבלה 5.

טבלה 5: לולאת החישוב של פונקציית הדחיסה [4].

$$n = 89, t = 16$$
$$\text{for j in range}(0, r)$$
$$\text{for i in range}(n + j \cdot t, n + (j+1) \cdot t)$$
$$\{$$
$$x = S_j \oplus A_{i-n} \oplus A_{i-t_0}$$
$$x = x \oplus (A_{i-t_1} \,\&\, A_{i-t_2}) \oplus (A_{i-t_3} \,\&\, A_{i-t_4})$$
$$x = x \oplus (x \gg r_{(i-n)\%16})$$
$$A_i = x \oplus (x \ll l_{(i-n)\%16})$$
$$\}$$

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 [4].

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.

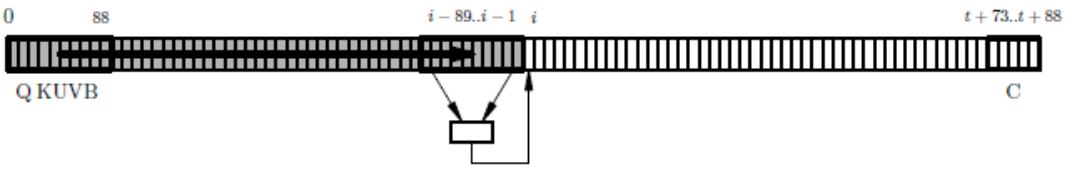
בשורה הרביעית עושים XOR עם התוצאה של השורה השלישית ועם אותה תוצאה אשר מוזזת שמאלה לפי וקטורי ההזזה l.

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.

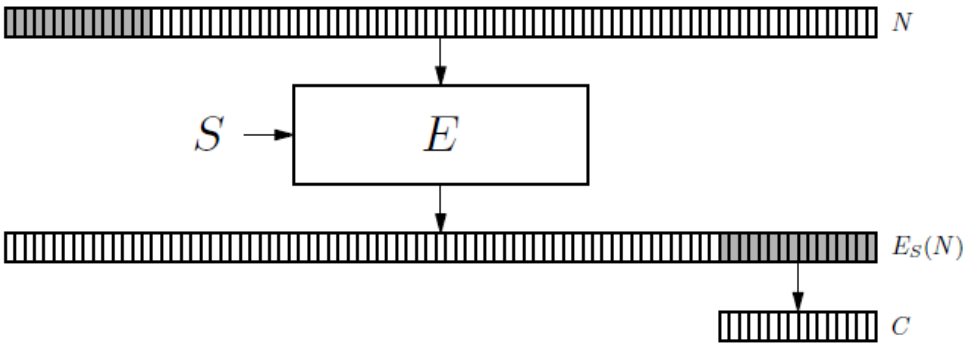


איור 2.9 – אופן הפעולה של לולאת החישוב [4].

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 [4].

3 פיתוח ושיטות

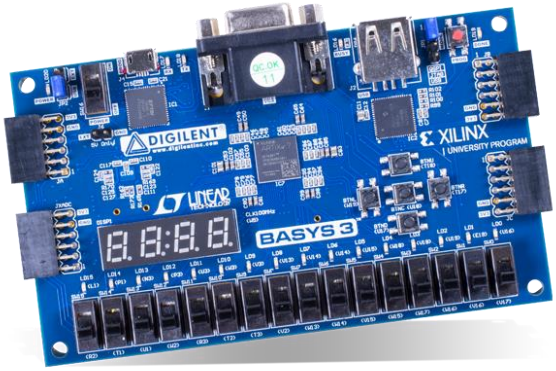
במסגרת הפרויקט, מומש אלגוריתם MD6 אשר מכיל פונקציה דחיסה אחת בלבד אשר פועלת לפי מצב הפעולה הסטנדרטי, עקב מחסור במשאבי רכיב החומרה כפי שיוסבר להלן.

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

פרק זה מחולק באופן הבא:

- ערכת הפיתוח ורכיב ה-FPGA
- מעטפת להעברת נתונים
- מימוש האלגוריתם בשפת חומרה

3.1 ערכת הפיתוח ורכיב ה-FPGA



איור 3.1 – ערכת הפיתוח Basys-3 [7].

במסגרת הפרויקט התכנון הוטמעה על רכיב FPGA מסוג Artix-7 של חברת Xilinx-AMD, הכלול בערכת הפיתוח Basys-3 של חברת Digilent [8]. כמופיע באיור 3.1.

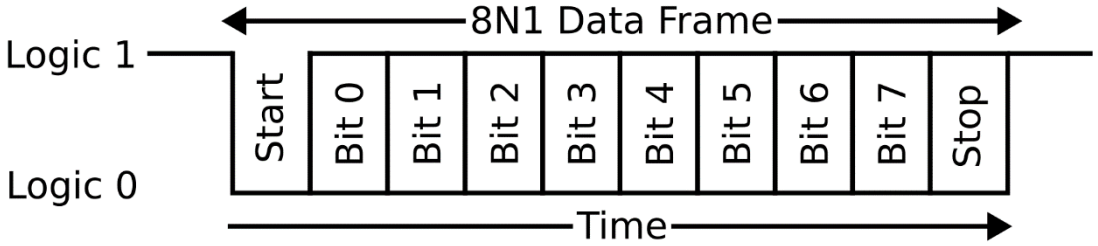
טבלה 6 מציגה את המאפיינים העיקריים של ערכת הפיתוח.

טבלה 6: 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
תצוגות	תצוגה אחת בת 4 ספרות בת 7 פלחים
מתגים ונוריות	• 16 מתגים • 16 נורות לד • 5 כפתורים
שעונים	מתנד בעל תדר של 100MHz
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].

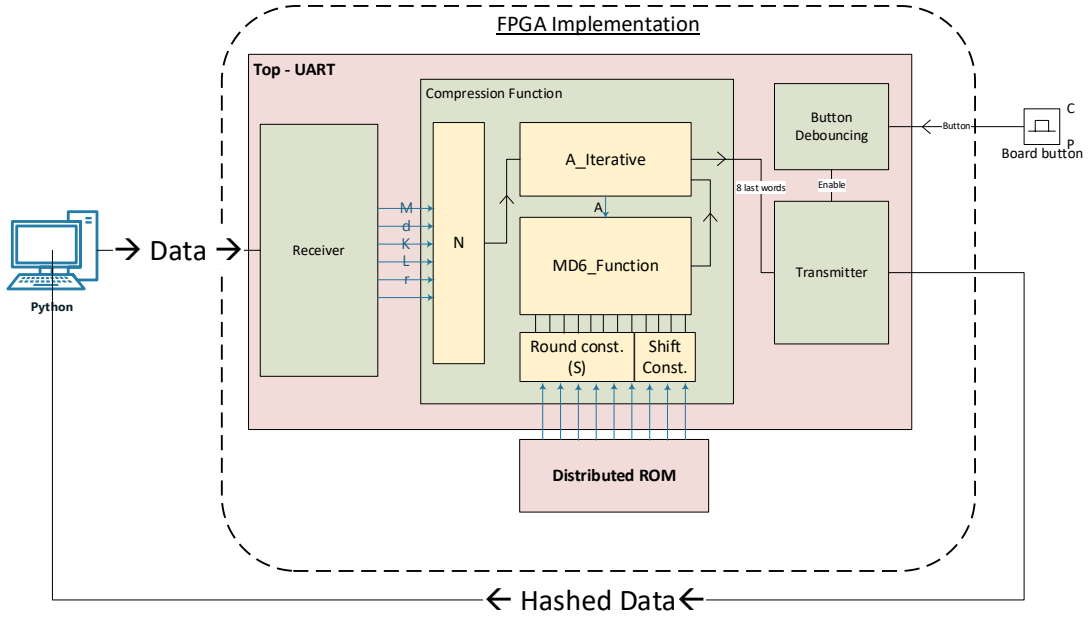


איור 3.2 – מבנה שליחת המידע בתקשורת UART מסוג 8N1 [10].

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 מוכן 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.

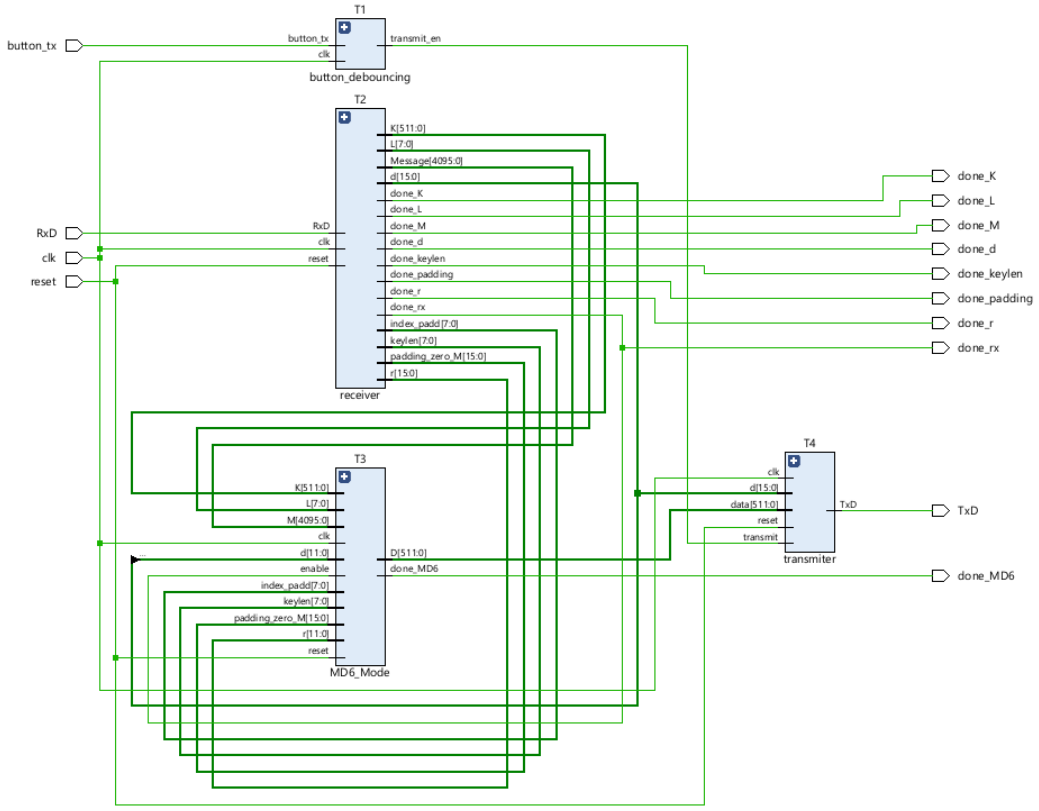
באיור 3.3 ניתן לראות דיאגרמת בלוקים של התכנון כולל מעטפת העברת המידע.



איור 3.3 – דיאגרמת בלוקים של התכנן כולל המעטפת.

3.3 מימוש האלגוריתם בשפת חומרה

בפרק זה נפרט על כתיבת קוד ה-RTL ב-Verilog שהטמענו על רכיב ה-FPGA כמוצג בדיאגרמת הבלוקים באיור 3.4.



איור 3.4 – דיאגרמת הבלוקים של המימוש.

פירוט ה-IO והאותות הפנימים של התכנון מופיע בטבלאות 7 ו-8.

טבלה 7: הקלט והפלט של התכנון.

שם האות	גודל	קלט/פלט	תיאור
Clk	1 סיביות	קלט	תדר השעון של הרכיב 100MHz
Reset	1 סיביות	קלט	אות האיפוס
Button tx	1 סיביות	קלט	אות לשליחת המידע המגובב חזרה
RxD	1 סיביות	קלט	סיבית קבלת המידע ע"י ה-UART
TxD	1 סיביות	פלט	סיבית החזרת המידע ע"י ה-UART
Done M	1 סיביות	פלט	נורת סימון לסיום קבלת ההודעה
Done d	1 סיביות	פלט	נורת סימון לסיום קבלת אורך הודעת הגיבוב
Done K	1 סיביות	פלט	נורת סימון לסיום קבלת המפתח
Done L	1 סיביות	פלט	נורת סימון לסיום קבלת פרמטר בקרת המצב
Done r	1 סיביות	פלט	נורת סימון לסיום קבלת מספר הסיבובים
Done keylen	1 סיביות	פלט	נורת סימון לסיום קבלת אורך המפתח
Done padding	1 סיביות	פלט	נורת סימון לסיום קבלת אורך הריפוד
Done MD6	1 סיביות	פלט	נורת סימון לסיום קבלת כל המידע

טבלה 8: האותות הפנימיים של התכנון.

שם האות	גודל	תיאור	מבלוק	לבלוק
Message	4096 סיביות	ההודעה	receiver	MD6 Mode
D	16 סיביות	אורך ההודעה המגובבת	receiver	MD6 Mode
K	512 סיבות	המפתח	receiver	MD6 Mode
L	8 סיביות	פרמטר בקרת המצב	receiver	MD6 Mode
R	16 סיביות	מספר הסיבובים	receiver	MD6 Mode
Keylen	8 סיביות	אורך המפתח בבתים	receiver	MD6 Mode
padding zero M	16 סיביות	אורך ריפוד ההודעה	receiver	MD6 Mode
index padd	8 סיביות	מספר פונקציות הדחיסה	receiver	MD6 Mode
Hash	512 סיביות	המידע המגובב	MD6 mode	transmitter
transmit en	1 סיביות	אות בקרה לשליחת המידע המגובב	Button debouncing block	transmitter

כפי שניתן לראות באיור 3.4, ההיררכיה הראשונה בתכנון מחולקת ל-4 בלוקים.

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

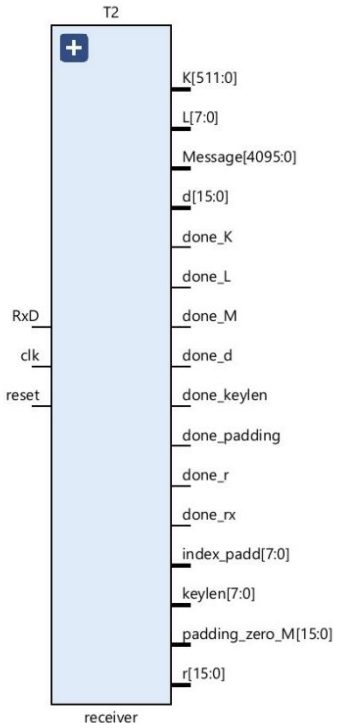
3.3.1 בלוק ה-Receiver

כפי שהוסבר לעיל העברת המידע לרכיב החומרה וקבלתו חזרה נעשתה ע"י תקשורת טורית – UART.

בלוק ה-receiver כמוצג באיור 3.5, לקבלת המידע לגיבוב, המידע מתקבל ב-bus אחד של סיביות המורכב מכמה סוגים של תתי מידע, כפי שמתואר באיור 3.6. בכך יש שליטה על סדר המידע המתקבל כך שכל חלק וחלק יגיע למקום הנכון. בשביל לתת למשתמש אינדיקציה ה-receiver מוציא אות ביציאה אשר מדליק נורה על ערכת הפיתוח ה-Basys3 בכל פעם שהוא מסיים לקבל חלק מהמידע.

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

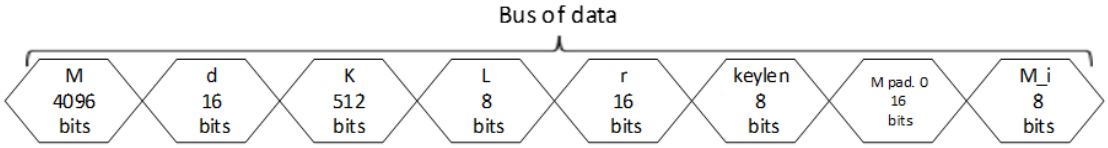
פירוט ה-IO והאותות הפנימים של ה-receiver מופיע בטבלאות 9 ו-10.



איור 3.5 – סימבול בלוק ה-receiver.

טבלה 9: הקלט והפלט של בלוק ה-receiver.

שם האות	גודל	קלט/פלט	תיאור
Clk	1 סיביות	קלט	תדר השעון של הרכיב 100MHz
Reset	1 סיביות	קלט	אות האיפוס
RxD	1 סיביות	קלט	קבלת המידע ע"י ה-UART
Message	4096 סיביות	פלט	ההודעה
D	16 סיביות	פלט	אורך ההודעה המגובבת
K	512 סיבות	פלט	המפתח
L	8 סיביות	פלט	פרמטר בקרת המצב
R	16 סיביות	פלט	מספר הסיבובים
Keylen	8 סיביות	פלט	אורך המפתח בבתים
padding zero M	16 סיביות	פלט	אורך ריפוד ההודעה
index M	8 סיביות	פלט	מספר פונקציות הדחיסה
done M	1 סיביות	פלט	נורת סימון לקבלת ההודעה בהצלחה
done d	1 סיביות	פלט	נורת סימון לקבלת אורך הודעת הגיבוב בהצלחה
done K	1 סיביות	פלט	נורת סימון לקבלת המפתח בהצלחה
done L	1 סיביות	פלט	נורת סימון לקבלת פרמטר בקרת המצב בהצלחה
done r	1 סיביות	פלט	נורת סימון לקבלת מספר הסיבובים בהצלחה
done keylen	1 סיביות	פלט	נורת סימון לקבלת אורך המפתח בהצלחה
done padding	1 סיביות	פלט	נורת סימון לקבלת אורך הריפוד בהצלחה
done rx	1 סיביות	פלט	נורת סימון לסיום קבלת כל המידע
done MD6	1 סיביות	פלט	נורת סימון לסיום גיבוב ההודעה



איור 3.6 – ה-bus של תתי המידע.

טבלה 10: האותות הפנימיים של בלוק ה-receiver.

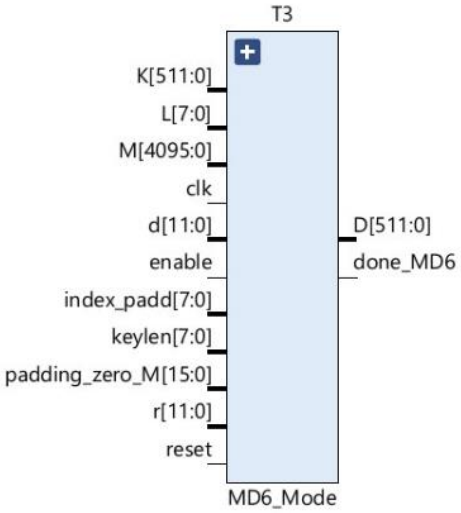
שם האות	גודל	תיאור
data reg M	4096 סיביות	אוגר לשמירת ההודעה
data reg d	16 סיביות	אוגר לשמירת אורך ההודעה המגובבת
data reg K	512 סיבות	אוגר לשמירת המפתח
data reg L	8 סיביות	אוגר לשמירת פרמטר בקרת המצב
data reg r	16 סיביות	אוגר לשמירת מספר הסיבובים
data reg keylen	8 סיביות	אוגר לשמירת אורך המפתח בבתים
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 ההודעה שהתקבלו
Index d	3 סיביות	אוגר למניית בתי אורך הודעת הגיבוב 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 פרמטר בקרת המצב שהתקבלו
Index r	3 סיביות	אוגר למניית בתי מספר הסיבובים שהתקבלו
Index keylen	2 סיביות	אוגר למניית בתי אורך המפתח בבתים 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 להזזת מידע
State	1 סיביות	אוגר בקרת המצבים של מכונת המצבים
Nextstate	1 סיביות	אות בקרה למעבר ממצב למצב במכונת המצבים
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 לספור עד 4 עבור oversampling
Counter	14 סיביות	מונה של 14 סיביות לספירת קצב ה-Baud rate
Rxshiftreg	10 סיביות	אוגר הזזת סיביות

3.3.2 בלוק ה-MD6 Mode

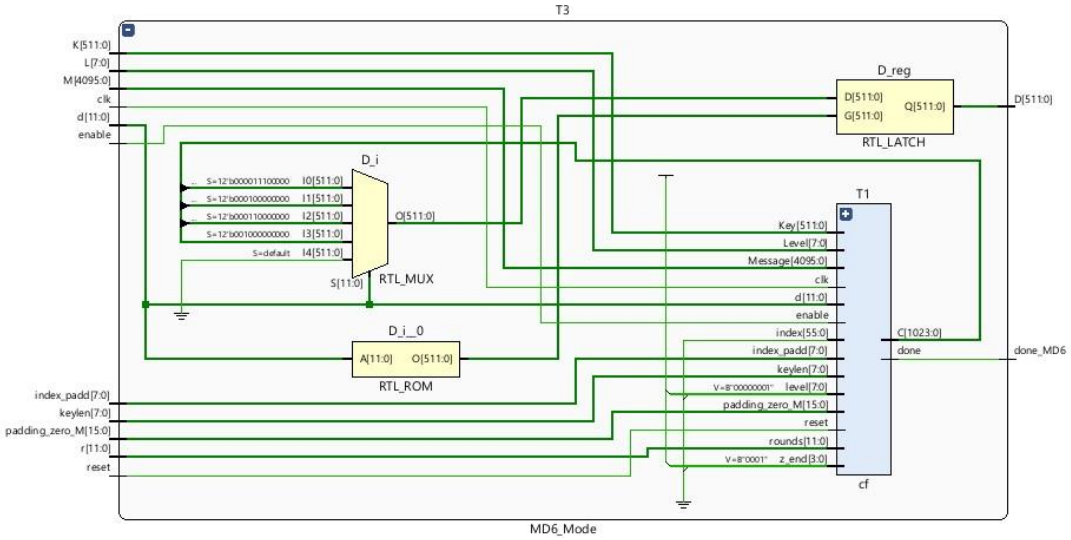
בלוק ה-MD6 Mode מקבל את המידע לאחר סידורו מבלוק ה-receiver ומוציא את הודעה המגובבת כמוצג באיור 3.7.

בלוק ה-MD6_Mode מכיל את פונקציה הדחיסה אשר מוציאה פלט של 16 מילים ומהם בלוק ה-MD6 mode בורר את ההודעה המגובבת לפי d (אורך ההודעה המגובבת שבחר המשתמש) הסיביות האחרונות ומרפד אותם באפסים (לפי הצורך) לגודל של 512 סיביות כמוצג באיור 3.8.

פירוט ה-IO והאותות הפנימים של ה-MD6 Mode מופיע בטבלאות 11 ו-12.



איור 3.7 – סימבול בלוק ה-MD6 Mode.



איור 3.8 – בלוק ה-MD6 Mode.

טבלה 11: הקלט והפלט של בלוק ה-MD6 Mode.

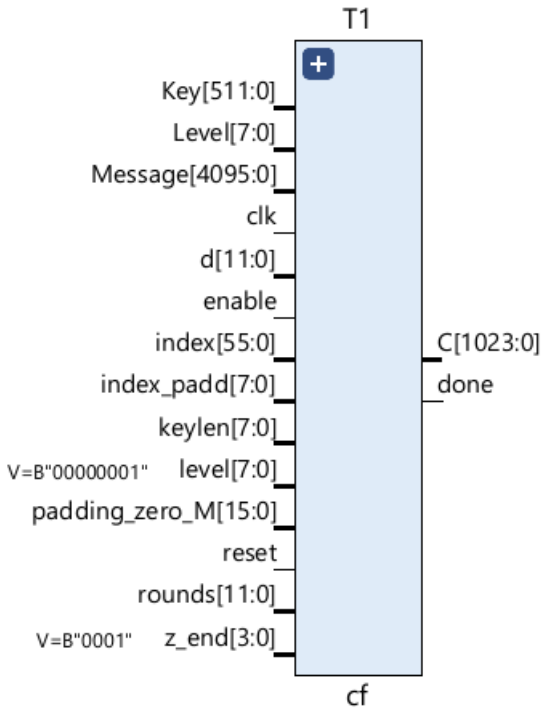
שם האות	גודל	קלט/פלט	תיאור
Clk	1 סיביות	קלט	תדר השעון של הרכיב 100MHz
Reset	1 סיביות	קלט	אות האיפוס
Enable	1 סיביות	קלט	אות בקרה להפעלת האלגוריתם
Message	4096 סיביות	קלט	ההודעה
D	16 סיביות	קלט	אורך ההודעה המגובבת
K	512 סיבות	קלט	המפתח
L	8 סיביות	קלט	פרמטר בקרת המצב
R	12 סיביות	קלט	מספר הסיבובים
Keylen	8 סיביות	קלט	אורך המפתח בבתים
padding zero M	16 סיביות	קלט	אורך ריפוד ההודעה
index padd	8 סיביות	קלט	מספר פונקציות הדחיסה
D	512 סיביות	פלט	ההודעה המגובבת
Done MD6	1 סיביות	פלט	נורת סימון לסיום גיבוב ההודעה

טבלה 12: האותות הפנימיים של בלוק ה-MD6 Mode.

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

3.3.2.1 בלוק ה-cf

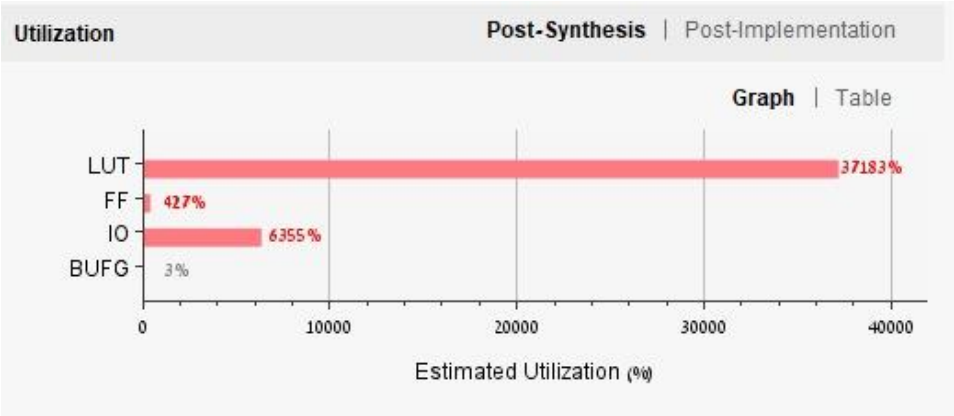
בלוק ה-cf (Compression function) כמוצג באיור 3.9 זו פונקציית הדחיסה המרכזית של אלגוריתם MD6 כמפורט בפרק 2.3.



איור 3.9 – סימבול בלוק ה-cf.

במסגרת הפרויקט תוכננה תחילה פונקציית הדחיסה בצורה מופשטת. הפונקציה מכילה אוגר שגודלו המקסימלי הוא 2,777 מילים של 64 סיביות (89 המילים הראשונות בבלוק הנתונים יחד עם מספר הסיבובים המקסימלי 168 כפול 16 מילים שנוצרות בכל סיבוב), והפעלנו לולאה אשר מחשבת 16 מילים בכל סיבוב ומכניסה אותם לבלוק הנתונים לפי הסדר, ובסוף היא מוציאה את 16 המילים האחרונות שנוצרו לאחר סיום לולאת החישוב.

החיסרון בתכנון זה הינו הדרישה לעודף משאבי החומרה הנצרכים למימוש התכנון (ביחס ל-spec של ה-Basys3, בסביבות 400% שימוש ב-FFs ו-37,000% שימוש ב-LUTs). עצם זה שקיים אוגר אשר מכיל גודל של 2,777 מילים, שבו עושים חישובים לוגיים ומשתמשים ב-FFs ו-LUTs לצורך שמירת המידע, גורם לחריגה כה גדולה מכמות המשאבים המוקצים כמתואר באיור 3.10 (אין להתייחס באיור לחלק של מספר היציאות והכניסות, כיוון שמדובר בניתוח בלוק ה-cf בלבד, ולא עם מעטפת ה-UART).

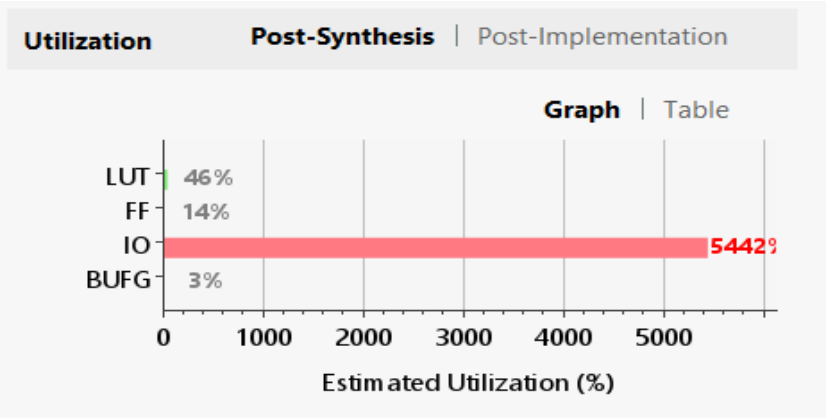


איור 3.10 – אחוז רכיבי החומרה המשומשים בתכנון הראשוני של בלוק ה-cf.

בשלב הראשון, מוזערו חישובים לא-אלגוריתמיים, כגון חישוב ריפוד אפס, סיבוב סדר הסיביות של הוקטור (כדי לקבל סדר Big-Endian) או חישוב המוצא של ערך ה-bus. לחילופין, נעשה שימוש בנתונים שהתקבלו מהקלט של פרוטוקול התקשורת UART

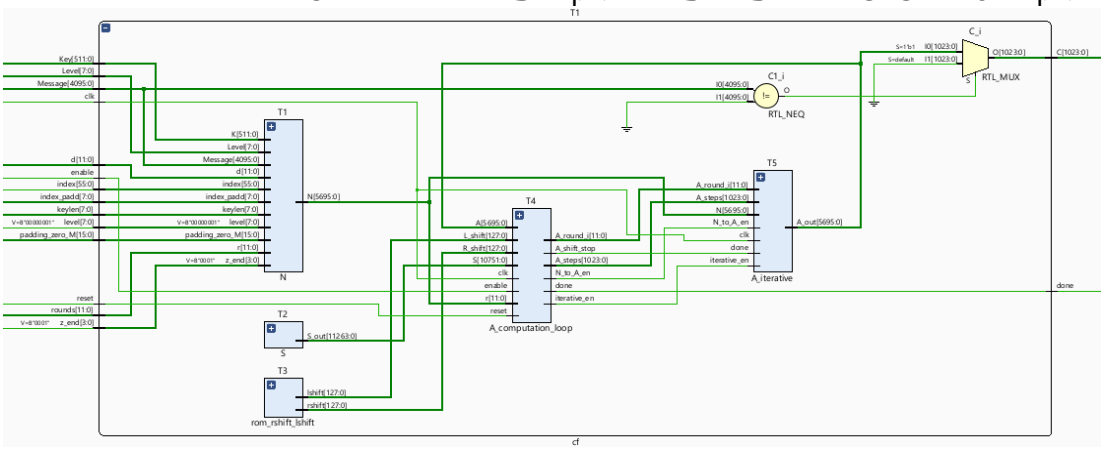
ומונף כוח העיבוד של המחשב בו פועל התכנון לביצוע חישובים אלו בתוכנה. בנוסף, נעשה שימוש ב-rom המבוזר המובנה של ה-FPGA כדי למנוע עומס יתר של משאבי ה-LUT.

בשלב השני, לולאת החישוב הראשית יושמה באמצעות מודל combinational logic. המשמעות היא שיש פחות שמריה של ערכי ביניים ובוצעו חישובים בכמה שפחות FFs/LUTs. למעשה נמצאה יתירות באלגוריתם בכך שנשמרו וקטורים גם שכבר לא היו בתוקף. החישובים מבוססים רק על 89 המילים האחרונות של הוקטור A (בלוק הנתונים). לכן, במקום לאחסן את כל הערכים שאינם בשימוש ובמקום לשמור את A כוקטור גדול של כל המילים שחושבו, הושגה דרך ליישם גישה איטרטיבית לחיסכון בשטח, על ידי ביצוע פעולת Shift להכנסת 16 המילים החדשות לתוך וקטור באורך קבוע של 89 מילים. כך הושג חסכון מאוד גדול ברכיבי חומרה והצלחנו לעמוד במספר המשאבים המוגבל כמוצג באיור 3.11.



איור 3.11 – אחוז רכיבי החומרה המשומשים בתכנון הסופי של בלוק ה-cf.

בלוק ה-cf האיטרטיבי מורכב מכמה בלוקים כמוצג באיור 3.12.



איור 3.12 – בלוק ה-cf.

פירוט ה-IO והאותות הפנימיים של ה-cf מופיע בטבלאות 13 ו-14.

טבלה 13: הקלט והפלט של בלוק ה-cf.

שם האות	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
Message	4096 סיביות	קלט	ההודעה
D	16 סיביות	קלט	אורך ההודעה המגובבת
Key	512 סיבות	קלט	המפתח
Level	8 סיביות	קלט	פרמטר בקרת המצב
Rounds	12 סיביות	קלט	מספר הסיבובים
Keylen	8 סיביות	קלט	אורך המפתח בבתים
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	שווה 1 אם זאת פונקציית הדחיסה האחרונה
C	1024 סיביות	פלט	16 המילים האחרונות שנוצרו
Done	1 סיביות	פלט	נורת סימון לסיום פונקציית הדחיסה

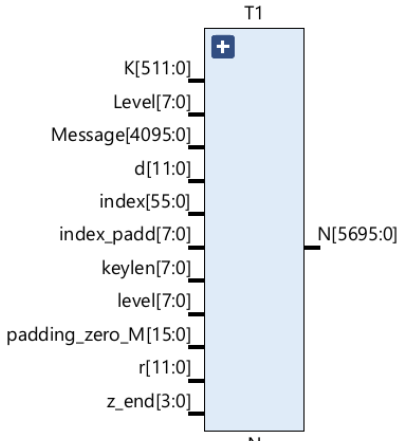
טבלה 14: האותות הפנימיים של בלוק ה-cf.

שם האות	גודל	תיאור	מבלוק	לבלוק
N	5696 סיביות	בלוק הנתונים N	N	A iterative
A steps	1024 סיביות	16 המילים הנוצרות בפונקציית הדחיסה	A computation loop	A iterative
A round i	12 סיבות	מספר הסיבוב הנוכחי	A computation loop	A iterative
N to A en	1 סיביות	אות בקרה להכנסת בלוק הנתונים הראשוני	A computation loop	A iterative
Iterative en	1 סיביות	אות בקרה לתחילת פעולת פונקציית הדחיסה	A computation loop	A iterative
A shift stop	1 סיביות	אות בקרה לסיום פעולת פונקציית הדחיסה	A computation loop	A iterative
S	10752 סיביות	וקטור הקבועים S	S	A computation loop
Rshift	128 סיביות	קבועי ההזזה ימינה	rom rshift lshift	A computation loop
Lshift	128 סיביות	קבועי ההזזה שמאלה	rom rshift lshift	A computation loop
R	12 סיביות	מספר הסיבובים	N	A computation loop
A	5696 סיביות	בלוק הנתונים הנכנס ללולאת החישוב	A iterative	A computation loop

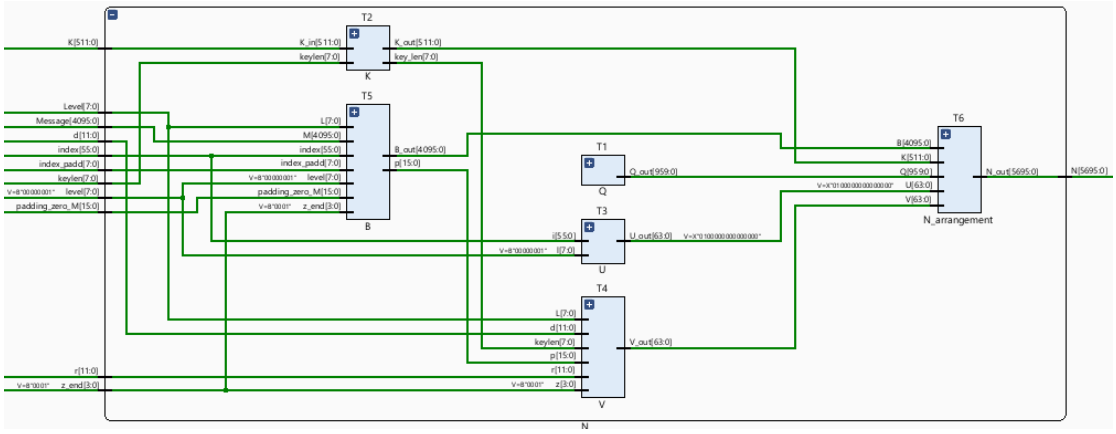
3.3.2.1.1 בלוק ה-N

תפקידו של בלוק ה-N המוצג באיור 3.13 לייצר מהמידע המתקבל מהמשתמש את בלוק הנתונים N המורכב מ-89 מילים אשר נכנס ללולאת החישוב של פונקציית הדחיסה, בלוק הנתונים N מורכב מכמה בלוקים כמוצג באיור 3.14, על אופן יצירת בלוק הנתונים N ועל חלקיו מפורט בפרק 2.3.4.

פירוט ה-IO והאותות הפנימיים של ה-N מופיע בטבלאות 15 ו-16.



איור 3.13 – סימבול בלוק ה-N.



איור 3.14 – בלוק ה-N.

טבלה 15: הקלט והפלט של בלוק ה-N.

שם האות	גודל	קלט/פלט	תיאור
Message	4096 סיביות	קלט	ההודעה
d	16 סיביות	קלט	אורך ההודעה המגובבת
Key	512 סיבות	קלט	המפתח
Level	8 סיביות	קלט	פרמטר בקרת המצב
rounds	12 סיביות	קלט	מספר הסיבובים
keylen	8 סיביות	קלט	אורך המפתח בבתים
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	שווה 1 אם זאת פונקציית הדחיסה האחרונה
N	5696 סיביות	פלט	בלוק הנתונים הראשוני - N

טבלה 16 האותות הפנימיים של בלוק N.

שם האות	גודל	תיאור	מבלוק	לבלוק
Q to N	960 סיביות	וקטור הקבועים Q	Q	N arrangement
K to N	512 סיביות	המפתח	K	N arrangement
Keylen to V	8 סיבות	אורך המפתח בבתים	K	N arrangement
U to N	64 סיביות	מזהה הצומת הייחודי	U	N arrangement
P to V	16 סיביות	אורך ריפוד ההודעה	B	V
V to N	64 סיביות	מילת הבקרה	V	N arrangement
B to N	4096 סיביות	ההודעה	B	N arrangement

3.3.2.1.2 בלוק ה-S

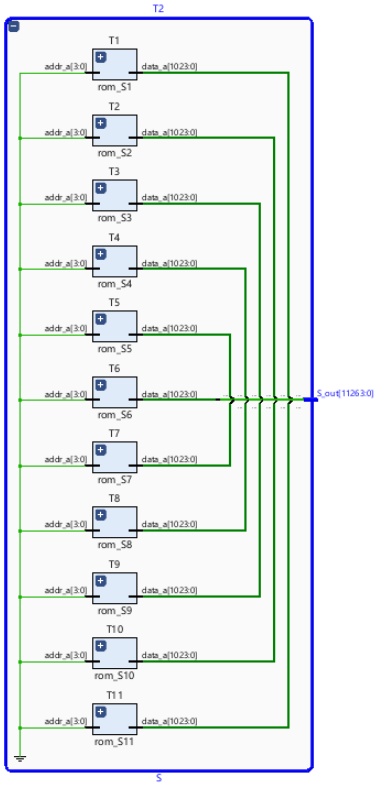
בלוק ה-S הוא בלוק המכיל את וקטור הקבועים S, אשר משמשים כחלק מחישוב פונקציית הדחיסה כמוסבר בפרק 2.3.2.

בלוק ה-S מורכב מ-11 בלוקי rom אסינכרונים אשר כל בלוק מכיל 1024 סיביות מוקטור הקבועים כמוצג באיור 3.15. (המידע מוכל בקבצי mem)

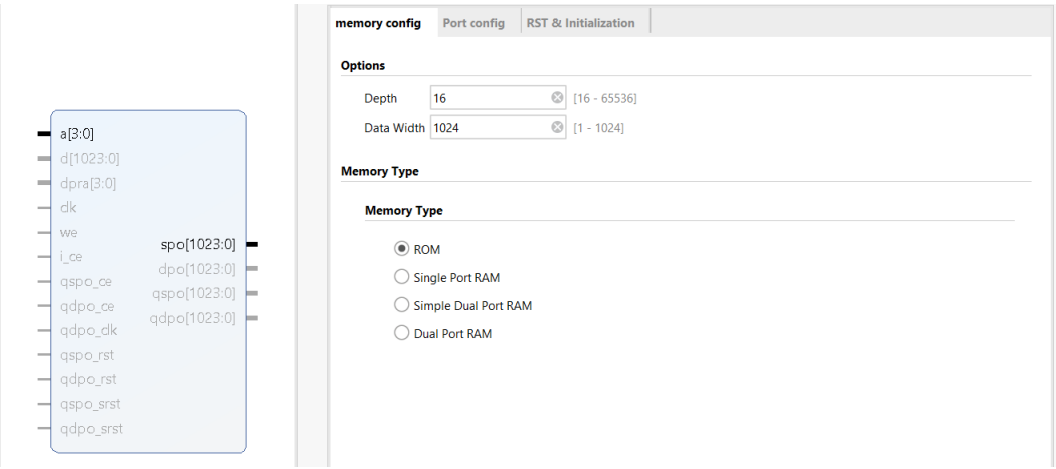
בלוקי ה-rom האסינכרונים ברכיב ה-FPGA בו מומש התכנון, מסוגלים להכיל 65,536 כתובת אשר בכל כתובת נכנסים 1024 סיביות כמוצג באיור 3.16.

כדי להעביר את הקבועים בפעימה אחת, נבחרה כתובת קבועה בכניסה לבלוקי ה-rom, ובכל בלוק, המידע אוחסן רק בכתובת הזו.

גודל וקטור ה-S הוא $168 \cdot 64 = 10752$, ובכל בלוק נכנס מידע בגודל 1024 סיביות, לכן מספר בלוקי ה-rom שהוצרכו זה $\frac{10752}{1024} = 10.5 \rightarrow 11$ (הבלוק האחרון של ה-rom מכיל בחציו הראשון את 512 האחרונים של וקטור הקבועים S וחציו השני מכיל אפסים).



איור 3.15 – סימבול בלוק ה-S.



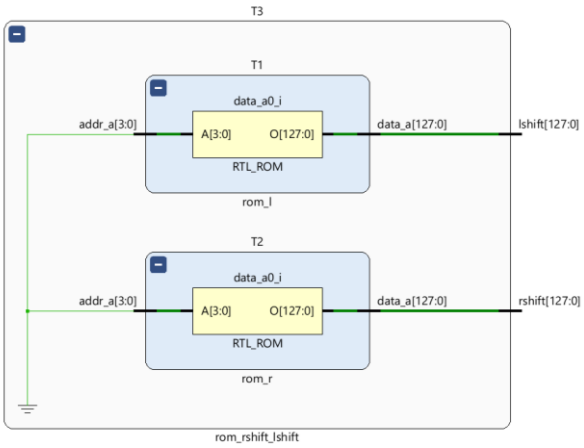
איור 3.16 - גודל בלוקי ה-rom האסינכרוני ברכיב ה-FPGA.

3.3.2.1.3 בלוק ה-rom rshift lshift

בלוק ה-rom rshift lshift מכיל את קבועי ההזזה rshift ו-lshift אשר משמשים כחלק מחישוב פונקציית הדחיסה כמוסבר בפרק 2.3.2.

כמו בלוק S גם הבלוק rom rshift lshift מורכב מבלוקי rom אסינכרונים כמוצג באיור 3.17.

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

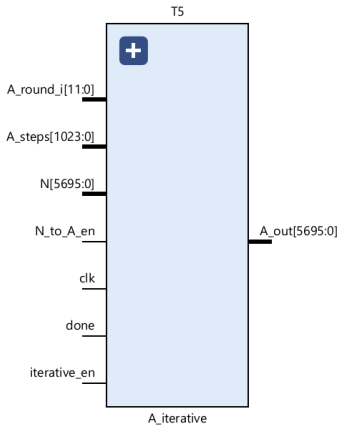


איור 3.17 – סימבול בלוק ה-rom rshift lshift.

3.3.2.1.4 בלוק ה-A iterative

בלוק A iterative תפקידו לייצור את האיטרטיביות של התכנון, כאשר הוא מקבל את הבלוק הנתונים ההתחלתי N ואת 16 המילים החדשות שנוצרו בבלוק החישוב כמוצג באיור 3.18.
בסיבוב הראשון הוא מוציא לבלוק החישוב את בלוק הנתונים N, ולאחר מכן בכל סיבוב הוא מוחק את 16 המילים הראשונות של הבלוק ומוסיף את 16 המילים החדשות שהתקבלו בסוף הבלוק, כך שבכל סיבוב יוצא שהוא אוגר 89 מילים.

פירוט ה-IO של ה-A iterative מופיע בטבלה 17 (אין אותות פנימיים).



איור 3.18 – סימבול בלוק ה- rom rshift lshift.

טבלה 17: הקלט והפלט של בלוק ה-A iterative.

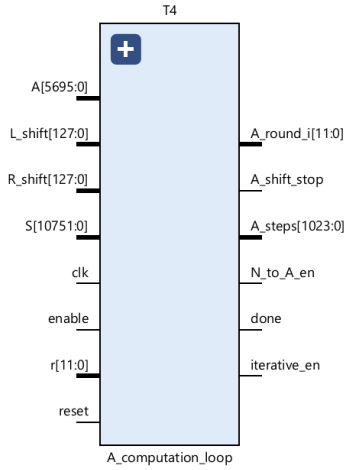
שם האות	גודל	קלט/פלט	תיאור
Clk	1 סיביות	קלט	תדר השעון של הרכיב 100MHz
N to A en	1 סיביות	קלט	אות בקרה להכנסת בלוק הנתונים הראשוני
Iterative en	1 סיבות	קלט	אות בקרה לתחילת פעולת פונקציית הדחיסה
Done	1 סיביות	קלט	אות בקרה לתחילת האיטרטיביות
A round i	12 סיביות	קלט	מספר הסיבוב הנוכחי
N	5696 סיביות	קלט	בלוק הנתונים N
A step	1024 סיביות	קלט	16 המילים הנוצרות בפונקציית הדחיסה
A out	5696 סיביות	קלט	בלוק הנתונים החדש

3.3.2.1.5 בלוק ה- A computation loop

בלוק ה-A computation loop מכיל את האלגוריתם של לולאת פונקציית הדחיסה כך שבכל סיבוב הבלוק מקבל בלוק נתונים של 89 מילים ומוציא 16 מילים חדשות כמוסבר בטבלה 5, כמוצג באיור 3.19.

הבלוק בכל סיבוב מחשב 16 מילים חדשות ומעביר אותם לבלוק ה-A iterative שמוסיף אותם לבלוק הנתונים ומוריד את 16 המילים הראשונות בבלוק הנתונים, וכך בלוק ה- A iterative מחזיר בלוק נתונים חדש לבלוק A computation loop וחוזר חלילה עד סיום מספר הסיבובים המתקבל מהמשתמש.

פירוט ה-IO והאותות הפנימיים של ה- A computation loop מופיע בטבלאות 18 ו-19.



איור 3.19 – סימבול בלוק ה-A computation loop.

טבלה 18: הקלט והפלט של בלוק ה-A computation loop.

שם האות	גודל	קלט/פלט	תיאור
Clk	1 סיביות	קלט	תדר השעון של הרכיב 100MHz
Reset	1 סיביות	קלט	אות האיפוס
Enable	1 סיבות	קלט	אות בקרה להפעלת האלגוריתם
R	12 סיביות	קלט	מספר הסיבובים
S	10752 סיביות	קלט	וקטור הקבועים S
R shift	128 סיביות	קלט	קבועי ההזזה ימינה
L shift	128 סיביות	קלט	קבועי ההזזה ימינה
A	5696 סיביות	קלט	בלוק הנתונים בכניסה
N to A en	1 סיביות	פלט	אות בקרה להכנסת בלוק הנתונים הראשוני
Iterative en	1 סיביות	פלט	אות בקרה לתחילת פעולת פונקציית הדחיסה
Done	1 סיביות	פלט	אות בקרה לתחילת 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 המילים החדשות שנוצרו

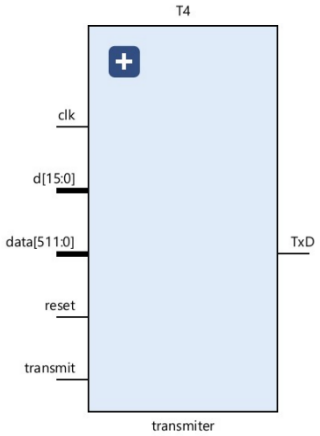
טבלה 19: באותות הפנימיים של בלוק ה- A computation loop.

שם האות	גודל	תיאור
J	12 סיביות	מונה למניית הסיבובים
State	3 סיביות	מצבי המכונת מצבים

3.3.3 בלוק ה-Transmitter

לאחר סיום גיבוב המידע יש לשלוח את המידע המגובב חזרה אל המשתמש, בלוק ה-transmitter כמופיע באיור 3.20 מקבל את המידע המגובב ואת אורכו ושולח אותו byte אחר byte חזרה למשתמש באמצעות תקשורת ה-UART, שליחת המידע נעשית לאחר קבלת אות מהמשתמש ע"י לחיצה על לחצן שליחת המידע בערכת הפיתוח.

פירוט ה-IO והאותות הפנימיים של ה-transmitter מופיע בטבלאות 20 ו-21.



איור 3.20 – סימבול בלוק ה-transmitter.

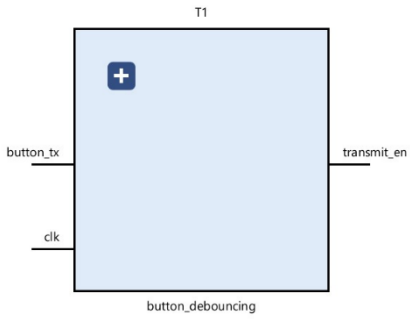
טבלה 20: הקלט והפלט של בלוק ה-transmitter.

שם האות	גודל	קלט/פלט	תיאור
Clk	1 סיביות	קלט	תדר השעון של הרכיב 100MHz
reset	1 סיביות	קלט	אות האיפוס
transmit	1 סיבות	קלט	אות בקרה לשליחת 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 21: האותות הפנימיים של בלוק ה-transmitter.

שם האות	גודל	תיאור
state	1 סיביות	אוגר בקרת המצבים של מכונת המצבים
nextstate	1 סיביות	אות בקרה למעבר ממצב למצב במכונת המצבים
shift	1 סיביות	אות הפעלה להזזת מידע
load	1 סיביות	אות בקרה כדי להתחיל לטעון את הנתונים לתוך אוגר ההזזה ולהוסיף סיביות התחלה ועצירה
clear	1 סיביות	אות בקרה לאיפוס ה-bitcounter
index	7 סיביות	אוגר עזר למיון הסיביות לשליחה
Index reg	7 סיביות	אוגר עזר למיון הבתים לשליחה
rightshiftreg	640 סיביות	אוגר שמירת המידע המרופד בסיביות התחלה ועצירה
bitcounter	11 סיביות	מונה למניית הסיביות בשליחת byte המרופד
counter	14 סיביות	מונה למניית קצב ה-baud rate

3.3.4 בלוק ה-Button Debouncing

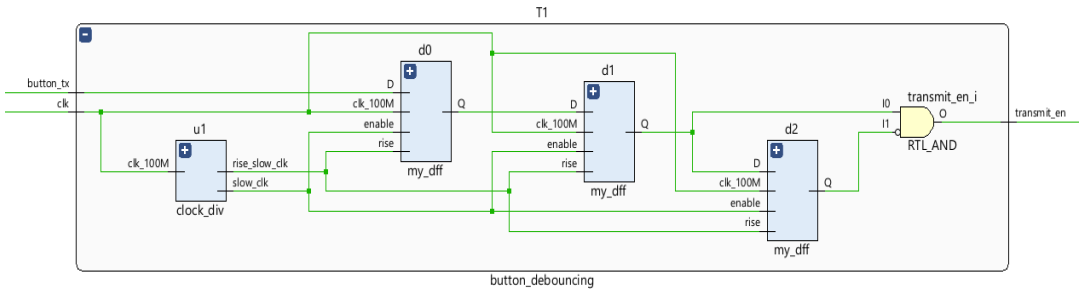


איור 3.21 – סימבול בלוק ה- Button debouncing.

קבלת המידע המגובב חזרה נעשית ע"י לחיצת כפתור בערכת הפיתוח ה-Basys3, כאשר לוחצים על כפתור הוא עלול לקפוץ פיזית עקב המגעים המכניים שבתוכו יוצרים חיבורים וניתוקים רגעיים ומהירים ואלו יכולים לגרום ליצירת אותות שווא מרובים במקום אות אחד יחיד.

לפתרון הבעיה תוכנן בלוק ה- Button debouncing כמופיע באיור 3.21.

כדי לבטל את אותות השווא, נוצר שעון בעל תדירות נמוכה ביחס לשעון הראשי, כך שיתאפשר לדגום את האות ולדלג על אותות השווא. לפי שעון זה, האות מועבר בשלושה FFs אשר לוכדים את האות בשלושה מחזורי שעון שונים כמופיע באיור 3.22, וכך כל אותות השווא שנוצרו מסוננים.



איור 3.22 – בלוק ה-Button debouncing .

פירוט ה-IO והאותות הפנימיים של ה-button debouncing מופיע בטבלאות 22 ו-23.

טבלה 22: הקלט והפלט של בלוק 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 סיבות	קלט	אות שליחת המידע אחרי סינון רעשים

טבלה 23: האותות הפנימיים של בלוק ה-Button debouncing.

שם האות	גודל	תיאור
Rise slow clk	1 סיביות	דלג לעליית שעון בשעון איטי
Slow clk	1 סיביות	השעון האיטי
Q1	1 סיבות	אות יציאה מה-FF הראשון
Q2	1 סיביות	אות יציאה מה-FF השני
Q2_bar	1 סיביות	היפוך ה-Q2
QO	1 סיביות	אות היציאה מה-FF השלישי

4 תוצאות

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

4.1 אימות תכנון החומרה בסימולציית ModelSim

4.1.1 תהליך אימות תכנון החומרה בסימולציית ModelSim

במסגרת הפרויקט בוצע אימות תכנון החומרה בכלי הסימולציה ModelSim שמדמה את פעילות התכנון על רכיב החומרה. בשביל ליצור הדמיה של הכניסות והיציאות, קובץ test bench עוטף את קוד החומרה ובו נכנסים הערכים הרצויים בקלטי התכנון.

הסימולציה כוללת גיבוב מידע אקראי שנכנס לאלגוריתם. 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 במימוש זה כדי לאמת את האלגוריתם, את תוצאת הגיבוב בתוכנה הוא מעביר לקובץ טקסט בשם soft hash.txt.

- tb top sim.v:
  קוד ה-test bench לוקח את מידע הקלט האקראי שנוצר בקובץ input data.txt ושומר אותו במערך של bytes. לאחר מכן ה-tb מפעיל לולאה שמעבירה כל byte ממידע הקלט לתכנון כנצרך, כמופיע באיור 4.1.

```
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 4.1 – שליחת המידע האקראי דרך ה-RxD.

לאחר סיום הגיבוב, ה-tb מקבל בחזרה את המידע המגובב ושומר אותו במערך מתאים, כמופיע באיור 4.2.

```
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 4.2 – קבלת המידע המגובב דרך ה-TxD.

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

4.1.2 הפעלת אימות תכנון החומרה בסימולציית ModelSim

פרק זה מפרט את תהליך הפעלת אימות תכנון החומרה בסימולציית ModelSim של אלגוריתם MD6.

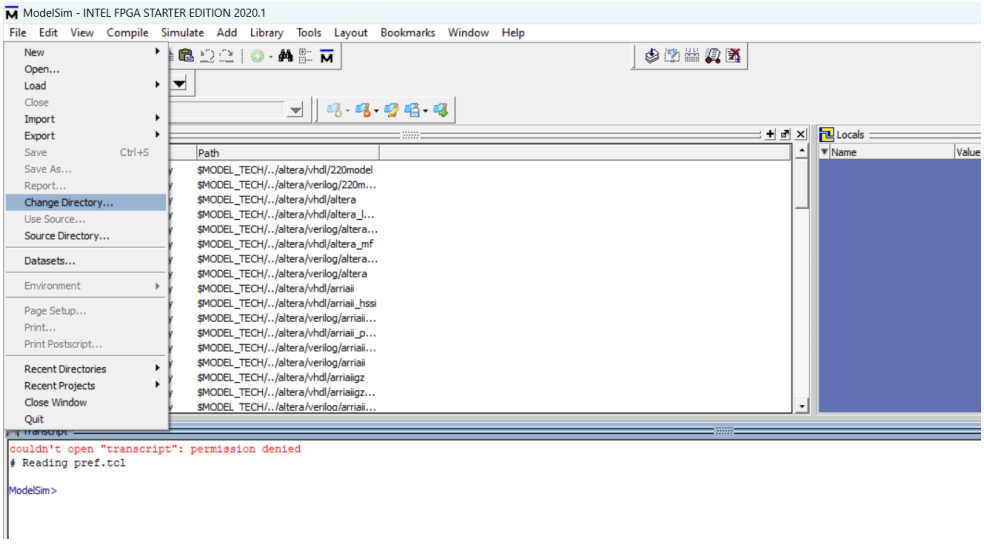
כדי שיהיה ניתן להפעיל את האלגוריתם, יש לוודא שהדברים הבאים נמצאים בהישג יד:

- תיקיית הקבצים של הפרויקט.
- כלי התוכנה ModelSim.
- שפת Python וסביבת עבודה מותאמת להפעלה.

קישור לתיקיית קבצי הפרויקט להפעלת האלגוריתם מצורפים בנספח א.

שלבי הפעלת סימולציית אימות תכנון החומרה בסימולציית ModelSim:

- תחילה יש להעביר את הקבצים הנצרכים לסימולציה לתיקייה אחת
  - קבצי ה-Source הנמצאים ב--md6_project main\MD6_CF_hardware\Code files.
  - קובץ ה-tb tb_top_sim.v הנמצא ב--md6_project main\MD6_CF_hardware\Test .Bench
  - קובץ ה-Python Test_Vector_Sim.py הנמצא ב- md6_project-main\ Executable files\Test_vector_sim.
- יש לפתוח את קובץ ה-Python Test_Vector_Sim.py, ולהפעיל אותו. הקובץ מייצר את 2 קבצי הטקסט בהם משתמש קובץ ה-test bench: מידע קלט אקראי לחומרה ותוצאת הגיבוב בתוכנה.
- יש להפעיל את כלי התוכנה ModelSim.
- יש ללחוץ על Change directory אשר נמצא תחת הכותרת file כמוצג באיור 4.3, ויש לשנות את מיקום התיקייה לתיקייה בה הועברו את הקבצים.



איור 4.3 – Change directory בכלי התוכנה ModelSim.

- ב-Transcript ניצור תיקיית עבודה ע"י שימוש בפקודה הבאה:

$$\boldsymbol{vlib}\ work$$

- לאחר יצירת התיקייה יש לקמפל את קבצי ה-Source ע"י שימוש בפקודה הבאה:

$$\boldsymbol{vlog}\ N.v\ MD6\_Mode.v\ cf.v\ ....$$

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

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

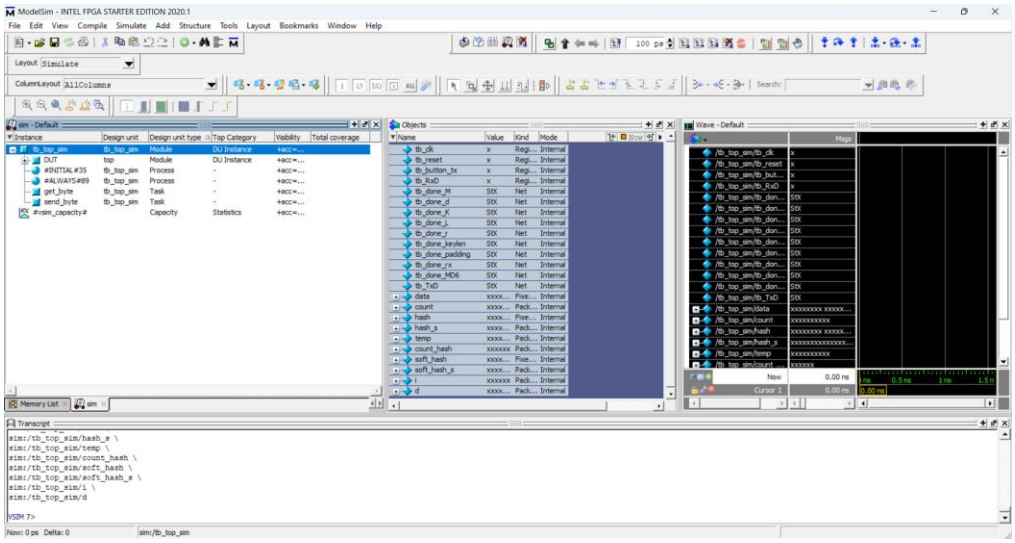
- לאחר שהקימפול של קבצי ה-Source עבר בהצלחה, יש לקמפל את קובץ ה-test bench:

$$\boldsymbol{vlog}\ tb\_top\_sim.v$$

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

$$\boldsymbol{vsim}\ tb\_top\_sim$$

- לאחר הפעלת פקודת ה-vsim יפתח החלון הראשי של הסימולציה כמוצג באיור 4.4.

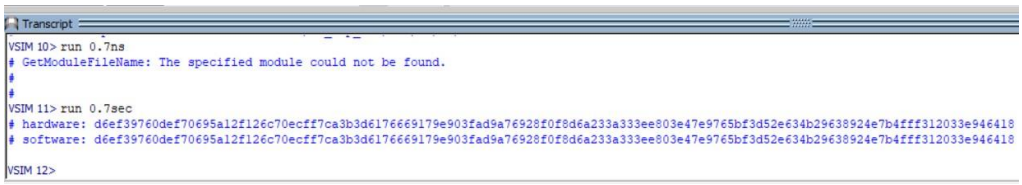


איור 4.4 – החלון הראשי של הסימולציה בכלי התוכנה ModelSim.

- כדי לראות את צורת הגלים של האותות, יש לבחור בחלון ה-Object את האותות הרצויים ולאחר מכן יש ללחוץ מקש ימני ולבחור Add Wave (או בקיצור Ctrl + w).
- להרצת הסימולציה יש לכתוב בשורת ה-Transcript את הפקודה run ואת זמן הריצה הרצוי.
- זמן הריצה הנדרש לסיום כל הסימולציה הינו לכל היותר 0.7sec. אי לכך יש להריץ את הפקודה הבאה:

$$\boldsymbol{run}\ 0.7sec$$

- אחרי סיום הרצת הסימולציה יתקבלו במסך ה-Transcript את תוצאות הגיבוב בחומרה ובתוכנה אחד אחרי השני כמוצג באיור 4.5.



איור 4.5 – תוצאות הגיבוב בחומרה ובתוכנה במסך ה-Transcript.

5 דיונים

בפרק זה נשווה בין תכנון אלגוריתם MD6 שתוכנן במסגרת הפרויקט, לבין התוצאות בדו"ח האלגוריתם של יוצר האלגוריתם Ronald Linn Rivest [4].

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 [4]. 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 [4].

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 [4].

כדי לסכם בצורה יותר ברורה מידע הקלט ותוצאת הגיבוב מוצגים בטבלה 24.

טבלה 24: 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 בסימולציית ModelSim

קובץ ה-test bench המשמש לבדיקת הדוגמא הראשונה הוא קובץ ה-tb_top.v אשר נמצא ב-md6_project-main\MD6_CF\Test Bench.

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

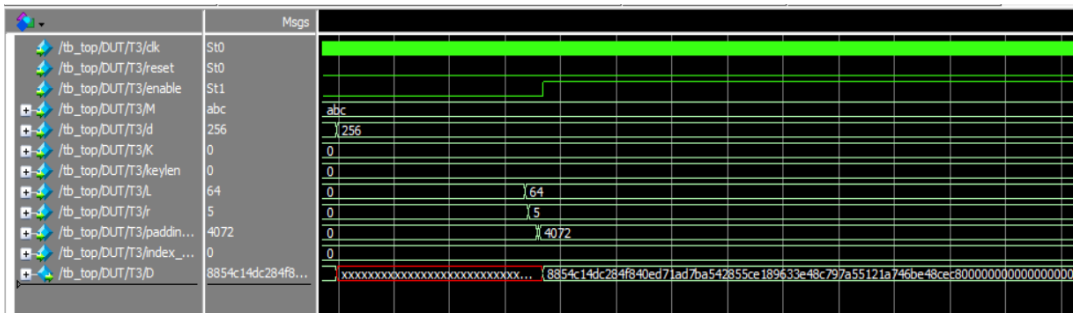
בנוסף, משום שהמידע מועבר באמצעות תקשורת טורית – UART, ה-test bench תוכנן כך שהמידע נכנס סיבית אחר סיבית כאשר כל byte עטוף בסיבית התחלה וסיבית עצירה, כמופיע באיור 5.3.

```
//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    = 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 [4].



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

באיור 5.4 מופיע חלון ה-wave של כלי תוכנת הסימולציה ModelSim אשר מציג את המידע שהתקבל ע"י תקשורת ה-UART ואת ההודעה המגובבת לפי הדוגמא הראשונה של האלגוריתם. ההודעה וההודעה המגובבת מוצגים בפורמט הקסדצימלי והשאר בפורמט דצימלי כדי להראות בצורה ברורה את נכונות המידע.

כפי שניתן לראות מהאיור, התקבלה תוצאת גיבוב נכונה המתאימה בדיוק לדו"ח האלגוריתם של Ronald Linn Rivest.

הפעלת סימולציה ב-ModelSim מפורטת צעד אחר צעד בפרק 4.1.2.

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 1:
(1)
$$\max \text{Freq} = \frac{1}{Crystal\ Oscillator - WNS}$$
כאשר:
Crystal Oscillator – זמן המחזור של מתנד הקריסטל של רכיב החומרה השווה ל-10ns, כמפורט בטבלה 6 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 בחלקי ההטמעה וקוד ההפעלה של הדו"ח המצורפים בנספח א.

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 2.
(2)
$$Throughput = \frac{num\ of\ bits}{num\ of\ rounds} \cdot max\ freq$$

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] (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)

[3] Cryptography Hash functions. (n.d.). Tutorialspoint.com. Retrieved December 16, 2023, from Cryptography Hash functions (tutorialspoint.com)

[4] 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)

[5] Dworkin, M. J. (2015). SHA-3 standard: Permutation-based hash and extendable-output functions. National Institute of Standards and Technology. Federal Information (nist.gov)

[6] Merkle, R. C. (n.d.). Ralphmerkle.com. Retrieved December 16, 2023, from Certified1979.pdf (ralphmerkle.com)

[7] (N.d.). Mouser.Co.Il. Retrieved December 11, 2023, from 102050644.png (600×436) (mouser.co.il)

[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] 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:
  Final_Project/Documents/Project reports/FPGA Implementation of MD6 Hash.pdf at main · aviel207/Final_Project (github.com)

- קישור לחלק קוד RTL וסימולציה של דו"ח הפרויקט:
  Final_Project/Documents/Project reports/FPGA Implementation of MD6 Hash - RTL Code and Simulation.pdf at main · aviel207/Final_Project (github.com)

- קישור לחלק הטמעה וקוד הפעלה של דו"ח הפרויקט:
  Final_Project/Documents/Project reports/FPGA Implementation of MD6 Hash - Implementation and Executable Code.pdf at main · aviel207/Final_Project (github.com)

- קישור לסרטון של הפעלת אלגוריתם MD6:
  Full Demonstration Of Implementing The CF MD6 On FPGA (youtube.com)

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 "Open Hardware Competition" של חברת AMD – Xilinx.

התחרות היא תחרות בין-לאומית בה מתחרים 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 המנצחים והפיינליסטים:

2023 Results Gallery - XOHW (openhw.eu)