

הפקולטה להנדסה ומדעי המחשב

החוג להנדסת חשמל ואלקטרוניקה

פרויקט גמר באלקטרוניקה

מימוש FPGA של אלגוריתם MD6 Hash

FPGA Implementation of the MD6 Hash Algorithm

|  |  |  |
| --- | --- | --- |
| מגישים: | אופק שרעבי, אביאל בירדואקר | ט' טבת תשפ"ד  ירושלים |
| מנחה: | מר אורי שטרו |

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.

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.

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.

תוכן עניינים

[רשימת איורים 7](#_Toc154137871)

[רשימת טבלאות 9](#_Toc154137872)

[1 מבוא 10](#_Toc154137873)

[2 רקע תיאורטי 11](#_Toc154137874)

[2.1 מבוא לקריפטוגרפיה [1] 11](#_Toc154137875)

[2.2 פונקציית גיבוב – HASH [2] 12](#_Toc154137876)

[2.3 אלגוריתם MD6 [3] 13](#_Toc154137877)

[2.3.1 הקלט והפלט של ה-MD6 13](#_Toc154137878)

[2.3.2 קבועים ב-MD6 14](#_Toc154137879)

[2.3.3 מצבי הפעולה של ה-MD6 17](#_Toc154137880)

[2.3.4 פונקציית הדחיסה 19](#_Toc154137881)

[3 פיתוח ושיטות 23](#_Toc154137882)

[3.1 ערכת הפיתוח ורכיב ה-FPGA [8] 23](#_Toc154137883)

[3.2 מעטפת להעברת נתונים 24](#_Toc154137884)

[3.3 מימוש האלגוריתם בשפת חומרה 25](#_Toc154137885)

[3.3.1 בלוק ה-Receiver 27](#_Toc154137886)

[3.3.2 בלוק ה-MD6 Mode 29](#_Toc154137887)

[3.3.3 בלוק ה-Transmitter 37](#_Toc154137888)

[3.3.4 בלוק ה- Button Debouncing 38](#_Toc154137889)

[3.4 הטמעה על רכיב ה-FPGA 39](#_Toc154137890)

[3.4.1 Constraint Specification – מפרט אילוצים 39](#_Toc154137891)

[3.4.2 Synthesis– סינתזה 41](#_Toc154137892)

[3.4.3 Place & Route – מיקום וחיוט 43](#_Toc154137893)

[3.4.4 Bitstream Generation and Device Programming 45](#_Toc154137894)

[3.5 יצירת קוד תוכנה לקישור בין החומרה למשתמש 45](#_Toc154137895)

[4 תוצאות 47](#_Toc154137896)

[4.1 הפעלת אלגוריתם MD6 47](#_Toc154137897)

[4.1.1 תהליך הטמעת אלגוריתם MD6 על רכיב ה-FPGA 47](#_Toc154137898)

[4.1.2 תהליך הפעלת גיבוב המידע על רכיב ה-FPGA 53](#_Toc154137899)

[4.2 אימות תכנון החומרה בסימולציית ModelSim 60](#_Toc154137900)

[4.2.1 תהליך אימות תכנון החומרה בסימולציית ModelSim 60](#_Toc154137901)

[4.2.2 הפעלת אימות תכנון החומרה בסימולציית ModelSim 61](#_Toc154137902)

[4.3 אימות תכנון החומרה על ערכת הפיתוח 63](#_Toc154137903)

[4.3.1 תהליך אימות תכנון החומרה על ערכת הפיתוח 63](#_Toc154137904)

[4.3.2 הפעלת אימות תכנון החומרה על ערכת הפיתוח 64](#_Toc154137905)

[5 דיונים 67](#_Toc154137906)

[5.1 השוואה לספרות 67](#_Toc154137907)

[5.1.1 השוואת תוצאות אימות תכנון החומרה בסימולציית ModelSim 68](#_Toc154137908)

[5.1.2 השוואת תוצאות אימות תכנון החומרה על ערכת הפיתוח 69](#_Toc154137909)

[5.2 השוואה לתוכנה 71](#_Toc154137910)

[6 מסקנות וסיכום 74](#_Toc154137911)

[מקורות מידע ומאמרים 75](#_Toc154137912)

[נספח א – קישורים למסמכי וקבצי הפרויקט 76](#_Toc154137913)

[נספח ב – פירוט תוכן תיקיית הפרויקט 77](#_Toc154137914)

[נספח ג – הגשה לתחרות 78](#_Toc154137915)

# רשימת איורים

[איור ‎2.1 – תקשורת בין אליס לבוב מותקפת על ידי איב [10]. 11](#_Toc154137763)

[איור ‎2.2 – מצב הפעולה המקבילי [3]. 17](#_Toc154137764)

[איור ‎2.3 – מצב הפעולה הטורי [3]. 18](#_Toc154137765)

[איור ‎2.4 – מצב הפעולה ההיברידי [3]. 18](#_Toc154137766)

[איור ‎2.5 – מצב הפעולה ההיברידי [3]. 19](#_Toc154137767)

[איור ‎2.6 – פריסת מזהה הצומת הייחודי U [3]. 19](#_Toc154137768)

[איור ‎2.7 – פריסת מילת הבקרה V [3]. 20](#_Toc154137769)

[איור ‎2.8 – לולאת החישוב מוצגת כאוגר הזזה לא לינארי [3]. 20](#_Toc154137770)

[איור ‎2.9 – אופן הפעולה של לולאת החישוב [3]. 21](#_Toc154137771)

[איור ‎2.10 – פונקציית הדחיסה f נראית כפעולת הצפנה ואחריה פעולת חיתוך [3]. 22](#_Toc154137772)

[איור ‎3.1 – ערכת הפיתוח Basys-3 [11]. 23](#_Toc154137773)

[איור ‎3.2 – מבנה שליחת המידע בתקשורת UART מסוג 8N1 [12]. 24](#_Toc154137774)

[איור ‎3.3 – דיאגרמת בלוקים של התכנן כולל המעטפת. 25](#_Toc154137775)

[איור ‎3.4 – דיאגרמת הבלוקים של המימוש. 25](#_Toc154137776)

[איור ‎3.5 – סימבול בלוק ה-receiver. 27](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137777)

[איור ‎3.6 – ה-bus של תתי המידע. 27](#_Toc154137778)

[איור ‎3.7 – סימבול בלוק ה-MD6 Mode. 29](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137779)

[איור ‎3.8 – בלוק ה-MD6 Mode. 29](#_Toc154137780)

[איור ‎3.9 – סימבול בלוק ה-cf. 30](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137781)

[איור ‎3.10 – אחוז רכיבי החומרה המשומשים בתכנון הראשוני של בלוק ה-cf. 30](#_Toc154137782)

[איור ‎3.11 – אחוז רכיבי החומרה המשומשים בתכנון הסופי של בלוק ה-cf. 31](#_Toc154137783)

[איור ‎3.12 – בלוק ה-cf. 31](#_Toc154137784)

[איור ‎3.13 – סימבול בלוק ה-N. מבחוץ. 33](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137785)

[איור ‎3.14 – בלוק ה-N. 33](#_Toc154137786)

[איור ‎3.15 – סימבול בלוק ה-S. 34](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137787)

[איור ‎3.16 - גודל בלוקי ה-rom האסינכרוני ברכיב ה-FPGA. 34](#_Toc154137788)

[איור ‎3.17 – סימבול בלוק ה-rom rshift lshift. 35](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137789)

[איור ‎3.18 – סימבול בלוק ה-rom rshift lshift. 35](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137790)

[איור ‎3.19 – סימבול בלוק ה-A computation loop. 36](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137791)

[איור ‎3.20 – סימבול בלוק ה-transmitter. 37](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137792)

[איור ‎3.21 – סימבול בלוק ה- Button debouncing. 38](file:///C:\Users\natib\Dropbox\דוחות%20משותפים\‏‏FPGA%20Implementation%20of%20MD6%20Hash.docx#_Toc154137793)

[איור ‎3.22 – בלוק ה- Button debouncing. 38](#_Toc154137794)

[איור ‎3.23 – חיבורי תקשורת ה-UART בערכת הפיתוח. 39](#_Toc154137795)

[איור ‎3.24 – חיבור הנוריות והלחצנים לפינים בערכת הפיתוח. 40](#_Toc154137796)

[איור ‎3.25 – מיקום הכפתורים והנוריות על ערכת הפיתוח. 40](#_Toc154137797)

[איור ‎3.26 – כלי החומרה המנוצילם לאחר סינתזה. 41](#_Toc154137798)

[איור ‎3.27 – מיקום רכיבי החומרה המשומשים לאלגוריתם. 43](#_Toc154137799)

[איור ‎3.28 – מפת הכניסות והיציאות של הרכיב. 44](#_Toc154137800)

[איור ‎3.29 – כלי החומרה המנוצלים לאחר מיקום וחיוט. 44](#_Toc154137801)

[איור ‎3.30 – ניתוח התזמון של התכנון. 44](#_Toc154137802)

[איור ‎4.1 – חלון הפתיחה של כלי התוכנה VIVADO. 47](#_Toc154137803)

[איור ‎4.2 – חלון ה-Create a New Vivado Project. 48](#_Toc154137804)

[איור ‎4.3 – חלון ה-Project Name. 48](#_Toc154137805)

[איור ‎4.4 – חלון ה-Project Type. 49](#_Toc154137806)

[איור ‎4.5 – חלון ה-Add Sources. 49](#_Toc154137807)

[איור ‎4.6 – חלון ה-Add Constraints. 50](#_Toc154137808)

[איור ‎4.7 – חלון ה-Default Part. 50](#_Toc154137809)

[איור ‎4.8 – חלון ה-New Project Summary. 51](#_Toc154137810)

[איור ‎4.9 – החלון הראשי של ניהול הפרויקט. 51](#_Toc154137811)

[איור ‎4.10 – חלון ה-HARDWARE MANAGER. 52](#_Toc154137812)

[איור ‎4.11 – חלון ה-HARDWARE MANAGER לאחר החיבור לרכיב החומרה. 52](#_Toc154137813)

[איור ‎4.12 – חלון ה-Program Device להטמעת התכנון. 53](#_Toc154137814)

[איור ‎4.13 – מציאת מספר ה-COM במנהל ההתקנים. 53](#_Toc154137815)

[איור ‎4.14 – חלון ההתחלה של תוכנית הגיבוב. 54](#_Toc154137816)

[איור ‎4.15 – חלון ה-INTRODUCTION (המבוא). 54](#_Toc154137817)

[איור ‎4.16 – חלון ה-INSTRUCTION (ההוראות). 55](#_Toc154137818)

[איור ‎4.17 – חלון ה-Definition questions (שאלות ההגדרה). 55](#_Toc154137819)

[איור ‎4.18 – חלון השגיאה על אי מילוי תאים. 56](#_Toc154137820)

[איור ‎4.19 – חלון ה-Message (ההודעה). 56](#_Toc154137821)

[איור ‎4.20 – חלון השגיאה על הכנסת מידע לא תואם. 56](#_Toc154137822)

[איור ‎4.21 – חלון ה-Key (המפתח) עם תיבת הכנסת המידע. 57](#_Toc154137823)

[איור ‎4.22 – חלון ה-Key (המפתח) עם ערך ברירת המחדל. 57](#_Toc154137824)

[איור ‎4.23 – חלון ה-d & L & r. 58](#_Toc154137825)

[איור ‎4.24 – חלון ה-COM number. 58](#_Toc154137826)

[איור ‎4.25 – חלון השגיאה על הכנסת מספר COM שגוי. 59](#_Toc154137827)

[איור ‎4.26 – חלוןthe hashed message (ההודעה המגובבת). 59](#_Toc154137828)

[איור ‎4.27 – שליחת המידע האקראי דרך ה-RxD. 60](#_Toc154137829)

[איור ‎4.28 – קבלת המידע המגובב דרך ה-TxD. 61](#_Toc154137830)

[איור ‎4.29 – Change directory בכלי התוכנה ModelSim. 62](#_Toc154137831)

[איור ‎4.30 – החלון הראשי של הסימולציה בכלי התוכנה ModelSim. 63](#_Toc154137832)

[איור ‎4.31 – תוצאות הגיבוב ביחד עם צורת הגלים של האותות הסופי. 63](#_Toc154137833)

[איור ‎4.32 – הכנסת מספר כניסת ה-COM. 64](#_Toc154137834)

[איור ‎4.33 – הכנסת מספר וקטורי הבדיקה. 65](#_Toc154137835)

[איור ‎4.34 – תצוגת הוקטור הראשון המגובב בתוכנה. 65](#_Toc154137836)

[איור ‎4.35 – תצוגת וקטור החומרה שהתקבל. 66](#_Toc154137837)

[איור ‎4.36 – קובץ csv להשוואה בין התוצאות. 66](#_Toc154137838)

[איור ‎5.1 – מידע הקלט של הדוגמא הראשונה [3]. 67](#_Toc154137839)

[איור ‎5.2 – תוצאת הגיבוב של הדוגמא הראשונה [3]. 67](#_Toc154137840)

[איור ‎5.3 – תוצאת הגיבוב של הדוגמא הראשונה [3]. 68](#_Toc154137841)

[איור ‎5.4 – חלון צורת הגלים של הקלט והפלט של הדוגמא הראשונה. 68](#_Toc154137842)

[איור ‎5.5 – התאמת שאלות ההגדרה לדוגמא הראשונה. 69](#_Toc154137843)

[איור ‎5.6 – התאמת ההודעה לדוגמא הראשונה. 69](#_Toc154137844)

[איור ‎5.7 – התאמת המפתח לדוגמא הראשונה. 70](#_Toc154137845)

[איור ‎5.8 – התאמת d & L & r לדוגמא הראשונה. 70](#_Toc154137846)

[איור ‎5.9 – תוצאת הגיבוב של הדוגמא הראשונה בחומרה. 71](#_Toc154137847)

# רשימת טבלאות

[טבלה 1: וקטורי הקבועים – Q. 14](#_Toc154137848)

[טבלה 2: וקטורי הקבועים – S. 15](#_Toc154137849)

[טבלה 3: וקטורי ההזזה r&l. 16](#_Toc154137850)

[טבלה 4: קבועי עמדות ההקשה – t. 16](#_Toc154137851)

[טבלה 5: מאפייני ערכת הפיתוח. 23](#_Toc154137852)

[טבלה 6: הקלט והפלט של התכנון. 26](#_Toc154137853)

[טבלה 7: האותות הפנימיים של התכנון. 26](#_Toc154137854)

[טבלה 8: הקלט והפלט של בלוק ה-receiver. 27](#_Toc154137855)

[טבלה 9: האותות הפנימיים של בלוק ה-.receiver 28](#_Toc154137856)

[טבלה 10: הקלט והפלט של בלוק ה-MD6 Mode. 29](#_Toc154137857)

[טבלה 11: האותות הפנימיים של בלוק ה-MD6 Mode. 30](#_Toc154137858)

[טבלה 12: הקלט והפלט של בלוק ה-cf. 32](#_Toc154137859)

[טבלה 13: האותות הפנימיים של בלוק ה-.cf 32](#_Toc154137860)

[טבלה 14: הקלט והפלא של בלוק ה-N. 33](#_Toc154137861)

[טבלה 15 האותות הפנימיים של בלוק N. 33](#_Toc154137862)

[טבלה 16: הקלט והפלט של בלוק ה-A iterative. 35](#_Toc154137863)

[טבלה 17: הקלט והפלט של בלוק ה-A computation loop. 36](#_Toc154137864)

[טבלה 18: באותות הפנימיים של בלוק ה- A computation loop. 36](#_Toc154137865)

[טבלה 19: הקלט והפלט של בלוק ה-transmitter. 37](#_Toc154137866)

[טבלה 20: האותות הפנימיים של בלוק ה-transmitter. 37](#_Toc154137867)

[טבלה 21: הקלט והפלט של בלוק ה-Button debouncing. 38](#_Toc154137868)

[טבלה 22: האותות הפנימיים של בלוק ה-.Button debouncing 38](#_Toc154137869)

[טבלה 23: המידע הנכנס לאלגוריתם בסימולציה. 67](#_Toc154137870)

# 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 חסכוניות 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.

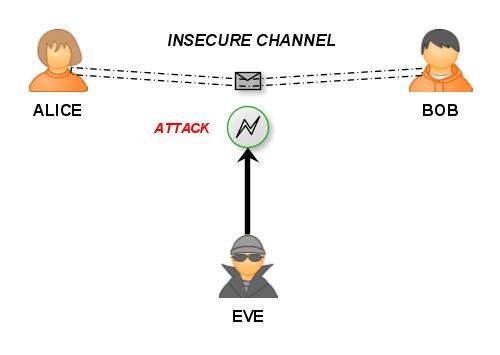
# רקע תיאורטי

## מבוא לקריפטוגרפיה [[1]](#a)

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]](#j).

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.

## פונקציית גיבוב – HASH [[2]](#b)

פונקציית גיבוב היא [פונקציה חד-כיוונית](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%94_%D7%97%D7%93-%D7%9B%D7%99%D7%95%D7%95%D7%A0%D7%99%D7%AA) הממירה קלט באורך כלשהו לפלט באורך קבוע וידוע מראש. [פונקציית גיבוב](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%99%D7%AA_%D7%92%D7%99%D7%91%D7%95%D7%91_%D7%A7%D7%A8%D7%99%D7%A4%D7%98%D7%95%D7%92%D7%A8%D7%A4%D7%99%D7%AA) 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 כ[חתימות דיגיטליות](https://he.wikipedia.org/wiki/%D7%97%D7%AA%D7%99%D7%9E%D7%94_%D7%93%D7%99%D7%92%D7%99%D7%98%D7%9C%D7%99%D7%AA) ,[קודי אימות](https://he.wikipedia.org/wiki/%D7%A7%D7%95%D7%93_%D7%90%D7%99%D7%9E%D7%95%D7%AA_%D7%9E%D7%A1%D7%A8%D7%99%D7%9D) ,[שמירת סיסמאות](https://he.wikipedia.org/wiki/%D7%A1%D7%99%D7%A1%D7%9E%D7%94#%D7%A0%D7%99%D7%A4%D7%95%D7%97_%D7%A1%D7%99%D7%A1%D7%9E%D7%94_(Salting)) ו[מחולל מספרים פסידו-אקראיים](https://he.wikipedia.org/wiki/%D7%9E%D7%97%D7%95%D7%9C%D7%9C_%D7%9E%D7%A1%D7%A4%D7%A8%D7%99%D7%9D_%D7%A4%D7%A1%D7%99%D7%93%D7%95-%D7%90%D7%A7%D7%A8%D7%90%D7%99%D7%99%D7%9D). ביישומים שאינם קריפטוגרפים הן משמשות לעיתים כמזהה ייחודי של קובץ לצורך בדיקת [שלמותו או נכונותו](https://he.wikipedia.org/wiki/%D7%A7%D7%95%D7%93_%D7%AA%D7%99%D7%A7%D7%95%D7%9F_%D7%A9%D7%92%D7%99%D7%90%D7%95%D7%AA) 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- 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 עבור קלט מייצרת ערך גיבוב , קשה למצוא כל קלט אחר כך ש- .

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

* Collision Resistance:

קשה מבחינה חישובית למצוא שני קלטים שונים שיובילו לאותו פלט של פונקציית הגיבוב.

במילים אחרות, עבור פונקציית גיבוב קשה למצוא ו- כך ש- כאשר .

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.

## אלגוריתם MD6 [[3]](#c)

אלגוריתם MD6 הוא פונקציית גיבוב קריפטוגרפית שפותחה על ידיRon Rivest מהמכון הטכנולוגי של מסצ'וסטס (MIT) וצוות מומחי הצפנה בראשותו. אלגוריתםMD6 הוגש כאחת מהצעות האלגוריתמים לתחרות תקן הגיבובSHA-3 של NIST [[4]](#d).

האלגוריתם פועל באמצעות מבנה Merkle-Damgård [[5]](#e), 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.

### הקלט והפלט של ה-MD6

הקלט ל- הן כדלקמן:

* - ההודעה המיועדת להתגבב (חובה).
* – אורך ההודעה המגובבת בסיביות (חובה).
* – ערך מפתח (אופציונלי).
* – בקרת מצב (אופציונלי).
* – מספר סיבובים (אופציונלי).

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

הפלט של הוא - מחרוזת סיביות באורך של סיביות.

#### ההודעה – M

הקלט הראשון ל-MD6 הוא ההודעה אותה המשתמש רוצה לגבב, אורך ההודעה צריך להיות בגודל סיביות.

#### אורך ההודעה המגובבת –d

הקלט השני ל- הוא אורך הסיביות של ההודעה המגובבת אשר ערכיו יכולים לנוע בין.

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

אורכי ה- כפי שנדרש מ-SHA-3 הם: .

#### המפתח – K

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

המפתח הוא קלט אופציונלי, ברירת המחדל כאשר המשתמש בוחר לא להכניס מפתח הוא ואורכו סיביות.

#### פרמטר בקרת המצב – L

הקלט הבא ל-MD6 הוא פרמטר בקרת המצב , המשמש לבחירת אחד ממצבי הפעולה של ה-MD6 אשר יפורטו להלן [בפרק 2.3.3](#_מצבי_הפעולה_של). אורך פרמטר בקרת המצב הוא .

פרמטר בקרת המצב הוא אופציונלי, ברירת המחדל כאשר המשתמש בוחר לא להכניס את הקלט הוא .

#### מספר הסיבובים –r

הקלט הבא ל-MD6 הוא מספר הסיבובים , המשמש למספר הסיבובים של הפעלת פונקציית הדחיסה של ה-MD6 כמוסבר להלן [בפרק 2.3.4](#_פונקציית_הדחיסה). אורך מספר הסיבובים הוא .

מספר הסיבובים הוא אופציונלי, ברירת המחדל כאשר המשתמש בוחר לא להכניס את הקלט הוא .

### קבועים ב-MD6

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

* וקטור הקבועים –
* וקטור הקבועים –
* וקטורי קבועי ההזזה (shift) –
* קבועי עמדות ההקשה (tap position) – t

#### וקטור הקבועים Q

וקטור הקבועים 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](#_פונקציית_הדחיסה).

ערכי ה- מוצגים בטבלה 1.

טבלה 1: וקטורי הקבועים – Q.

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

#### וקטור הקבועים S

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

ערכי ה- מוצגים בטבלה.

טבלה 2: וקטורי הקבועים – S.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0x058e571c26c8eadc | **2** | 0x0347cace1376567e | **1** | 0x0123456789abcdef | **0** |
| 0x3e5330e1c66763a0 | **5** | 0x16291870f3233150 | **4** | 0x0a1cec3869911f38 | **3** |
| 0xedee878b23c997e1 | **8** | 0xdf7f828511f68d60 | **7** | 0x4eb7614288eb84e0 | **6** |
| 0xcc55b5def66e796e | **11** | 0x47aa9bafeb25d8e7 | **10** | 0xbadd8d976792a863 | **9** |
| 0xa3cb28f523a234b7 | **14** | 0xe165147a91d1fc5b | **13** | 0xd8baeb3dc8f8bbfd | **12** |
| 0x736e072ef5fdaa3d | **17** | 0xa93fe2d7eaec961e | **16** | 0x6497516b67646dcf | **15** |
| 0x475031f53753a7ca | **20** | 0x3aa818ba9bb972b5 | **19** | 0x95dc0c5dcfdede5a | **18** |
| 0xe6f1892e2ef3f873 | **23** | 0xda7084d795695829 | **22** | 0xcdb0636b4aa6c814 | **21** |
| 0x89facff1a710bb1e | **26** | 0x7cf5a6b8d388790f | **25** | 0xaff2925c79c638c7 | **24** |
| 0x5c963709cce469b4 | **29** | 0x37cbfac4f462375a | **28** | 0x12e55d626a21fd3d | **27** |
| 0x55d1b04b5bdef123 | **32** | 0xb36898253ffdbf11 | **31** | 0xe93c6c129dec9ac8 | **30** |
| 0x0dfb03dc96a7ce7b | **35** | 0xfab2e097b7b92366 | **34** | 0xfab2e097b7b92366 | **33** |
| 0x6c9f16e7c5cd0978 | **38** | 0x27cf0a7372f4e72c | **37** | 0x1ae70539296a52d6 | **36** |
| 0xc5aff9bb36577462 | **41** | 0x435f5c9d1b3b3c21 | **40** | 0xb92f2f4e8f9f1bd0 | **39** |
| 0xff588ade22c96ff7 | **44** | 0xd6ac656f9176d56b | **43** | 0xca5e33f748abace5 | **42** |
| 0x2685d8f1fa69c1be | **47** | 0x1a42ac78ef26a09f | **46** | 0x8da1973c6593904f | **45** |
| 0x4b39c70a7f8d4951 | **50** | 0xbd14a2c5adc4c738 | **49** | 0x6f0a7162d4f242dc | **48** |
| 0x848970524854b56b | **53** | 0xfbc4d829b63a7ce5 | **52** | 0xd5624c14db1fdba2 | **51** |
| 0x357d431362d8209c | **56** | 0x1336c1c9217e104e | **55** | 0x0913a0a490adeff7 | **54** |
| 0xa9bcd09dfa814721 | **59** | 0xe4d6484eef40c2d0 | **58** | 0x5bebc427e5b041b8 | **57** |
| 0x3fb6856a7808fc6d | **62** | 0x96d383f5ae065be6 | **61** | 0x726961bad503c963 | **60** |
| 0xe1d5ac52606797a9 | **65** | 0xd8ea9729a0236d54 | **64** | 0x4c7d8ad4d01134fa | **63** |
| 0xa2bad8a4e0eaa8f3 | **68** | 0xa2bad8a4e0eaa8f3 | **67** | 0xa2bad8a4e0eaa8f3 | **66** |
| 0x0d6bd095f6422ed5 | **71** | 0x7a96c425e798bc9d | **70** | 0x7a96c425e798bc9d | **69** |
| 0x6969852a221d0fc8 | **74** | 0x24bc83d5910ce574 | **73** | 0x1bd661aac884532a | **72** |
| 0x83e5dd22dd4bc0e5 | **77** | 0x54b596a8ec5b2021 | **76** | 0xb3d28a54643f1010 | **75** |
| 0x0994b68a5928c3f6 | **80** | 0x04ca7a45be96416b | **79** | 0x83e5dd22dd4bc0e5 | **78** |
| 0x5ec43bd38d8655b0 | **83** | 0x36621da944c3cc98 | **82** | 0x1239ef94b271444c | **81** |
| 0x4831d69854232503 | **86** | 0xbc10aa4c3a111301 | **85** | 0xef8875261f08eec0 | **84** |
| 0x91f9bb43ddf6631b | **89** | 0xf0f49de17cebd10d | **88** | 0xd0726fb0ac674f06 | **87** |
| 0xfc8b539bb722919c | **92** | 0x57c5298d5b918f4e | **91** | 0x32e2f486bfc88537 | **90** |
| 0x356c15592df94826 | **95** | 0x133e0bec94eec7d3 | **94** | 0x8917e5b64a65a2b9 | **93** |
| 0xa940ed4eb768b951 | **98** | 0xe4a057e7dba678f8 | **97** | 0x5bd82ab37fd3d86c | **96** |
| 0x38076df62f84756d | **101** | 0x940337bb95c23ce6 | **100** | 0x73811a9d4af1fba3 | **99** |
| 0xc02de931a83e60a9 | **104** | 0xc01eb4d8d61dd054 | **103** | 0x400f9b6c7b0caffa | **102** |
| 0xc1484f098142868f | **107** | 0xc0a426c4c0b18247 | **106** | 0xc05a1262705881f3 | **105** |
| 0xc873b0480e19b5df | **110** | 0xc4317824050e9cbf | **109** | 0xc390dc1202858b9f | **108** |
| 0x92eb03c0408f26bf | **113** | 0xf1fd01a03045125f | **112** | 0xd0f6e0901832ee3f | **111** |
| 0xe96f1603044a745c | **116** | 0x5cbf0a010237dc3e | **115** | 0x37d70500811b4bdf | **114** |
| 0xf95ffe1f3e7e6e26 | **119** | 0x54af5e0f1d2dd5d3 | **118** | 0xb3df2e070c94acb9 | **117** |
| 0x08a8befe4342cb56 | **122** | 0x045c7e7fb1b1a6fb | **121** | 0x83ae3e3f58d8926d | **120** |
| 0x54747973121a2b50 | **125** | 0x33b23cf9090ff6f8 | **124** | 0x1151ff7c86855dac | **123** |
| 0x02c20c9ffc9d2b63 | **128** | 0x81e1e74f6c4cf6e1 | **127** | 0xf8f8b2e724345da0 | **126** |
| 0x1810657a58b62af8 | **131** | 0x0c0833fdbe5bf66c | **130** | 0x078419bedd3f5de6 | **129** |
| 0xa0f2acd3852122e0 | **134** | 0x607197694290f1a0 | **133** | 0x20308af4b1485f50 | **132** |
| 0x67e4a69831ea5a65 | **137** | 0xa2fa724c18e7c9e2 | **136** | 0x61f5d9260e634761 | **135** |
| 0x8234fb410b9f65ca | **140** | 0x79925de087cd3375 | **139** | 0xacc9cfb043f4feea | **138** |
| 0x1fc7938f74a9a674 | **143** | 0x0ee369872a56922a | **142** | 0x06793483173b8e15 | **141** |
| 0x832a55fd398ff120 | **146** | 0x791dcbbe9ec55f10 | **145** | 0x2c8ea59fcd72cac8 | **144** |
| 0x1463296e684cce64 | **149** | 0x0bb914f7a63445e2 | **148** | 0x0554eb7b531a2361 | **147** |
| 0xc21e0c72825a09c0 | **152** | 0x418fe739c13fe770 | **151** | 0x38c752dcf09d52e8 | **150** |
| 0xdea062971e9c7207 | **155** | 0xce58314b0d4c3e03 | **154** | 0xc62c18e504b41a01 | **153** |
| 0x4a029b3fa4be5e3f | **158** | 0xbd818ddf525dca1f | **157** | 0xef4087af393ca60f | **156** |
| 0x8e151179d9434bdb | **161** | 0xfe0ae8fcfeb126bd | **160** | 0xd605b47e6d58f25e | **159** |
| 0x7ecfcccb8e14ac9c | **164** | 0x2e674765450a744e | **163** | 0x1e3b22f2b287dc37 | **162** |
| 0x2859265840d89322 | **167** | 0x1c2cf22c307e6ed1 | **166** | 0x8f9e5916182dd5b8 | **165** |

#### וקטורי ההזזה r&l

וקטורי הקבועים 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](#_פונקציית_הדחיסה) לכל מילה משתמשים בערך אחר של הזזה לפי המיקום בוקטור.

ערכי ברירת המחדל מוצגים בטבלה 3.

טבלה 3: וקטורי ההזזה r&l.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **%16** |
| 12 | 6 | 7 | 11 | 13 | 7 | 15 | 14 | 7 | 2 | 12 | 11 | 10 | 13 | 5 | 10 |  |
| 9 | 31 | 5 | 15 | 8 | 19 | 2 | 6 | 15 | 27 | 9 | 15 | 16 | 9 | 24 | 11 |  |

#### קבועי עמדות ההקשה t

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

קבועי עמדות ההקשה צריכים להיות בטווח .

ערכי ברירת המחדל מוצגים בטבלה 4.

טבלה 4: קבועי עמדות ההקשה – t.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| 67 | 31 | 21 | 18 | 17 |

### מצבי הפעולה של ה-MD6

באלגוריתם 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 – .

* מצב פעולה מקבילי:

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 , וכל 4 בלוקים נכנסים לפונקציית דחיסה אחת (נרחיב עליה [בפרק 2.3.4](#_פונקציית_הדחיסה)) 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 .

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

תמונה שמכילה שמים, מכשיר, קנה מידה, קו

התיאור נוצר באופן אוטומטי

איור ‎2.2 – מצב הפעולה המקבילי [[3]](#c).

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.

* מצב פעולה טורי:

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 .

תמונה שמכילה קו, טיפוגרפיה

התיאור נוצר באופן אוטומטי

איור ‎2.3 – מצב הפעולה הטורי [[3]](#c).

* מצב פעולה היברידי:

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-, 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.

תמונה שמכילה מכשיר, קנה מידה

התיאור נוצר באופן אוטומטי

איור ‎2.4 – מצב הפעולה ההיברידי [[3]](#c).

### פונקציית הדחיסה

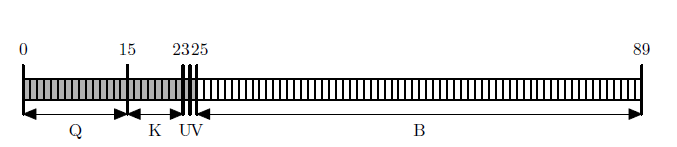
פונקציית הדחיסה, היא פונקציית החישוב המרכזית של אלגוריתם MD6, כל פונקציית דחיסה מקבלת 64 מילים של 64 סיביות שהם 4096 סיביות של הודעה, ומוציאה גיבוב של 16 מילים של 64 סיביות שהם 1024 סיביות מידע. בפרק זה נפרט על מבנה הפונקציה ואופן פעולתה.

#### מבנה פונקציית הדחיסה

קלט פונקציה הדחיסה הוא בלוק מידע בעל 64 מילים, והוא נטמע בתוך בלוק נתונים 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

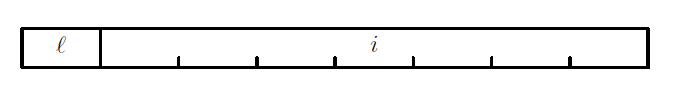
על הכניסות פירטנו בפרקים לעיל, נפרט על הכניסות .



איור ‎2.5 – מצב הפעולה ההיברידי [[3]](#c).

##### 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 מכילים את ערך ה-index וה-byte שנשאר מכיל את ערך ה-level, כמופיע באיור 2.6.



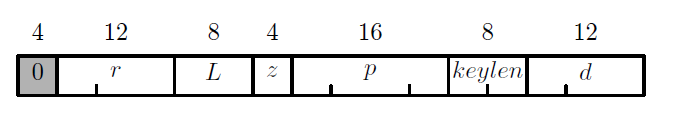
איור ‎2.6 – פריסת מזהה הצומת הייחודי U [[3]](#c).

##### 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]](#c).

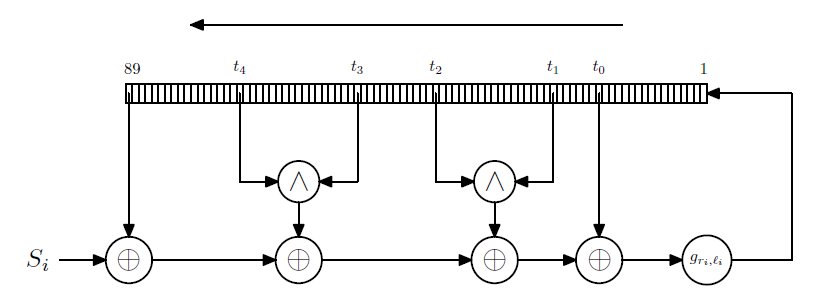
#### פעולת החישוב של פונקציית הדחיסה

כמו שהוסבר בפרק הקודם, פונקציית הדחיסה מעבדת את בלוק הנתונים , ומפעילה את לולאת החישוב המוצגת ב[משוואה 1](#משוואה1).

{

}

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]](#c).

בכל סיבוב, הפונקציה מחשבת 16 מילים חדשות, המילה המחושבת הראשונה תתווסף לבלוק הנתונים כך שכעת גודלו יהיה 90 מילים.

בשורת החישוב הראשונה, עושים בין מילה בוקטור המשתנה בכל סיבוב, לבין שני מילים בבלוק הנתונים כאשר הבחירה של המילים נעשית ע"י index המיקום ואחד מקבועי עמדות ההקשה .

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

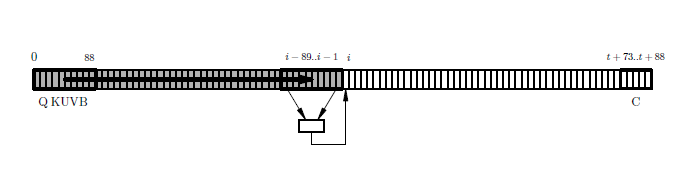
בשורה השלישית עושים עם התוצאה של השורה השנייה ועם אותה תוצאה אשר מוזזת ימינה לפי וקטורי ההזזה .

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

התוצאה מוכנסת לבלוק הנתונים במיקום של ה-index – .

קבועי עמדות ההקשה כמו שהסברנו לעיל נמצאים בטווח הנ"ל 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.

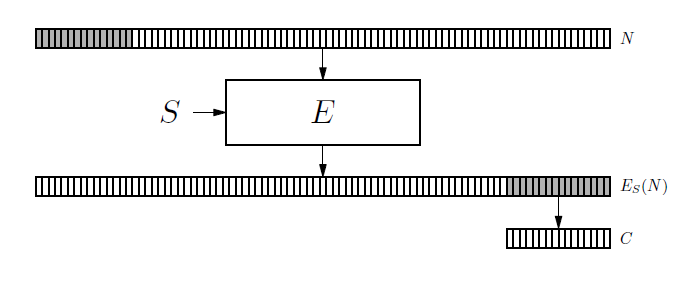


איור ‎2.9 – אופן הפעולה של לולאת החישוב [[3]](#c).

#### 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]](#c).

# פיתוח ושיטות

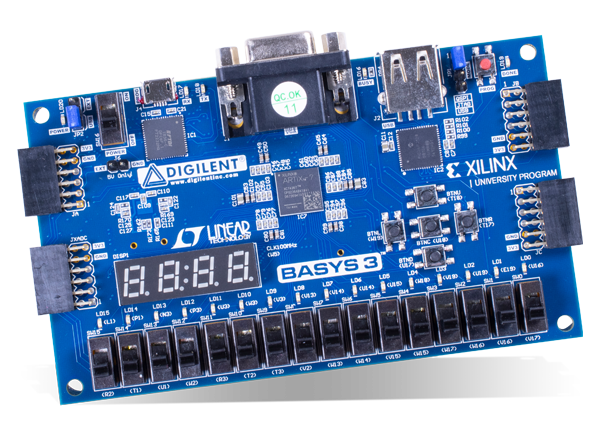
במסגרת הפרויקט, מומש אלגוריתם MD6 אשר מכיל פונקציה דחיסה אחת בלבד אשר פועלת לפי מצב הפעולה הסטנדרטי, עקב מחסור במשאבי רכיב החומרה כפי שיוסבר להלן.

קישור למימוש מצבי הפעולה (הנמצאים בתיקיית הפרויקט) נמצא [בנספח א](#_נספח_א_–).

פרק זה מחולק באופן הבא:

* ערכת הפיתוח ורכיב ה-FPGA
* מעטפת להעברת נתונים
* מימוש האלגוריתם בשפת חומרה
* הטמעה על רכיב ה-FPGA
* יצירת קוד תוכנה עם תוספת ממשק משתמש

## ערכת הפיתוח ורכיב ה-FPGA [[8]](#h)



איור ‎3.1 – ערכת הפיתוח Basys-3 [[11]](#k).

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.

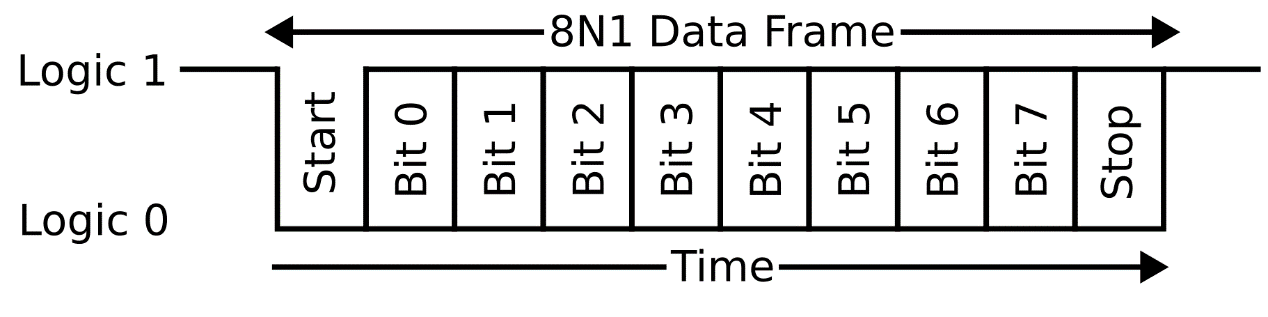
טבלה 5 מציגה את המאפיינים העיקריים של ערכת הפיתוח.

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 Serial Flash 32Mb |
| תצוגות | תצוגה אחת בת 4 ספרות בת 7 פלחים |
| מתגים ונוריות | * 16 מתגים * 16 נורות לד * 5 כפתורים |
| שעונים | מתנד בעל תדר 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 |

## 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]](#i)  
(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]](#h).



איור ‎3.2 – מבנה שליחת המידע בתקשורת UART מסוג 8N1 [[12]](#l).

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.

## 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.

תמונה שמכילה תרשים, טקסט, תוכנית, שרטוט טכני

התיאור נוצר באופן אוטומטי

איור ‎3.4 – דיאגרמת הבלוקים של המימוש.

פירוט ה-IO והאותות הפנימים של התכנון מופיע בטבלאות 6 ו-7.

טבלה 6: 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 7: 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.

### 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 8 ו-9.

איור ‎3.5 – סימבול בלוק ה-receiver.

תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי

איור ‎3.6 – ה-bus של תתי המידע.

טבלה 8: 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 |

טבלה 9: 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 |

### בלוק ה-MD6 Mode

איור ‎3.7 – סימבול 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 10 ו-11.

תמונה שמכילה טקסט, תרשים, תוכנית, קו

התיאור נוצר באופן אוטומטי

איור ‎3.8 – בלוק ה-MD6 Mode.

טבלה 10: 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 11: האותות הפנימיים של בלוק ה-MD6 Mode.

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

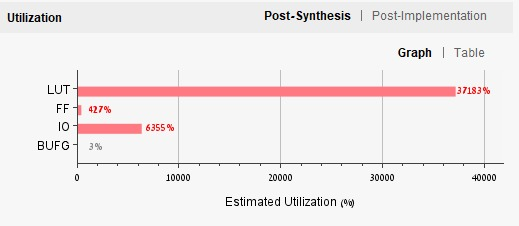
#### בלוק ה-cf

בלוק ה-cf (Compression function) כמוצג באיור 3.9 זו פונקציית הדחיסה המרכזית של אלגוריתם MD6 כמפורט [בפרק 2.3.](#_אלגוריתם_ה-MD6_[3])

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).

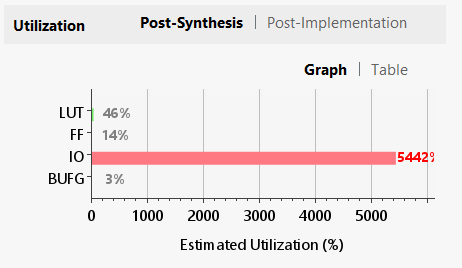
איור ‎3.9 – סימבול בלוק ה-cf.



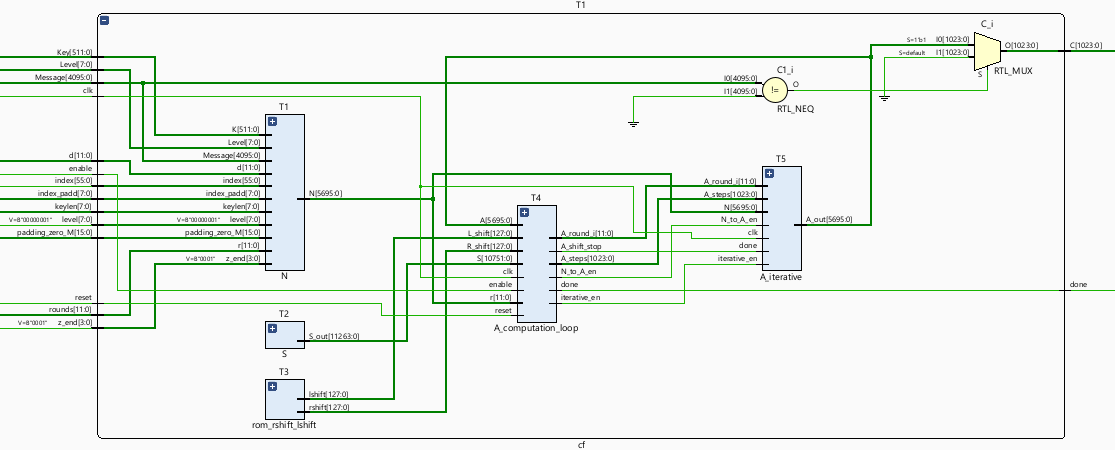
איור ‎3.10 – אחוז רכיבי החומרה המשומשים בתכנון הראשוני של בלוק ה-cf.

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 12 ו-13.

טבלה 12: 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 13: האותות הפנימיים של בלוק ה-.cf

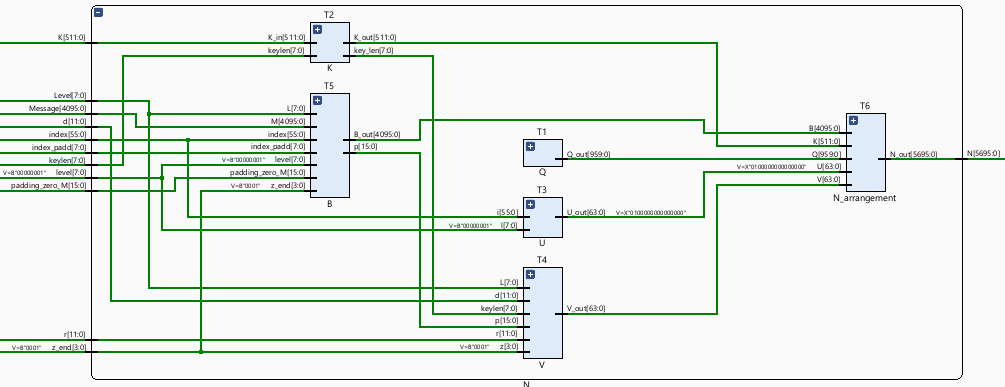
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם האות | גודל | תיאור | מבלוק | לבלוק |
| N | 5696 סיביות | בלוק הנתונים N | N | A iterative |
| A steps | 1024 סיביות | 16 המילים הנוצרות בפונקציית הדחיסה | A computation loop | A iterative |
| A round i | 12 סיבות | מספר הסיבוב הנוכחי | A computation 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 |

##### 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](#_פונקציית_הדחיסה).

פירוט ה-IO והאותות הפנימיים של ה-N מופיע בטבלאות 14 ו-15.

איור ‎3.13 – סימבול בלוק ה-N. מבחוץ.



איור ‎3.14 – בלוק ה-N.

טבלה 14: הקלט והפלא 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 15 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 |

##### תמונה שמכילה טקסט, צילום מסך, תרשים, קו התיאור נוצר באופן אוטומטיבלוק ה-S

בלוק ה- הוא בלוק המכיל את וקטור הקבועים , אשר משמשים כחלק מחישוב פונקציית הדחיסה כמוסבר [בפרק 2.3.2](#_קבועים_ב-MD6).

בלוק ה- 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- הוא , ובכל בלוק נכנס מידע בגודל 1024 סיביות, לכן מספר בלוקי ה-rom שהוצרכו זה (הבלוק האחרון של ה-rom מכיל בחציו הראשון את 512 האחרונים של וקטור הקבועים וחציו השני מכיל אפסים).

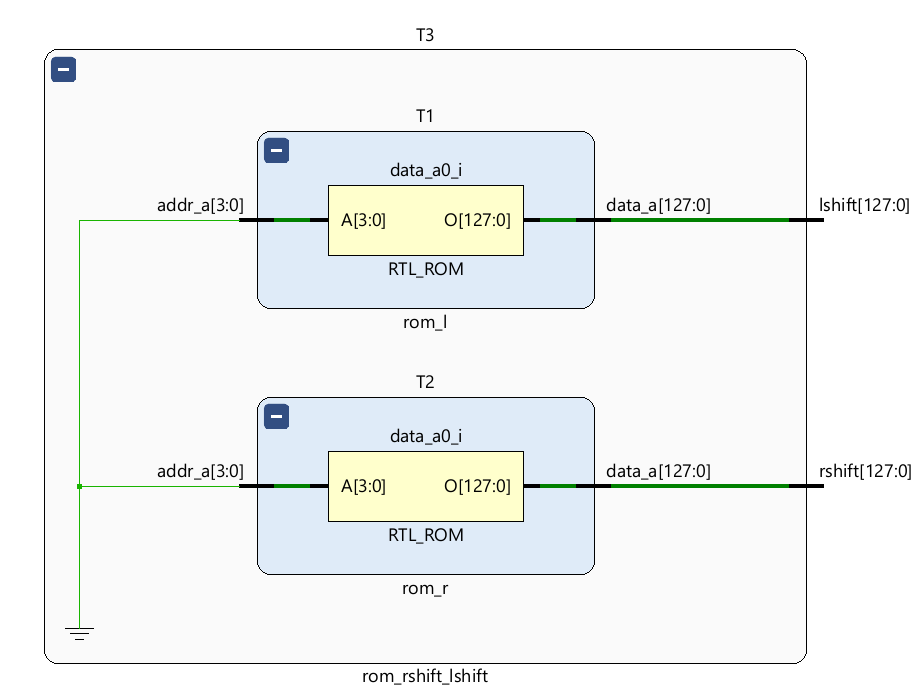
איור ‎3.15 – סימבול בלוק ה-S.

תמונה שמכילה טקסט, צילום מסך, תוכנה, מספר

התיאור נוצר באופן אוטומטי

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

##### 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](#_קבועים_ב-MD6).

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- rshiftול-lshift מספיק בלוק rom אחד לכל אחד, כי כל אחד צורך מקום של  *סיביות.*

איור ‎3.17 – סימבול בלוק ה-rom rshift lshift.

##### תמונה שמכילה טקסט, צילום מסך, תרשים התיאור נוצר באופן אוטומטיבלוק ה- A iterative

בלוק A iterative תפקידו לייצור את האיטרטיביות של התכנון, כאשר הוא מקבל את הבלוק הנתונים ההתחלתי ואת 16 המילים החדשות שנוצרו בבלוק החישוב כמוצג באיור 3.18.  
בסיבוב הראשון הוא מוציא לבלוק החישוב את בלוק הנתונים , 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 16 (אין אותות פנימיים).

איור ‎3.18 – סימבול בלוק ה-rom rshift lshift.

טבלה 16: 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 |

##### תמונה שמכילה טקסט, צילום מסך, תרשים התיאור נוצר באופן אוטומטיבלוק ה-A computation loop

בלוק ה- A computation loopמכיל את האלגוריתם של לולאת פונקציית הדחיסה כך שבכל סיבוב הבלוק מקבל בלוק נתונים של 89 מילים ומוציא 16 מילים חדשות כמוסבר ב[משוואה 1](#משוואה1), 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 17 ו-18.

איור ‎3.19 – סימבול בלוק ה-A computation loop.

טבלה 17: 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 18: באותות הפנימיים של בלוק ה- A computation loop.

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

### 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 19 ו-20.

איור ‎3.20 – סימבול בלוק ה-transmitter.

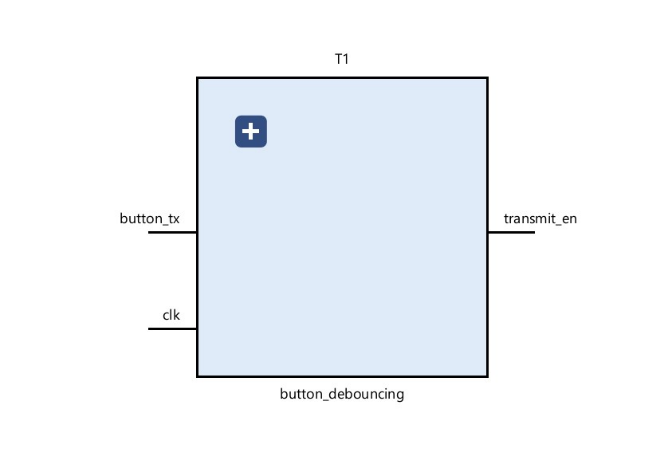
טבלה 19: הקלט והפלט של בלוק ה-transmitter.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 20: 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 |

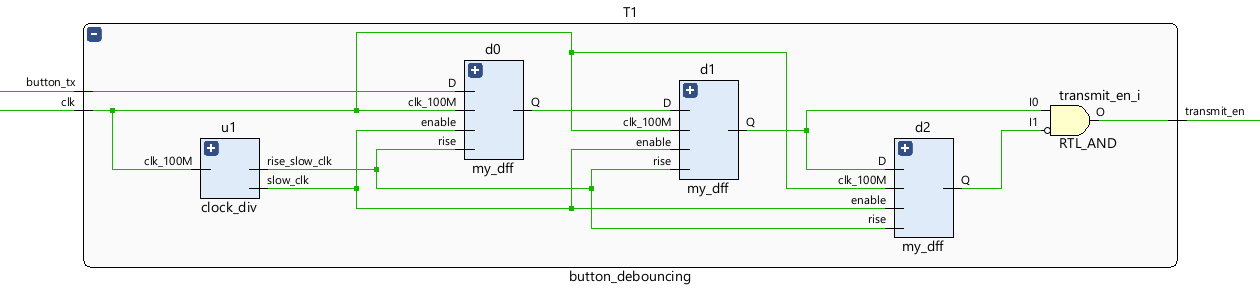
### בלוק ה- Button Debouncing

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.

איור ‎3.21 – סימבול בלוק ה- Button debouncing.

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 21 ו-22.

טבלה 21: 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 22: 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 |

## 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.

### 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.

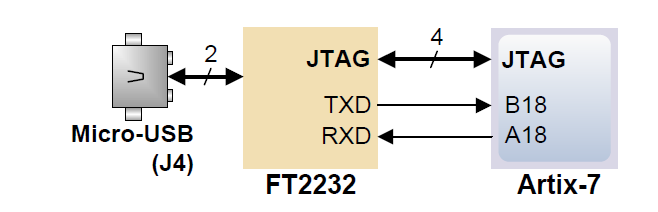
להלן פירוט חיבורי היציאות לפינים:

* + 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 ‎3.23 – 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.

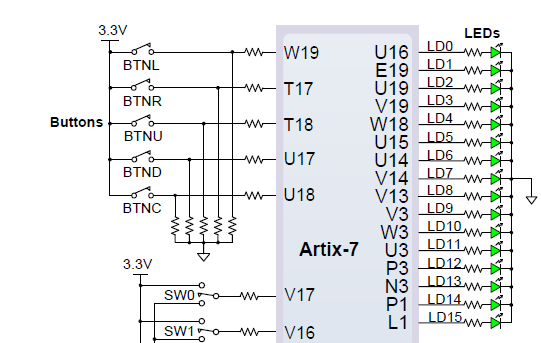
Done keylen – מחובר לפין U15 אשר מחובר לנורה LD5.

Done padding – מחובר לפין U14 אשר מחובר לנורה LD6.

Done rx – מחובר לפין V14 אשר מחובר לנורה LD7.

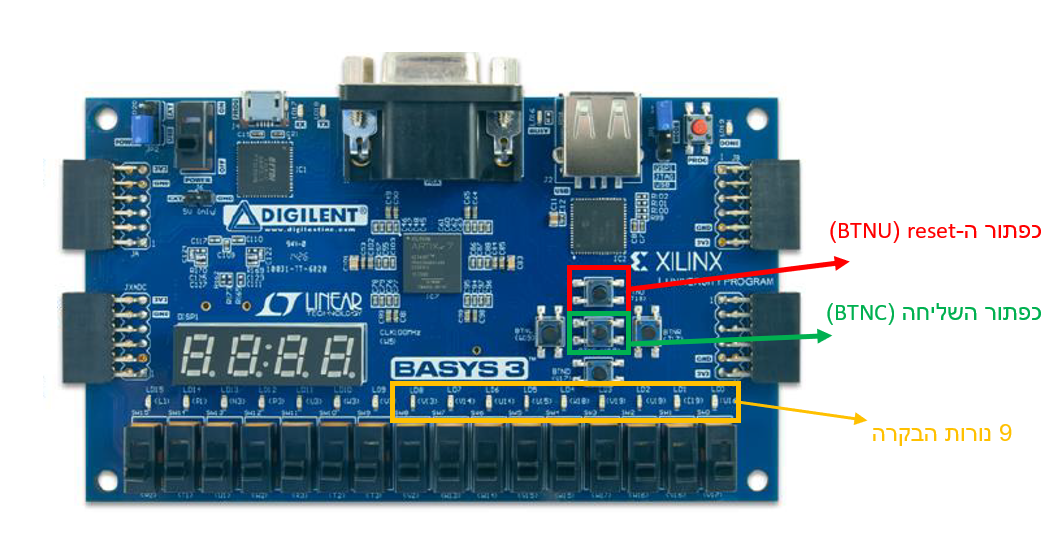
Done MD6 – מחובר לפין V13 אשר מחובר לנורה LD8.

כמוצג באיור 3.24.



איור ‎3.24 – חיבור הנוריות והלחצנים לפינים בערכת הפיתוח.

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



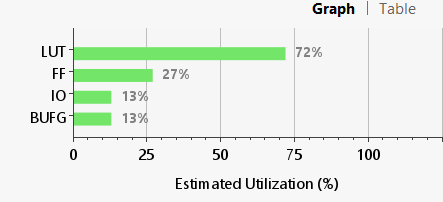
איור ‎3.25 – מיקום הכפתורים והנוריות על ערכת הפיתוח.

### 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](#_נספח_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.

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטי

איור ‎3.26 – 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 . 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 . 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.

### 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.

תמונה שמכילה צילום מסך, טקסט, צבעוני

התיאור נוצר באופן אוטומטי

איור ‎3.27 – מיקום רכיבי החומרה המשומשים לאלגוריתם.

תמונה שמכילה צילום מסך, תצוגה, טקסט, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

איור ‎3.28 – 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.

תמונה שמכילה טקסט, צילום מסך, קו, תרשים

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

איור ‎3.29 – 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.

תמונה שמכילה טקסט, גופן, קו, מספר

התיאור נוצר באופן אוטומטי

איור ‎3.30 – ניתוח התזמון של התכנון.

### 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.

## 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.

* קבלת המידע מהמשתמש:

כמוסבר [בפרק 2.3](#_אלגוריתם_ה-MD6_[3]) 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

* יצירת מידע עזר:

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

* + keylen – אורך המפתח ב-byte.
  + Padding M – אורך ריפוד המידע להודעה.
  + Index M – מספר פונקציות הדחיסה.
* המרת המידע לפורמט byte:

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

* ריפוד המידע באפסים וסידורו:

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

* M – 512 bytes
* d – 2 bytes
* K – 8 bytes
* L – 1 byte
* r – 2 bytes
* keylen – 1 byte
* padding M – 2 bytes
* Index M – 1 byte

בנוסף חוץ מההודעה והמפתח, כל בלוק מידע עובר היפוך כך שה-byte הראשון מועבר לסוף וכן הלאה, כדי להתאים את שלחית המידע לאלגוריתם.

* שרשור כל המידע:

לשם שליחת המידע דרך תקשורת הטורית, כל המידע משורשר אחד אחרי השני לשליחה בצורה יעילה ומהירה.

* שליחת וקבלת המידע דרך התקשורת הטורית:

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().

# תוצאות

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

## 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:

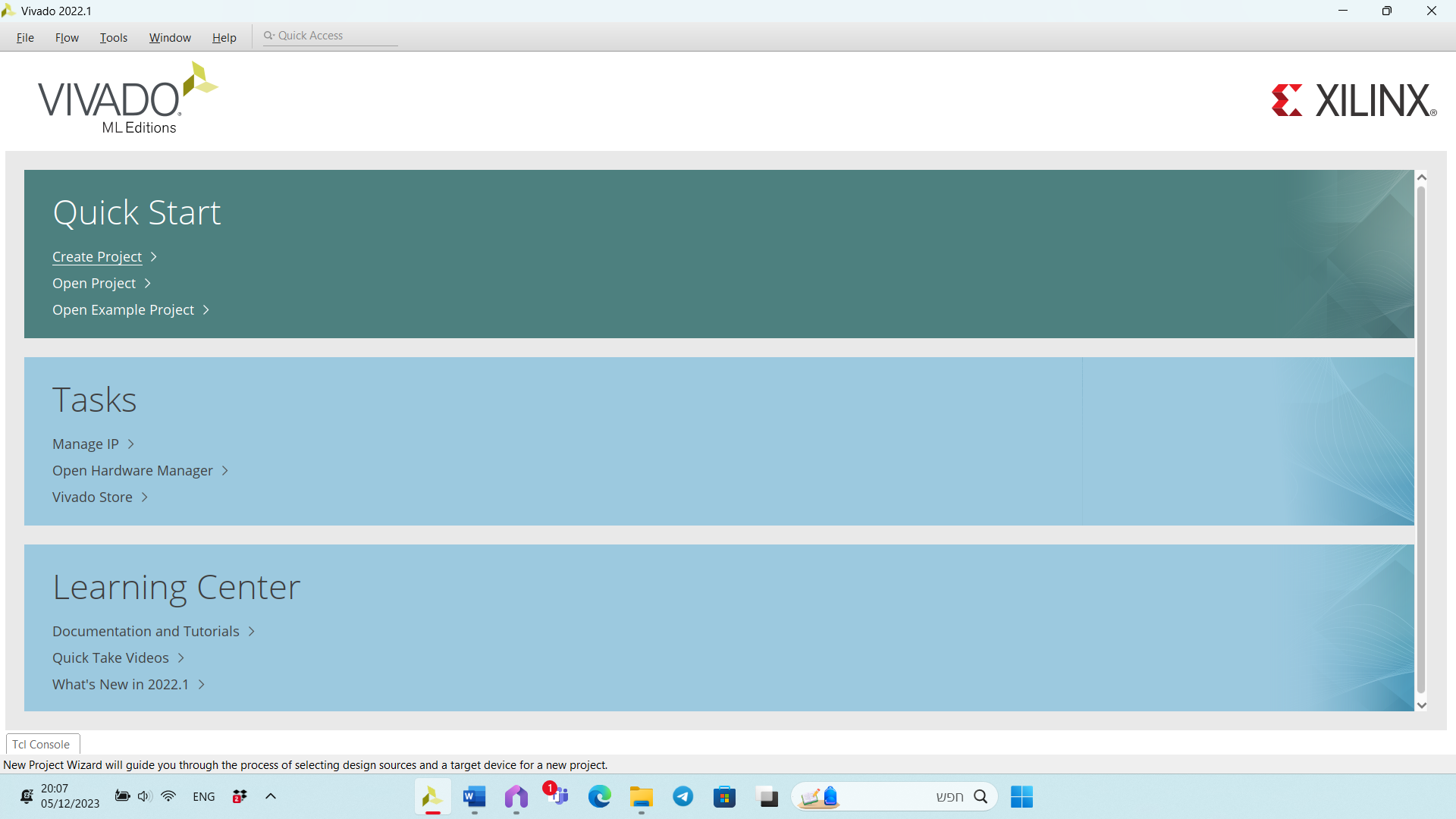
* תיקיית הקבצים של הפרויקט.
* ערכת הפיתוח .Basys3
* כלי התוכנה VIVADO.

קישור לתיקיית קבצי הפרויקט ומדריך בוידאו להפעלת האלגוריתם מצורפים [בנספח א'](#_נספח_א_–)

### תהליך הטמעת אלגוריתם MD6 על רכיב ה-FPGA

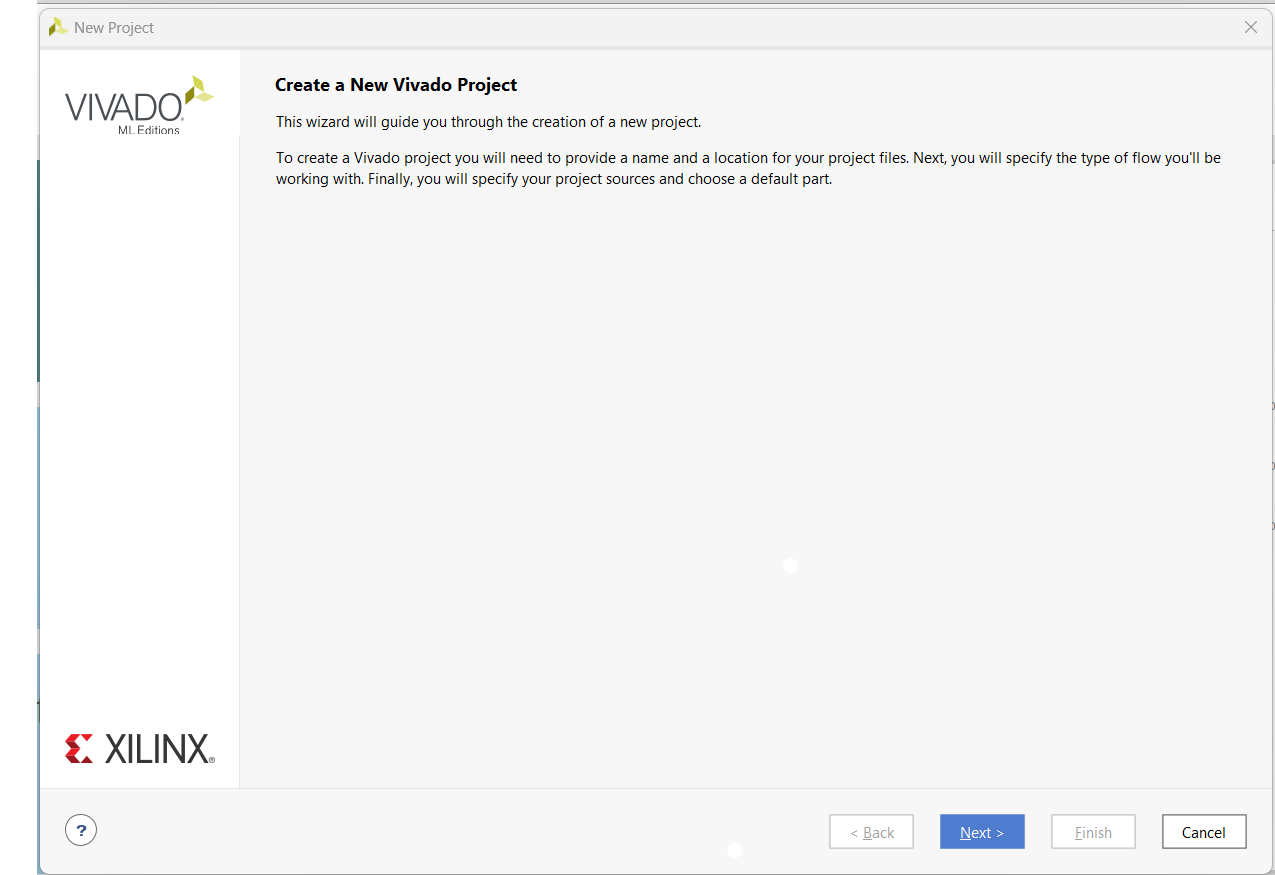
פרק זה מפרט את תהליך הטמעת אלגוריתם MD6 על רכיב ה-FPGA.

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



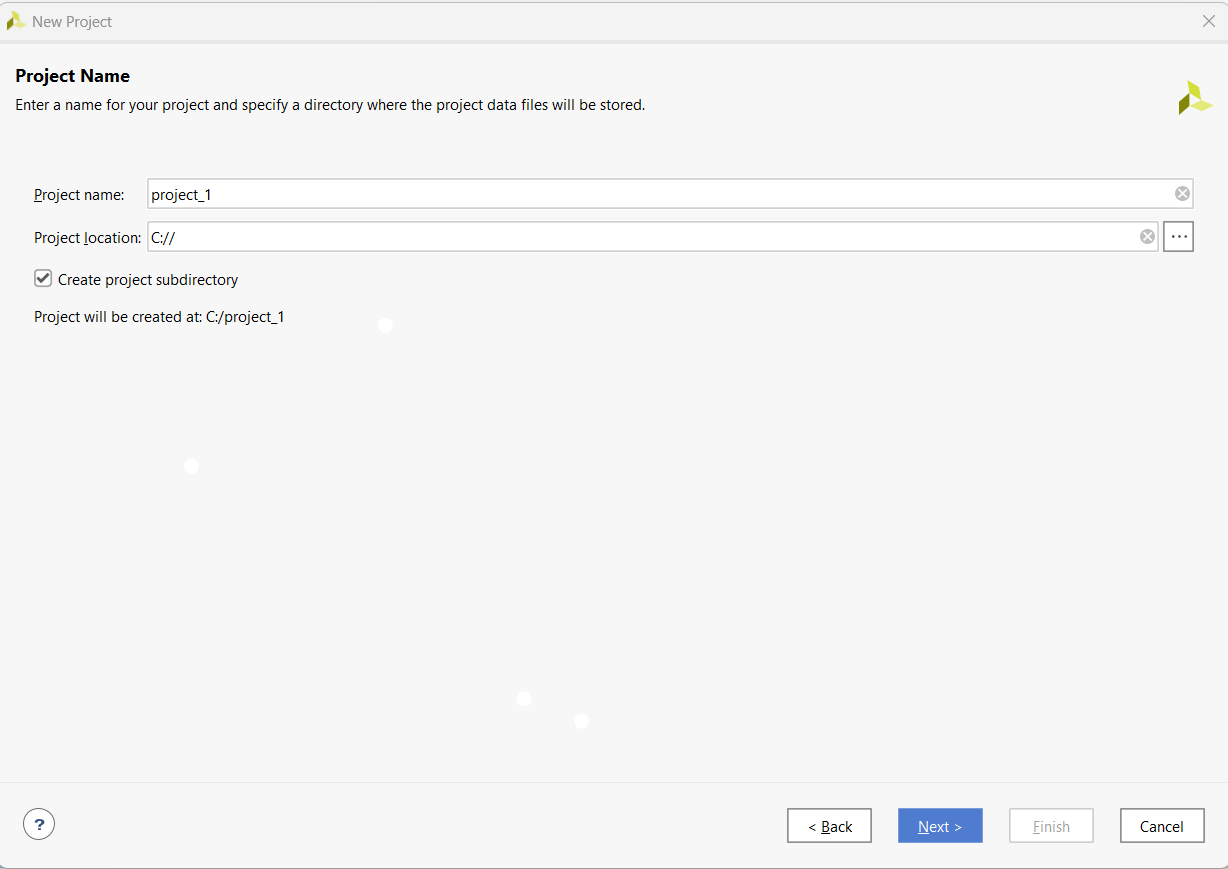
איור ‎4.1 – חלון הפתיחה של כלי התוכנה VIVADO.

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



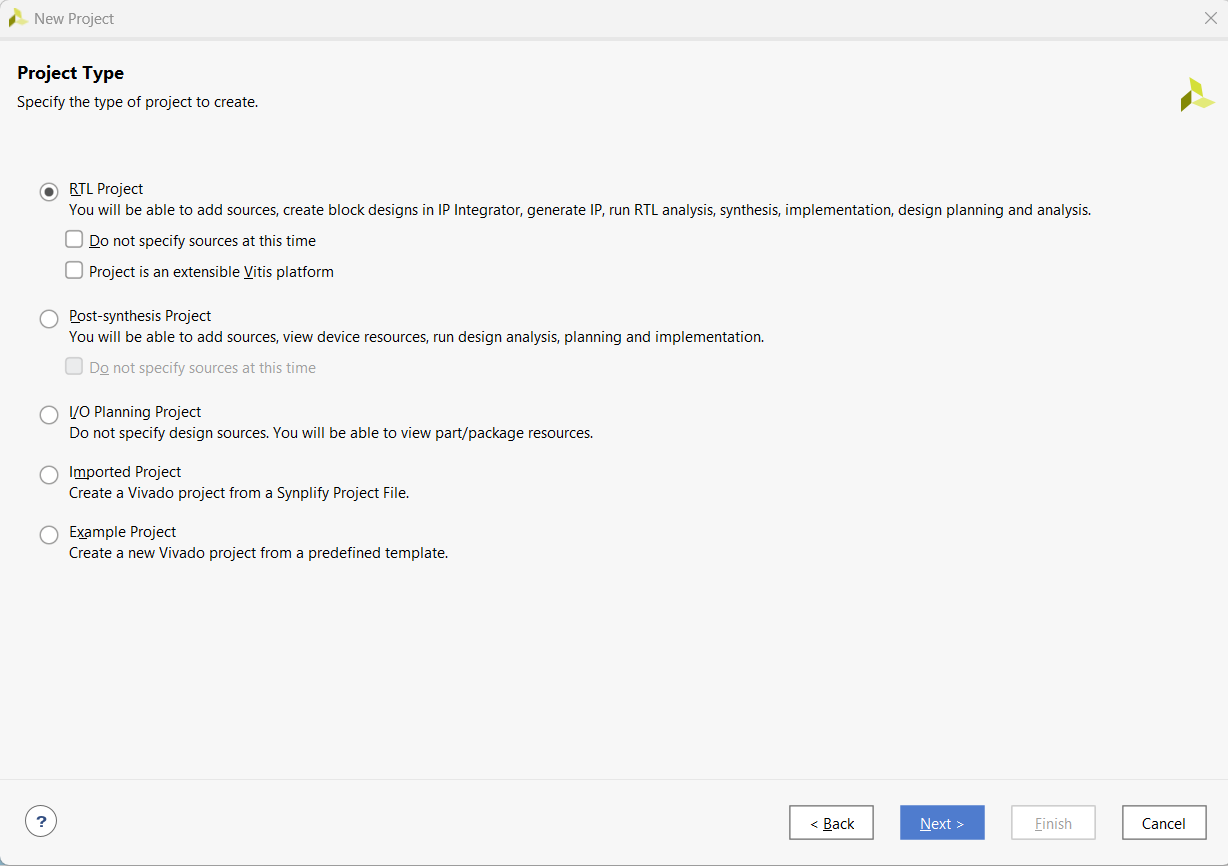
איור ‎4.2 – חלון ה-Create a New Vivado Project.

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



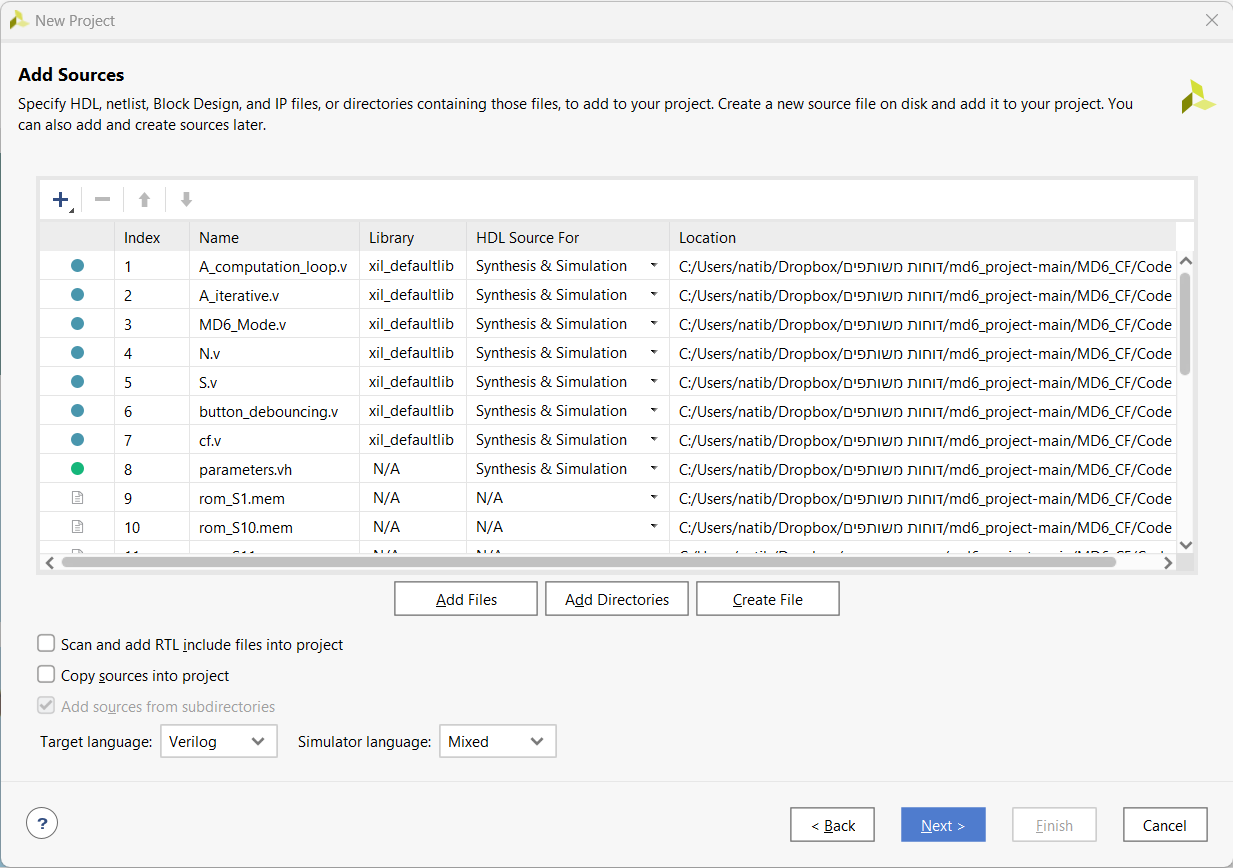
איור ‎4.3 – חלון ה-Project Name.

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



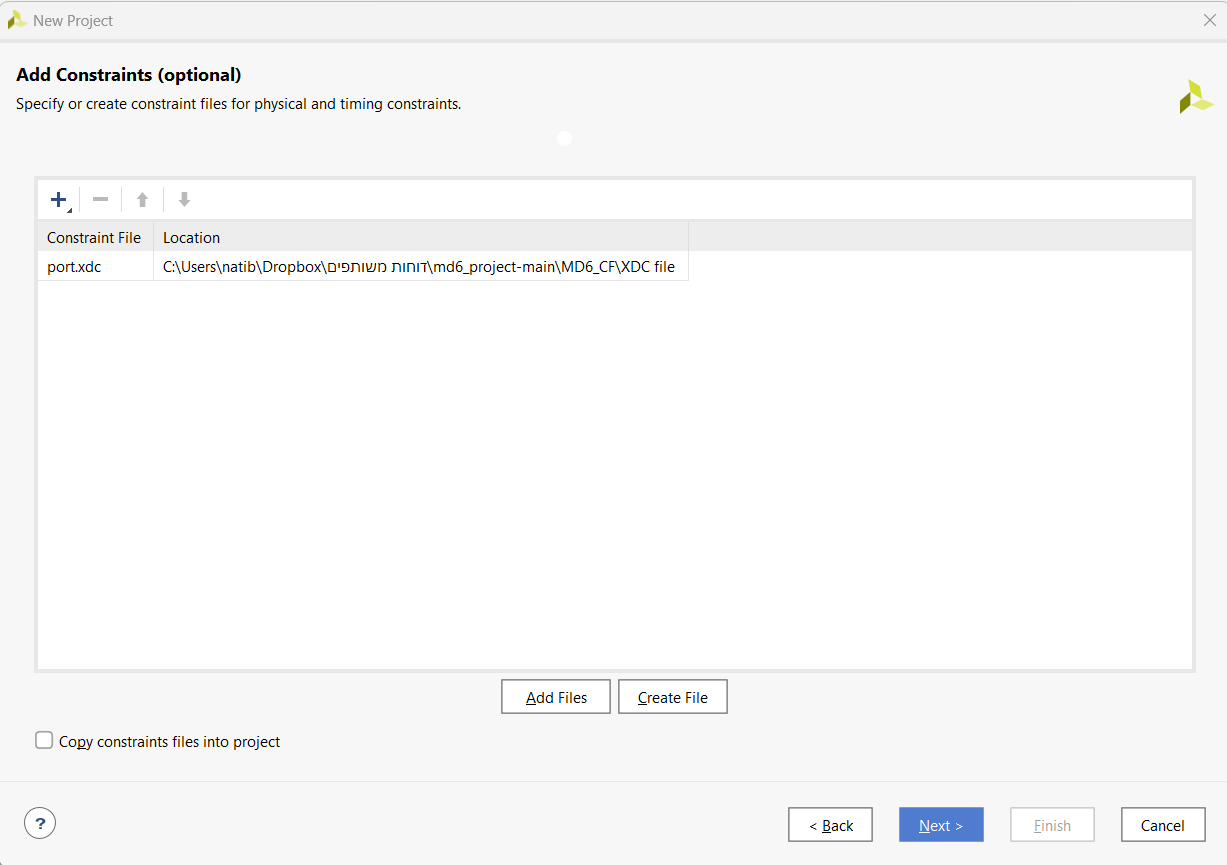
איור ‎4.4 – חלון ה-Project Type.

* 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.



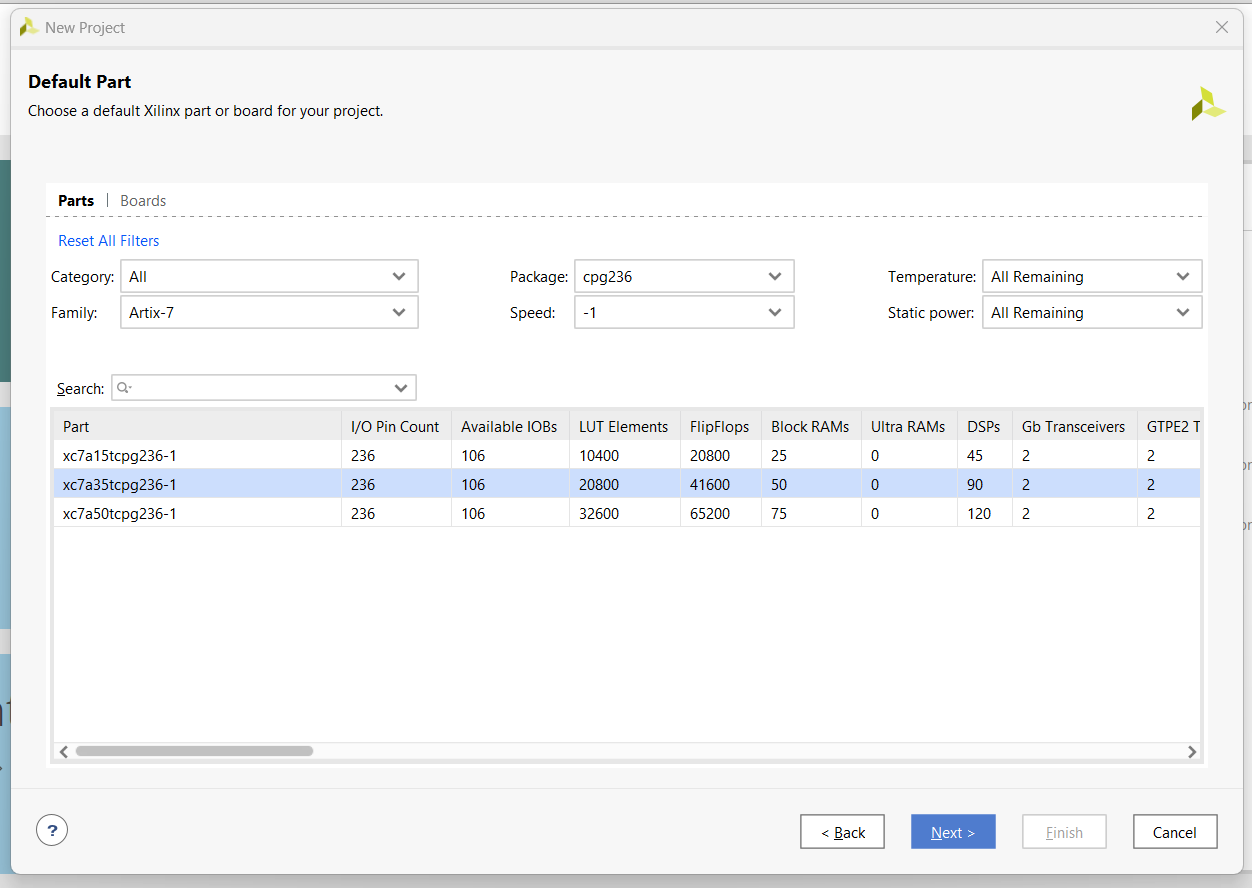
איור ‎4.5 – חלון ה-Add Sources.

* 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.



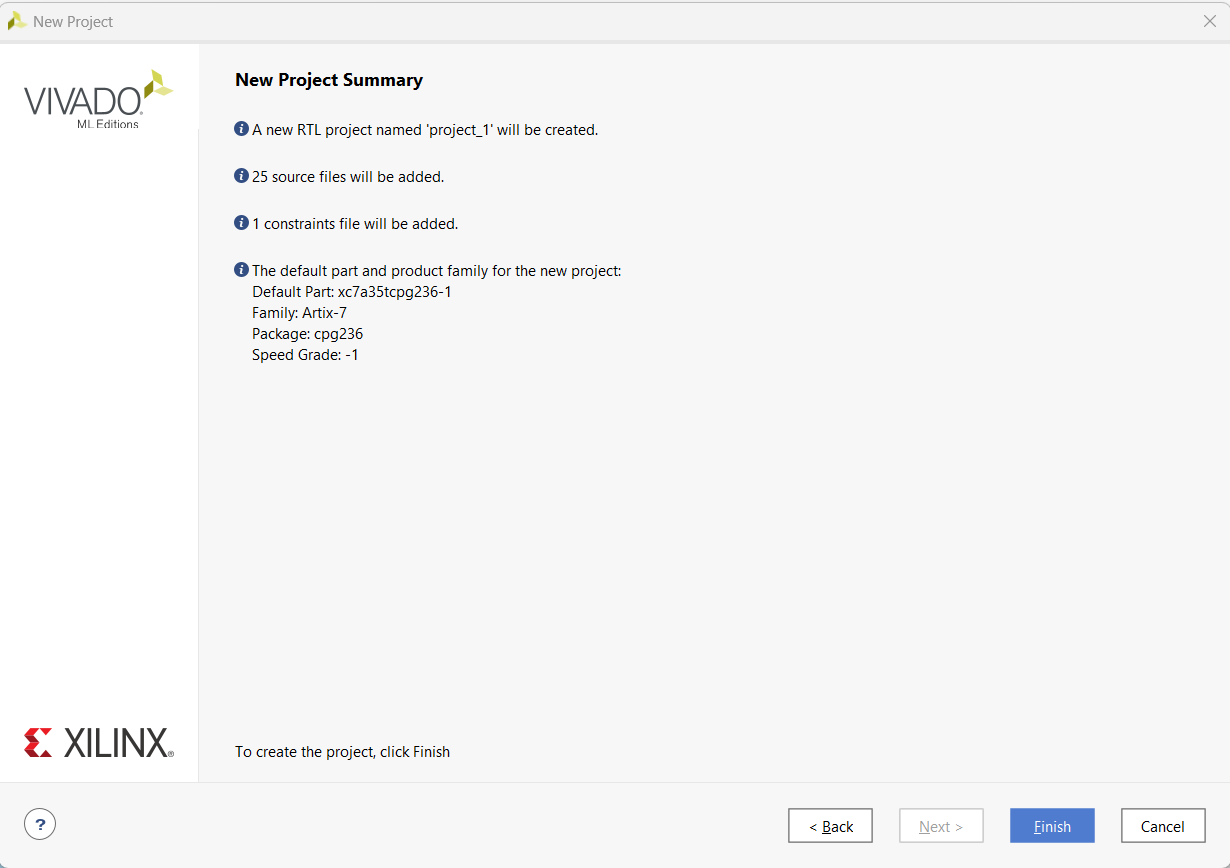
איור ‎4.6 – חלון ה-Add Constraints.

* 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.

תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

התיאור נוצר באופן אוטומטי

איור ‎4.9 – החלון הראשי של ניהול הפרויקט.

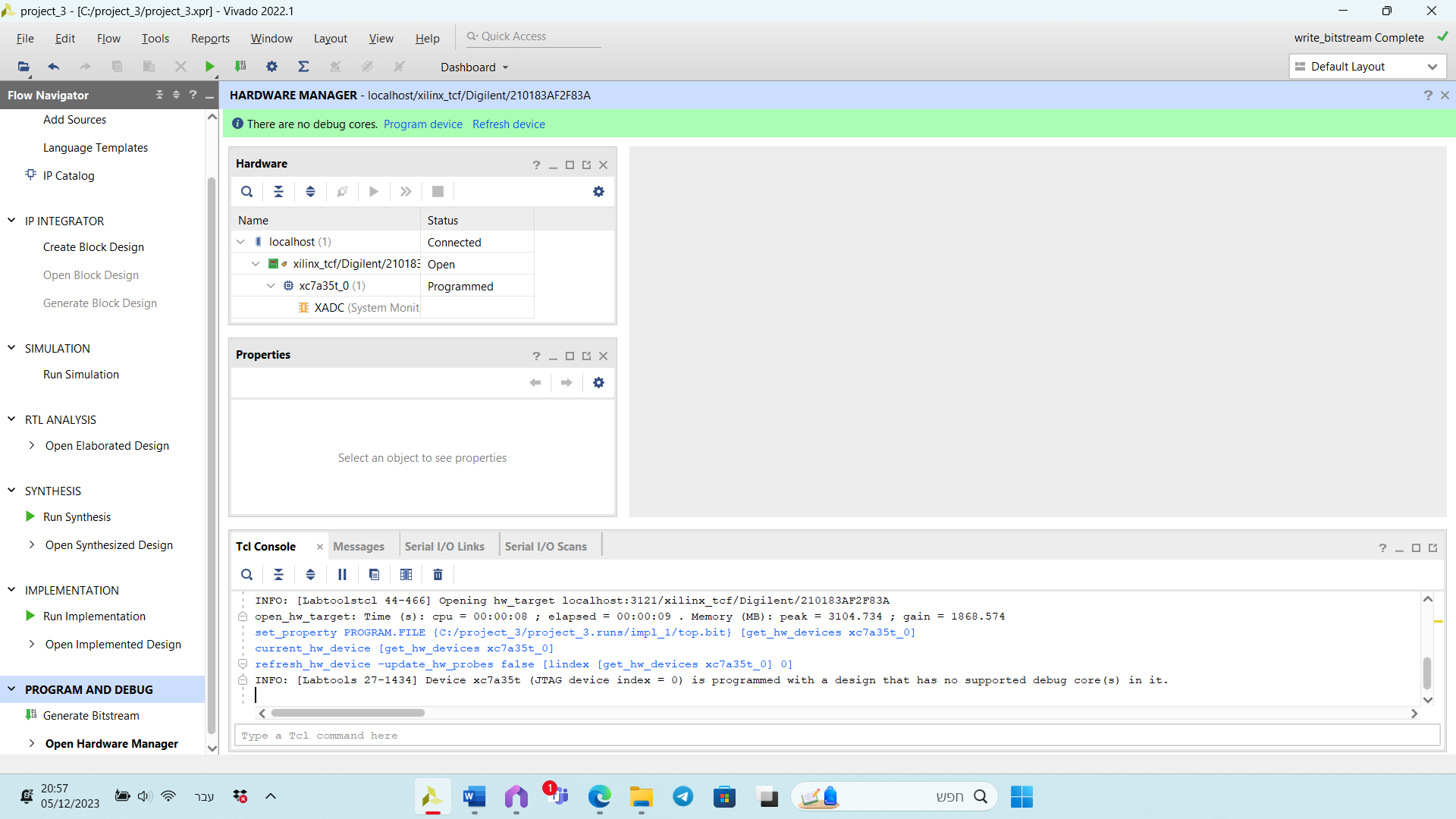
* יש להפעיל תחילה את הסינתזה ע"י לחיצה על Run Synthesis, שנמצא תחת הכותרת SYNTHESIS ב-PROJECT MANAGER.
* אחרי סיום תהליך הסינתזה, יש להפעיל את תהליך ה"מיקום וחיווט" ע"י לחיצה על Run Implementation, שנמצא תחת הכותרת IMPLEMENTATION ב-PROJECT MANAGER.
* אחרי סיום המיקום והחיווט, יש לייצר את קובץ ה- Bitstreamע"י 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 עלOpen 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.

תמונה שמכילה טקסט, צילום מסך, תוכנה, תצוגה

התיאור נוצר באופן אוטומטי

איור ‎4.10 – חלון ה-HARDWARE MANAGER.

* לאחר שכלי התוכנה מצא את הרכיב המיועד להטמעה, יש ללחוץ עלProgram 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.

תמונה שמכילה טקסט, חשמל, צילום מסך, תצוגה

התיאור נוצר באופן אוטומטי

איור ‎4.12 – חלון ה-Program Device להטמעת התכנון.

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

### תהליך הפעלת גיבוב המידע על רכיב ה-FPGA

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

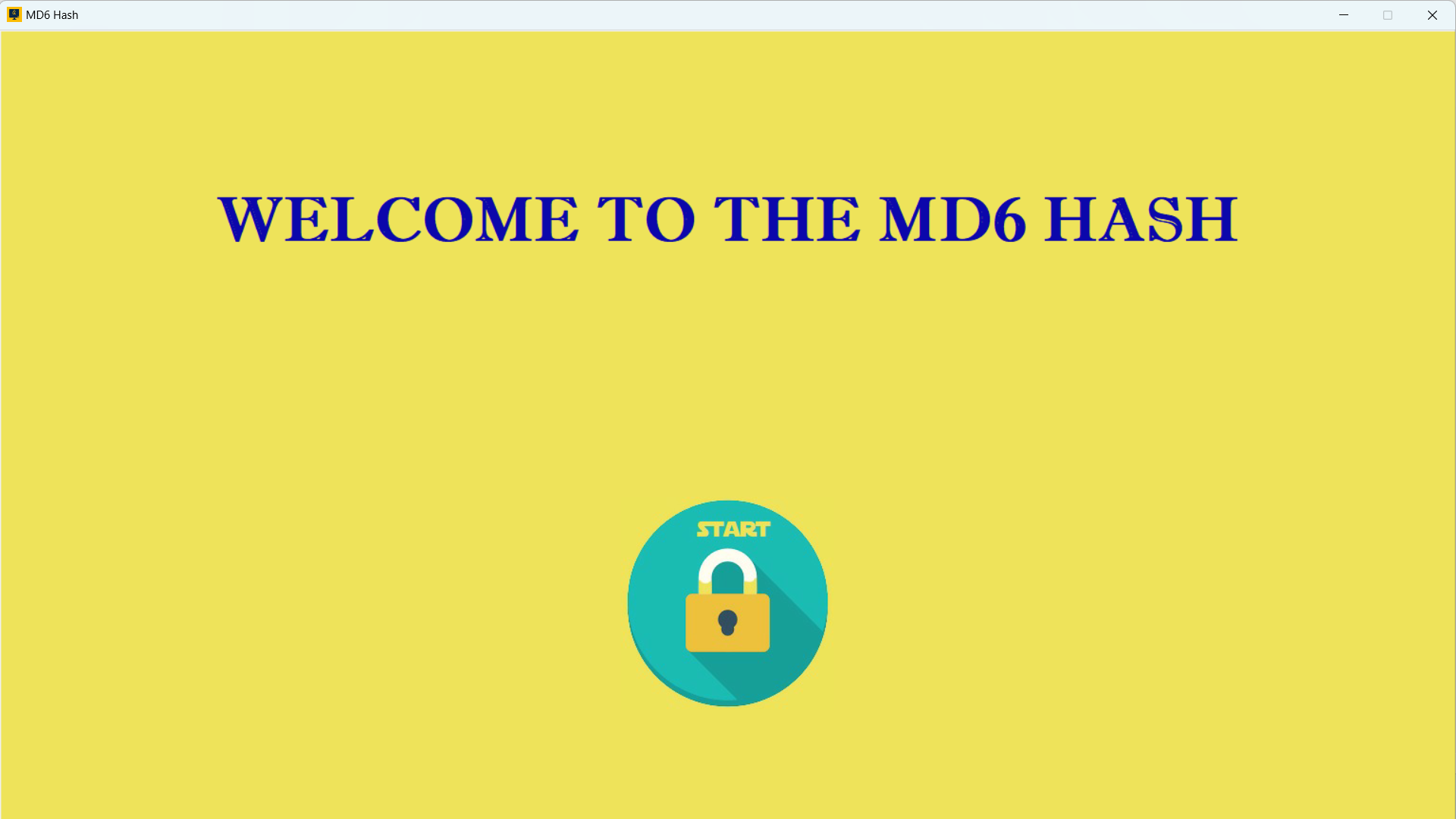
* קודם החלת הגיבוב יש לבדוק את מספר יציאת ה-COM בו מחובר רכיב החומרה. ניתן לבדוק זאת במנהל ההתקנים, כמוצג באיור 4.13 .

תמונה שמכילה טקסט, צילום מסך, תוכנה, מחשב

התיאור נוצר באופן אוטומטי

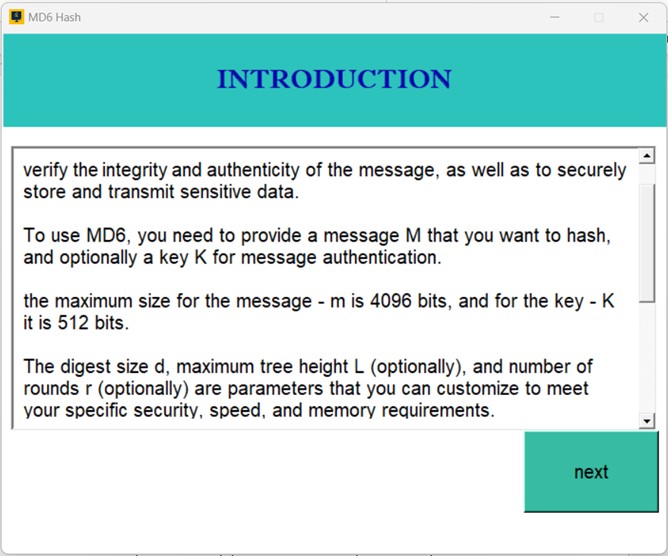
איור ‎4.13 – מציאת מספר ה-COM במנהל ההתקנים.

* יש ללחוץ על לחצן ה-reset (BTNU) ברכיב החומרה כדי לאפס אותו.
* להפעלת תהליך הגיבוב יש לפתוח את תוכנית הגיבוב MD6\_CF.exeאשר 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.



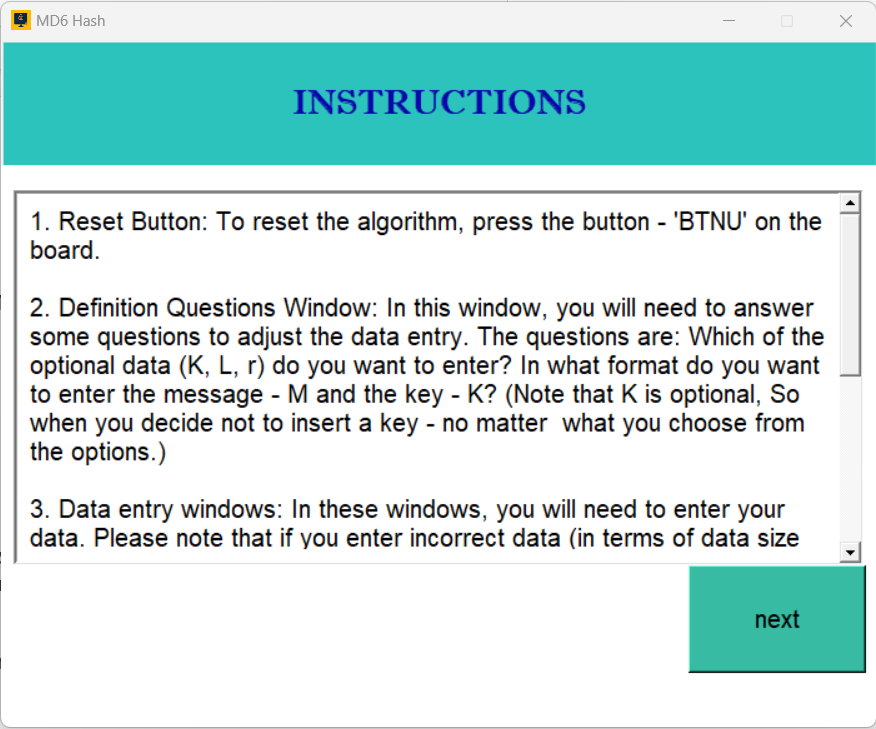
איור ‎4.14 – חלון ההתחלה של תוכנית הגיבוב.

* 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.

תמונה שמכילה טקסט, צילום מסך, גופן, סמל

התיאור נוצר באופן אוטומטי

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.

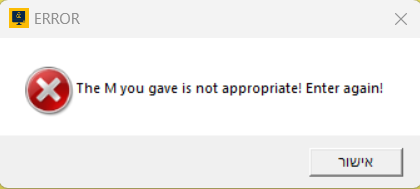
תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטי

איור ‎4.19 – חלון ה-Message (ההודעה).

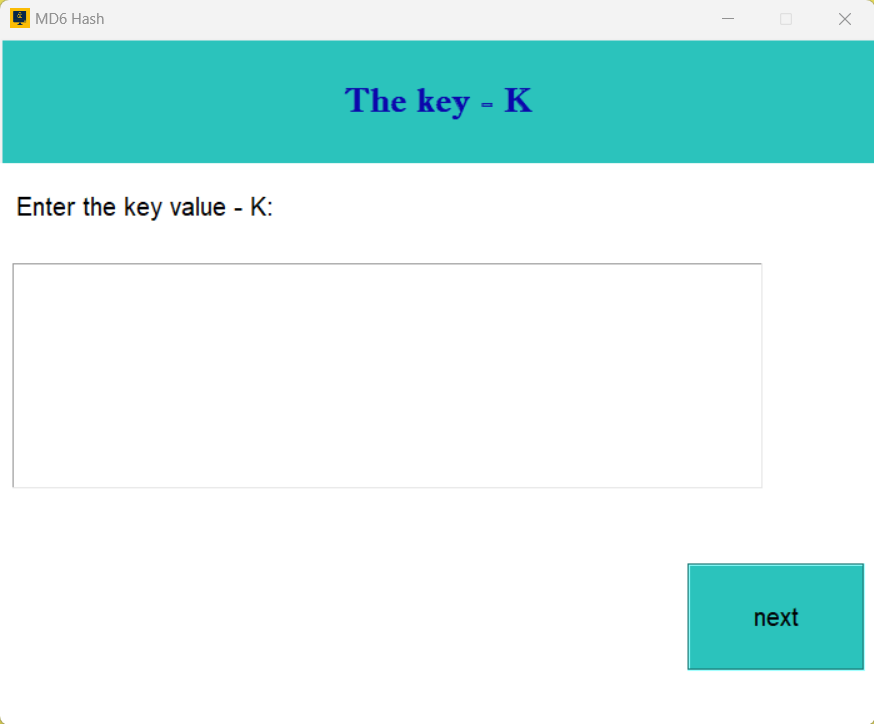
* יש לשים לב – אם ההודעה איננה בגודל הנכון או איננה בפורמט הנכון, התוכנית תשלח הודעת שגיאה כמוצג באיור 4.20.

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.



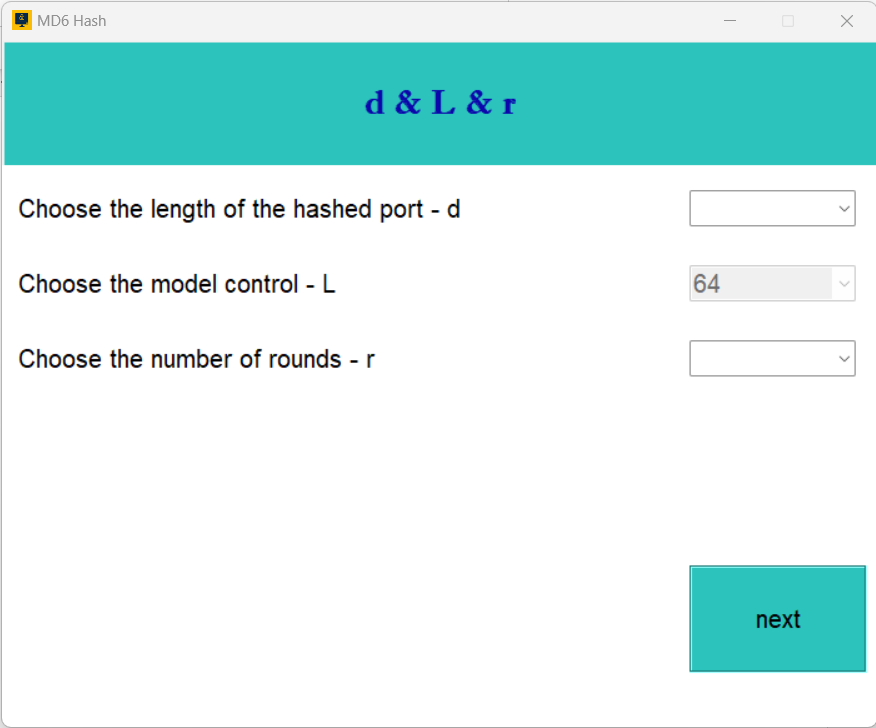
איור ‎4.21 – חלון ה-Key (המפתח) עם תיבת הכנסת המידע.

תמונה שמכילה טקסט, צילום מסך, תוכנה, דף אינטרנט

התיאור נוצר באופן אוטומטי

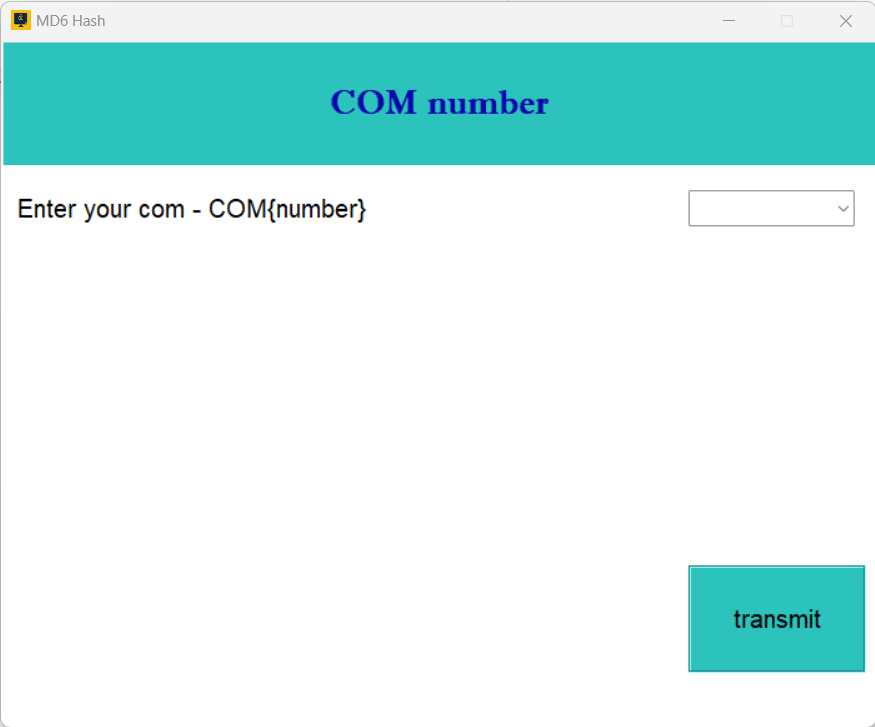
איור ‎4.22 – חלון ה-Key (המפתח) עם ערך ברירת המחדל.

* לאחר לחיצה על לחצן ה-next, יפתח חלון ה-d & L & r, 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.



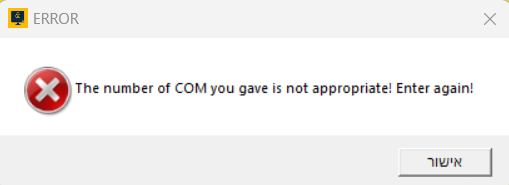
איור ‎4.23 – חלון ה-d & L & r.

* 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.



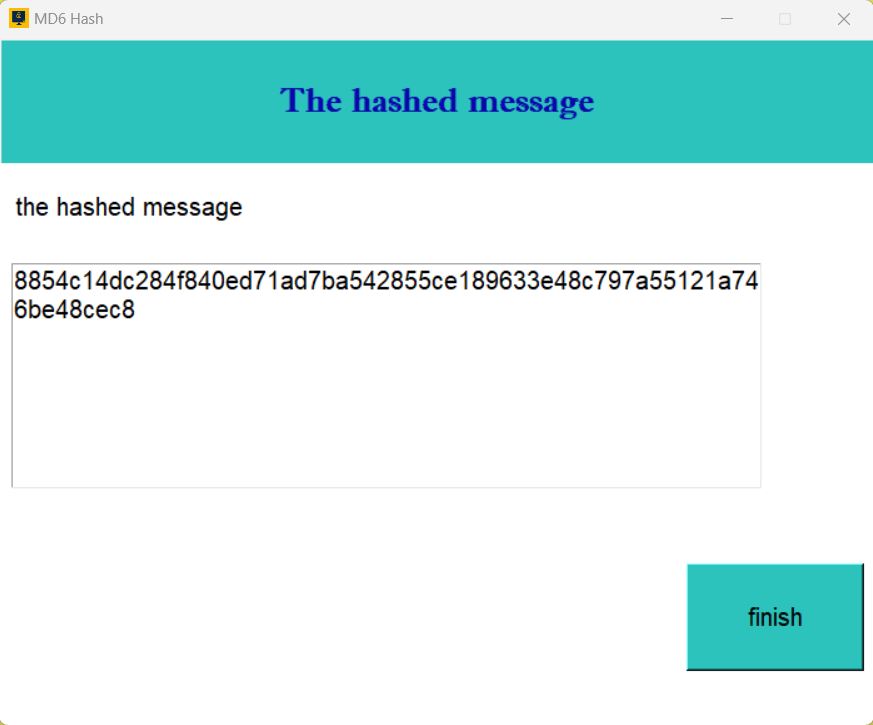
איור ‎4.24 – חלון ה-COM number.

* 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 – חלוןthe hashed message (ההודעה המגובבת).

* לסגירת התוכנית יש ללחוץ על לחצן ה-finish.

## אימות תכנון החומרה בסימולציית ModelSim

### 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:

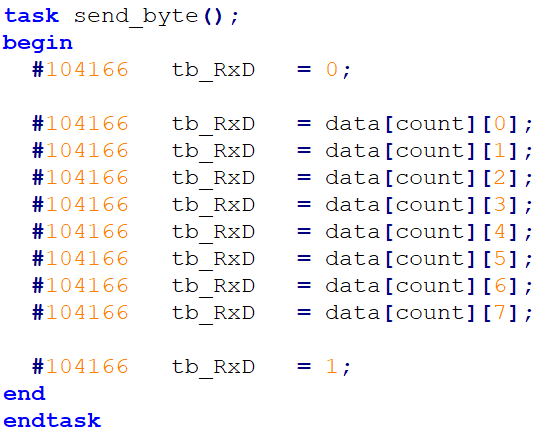
* Test Vector Sim.py:

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 4.27.



איור ‎4.27 – שליחת המידע האקראי דרך ה-RxD.

לאחר סיום הגיבוב, ה-tb מקבל בחזרה את המידע המגובב ושומר אותו במערך מתאים, כמופיע באיור 4.28.

תמונה שמכילה טקסט, צילום מסך, גופן, מספר

התיאור נוצר באופן אוטומטי

איור ‎4.28 – קבלת המידע המגובב דרך ה-TxD.

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

### הפעלת אימות תכנון החומרה בסימולציית ModelSim

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

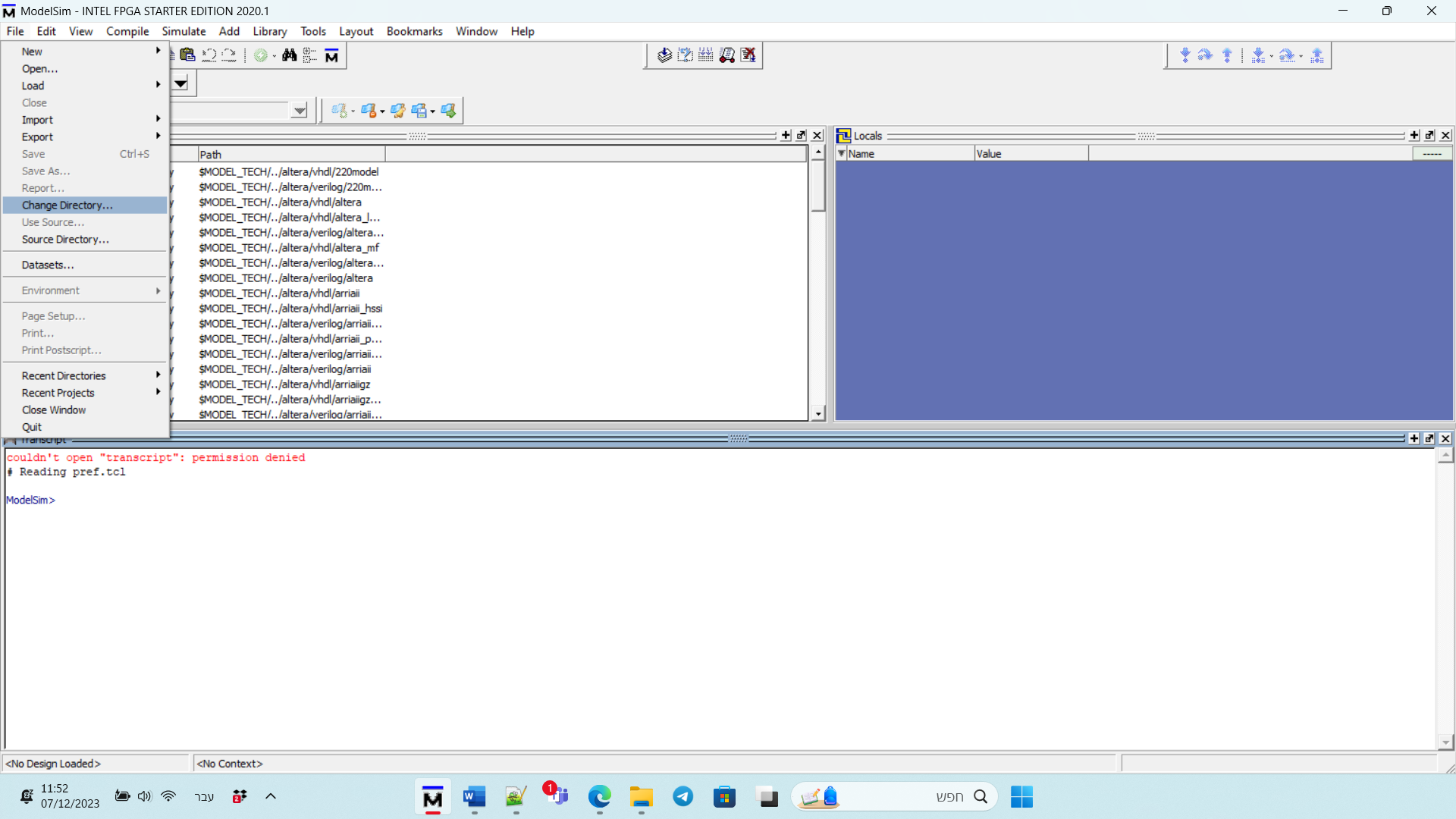
כדי שיהיה ניתן להפעיל את האלגוריתם, יש לוודא שהדברים הבאים נמצאים בהישג יד:

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

קישור לתיקיית קבצי הפרויקט להפעלת האלגוריתם מצורפים [בנספח א](#_נספח_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 4.29, ויש לשנות את מיקום התיקייה לתיקייה בה הועברו את הקבצים.



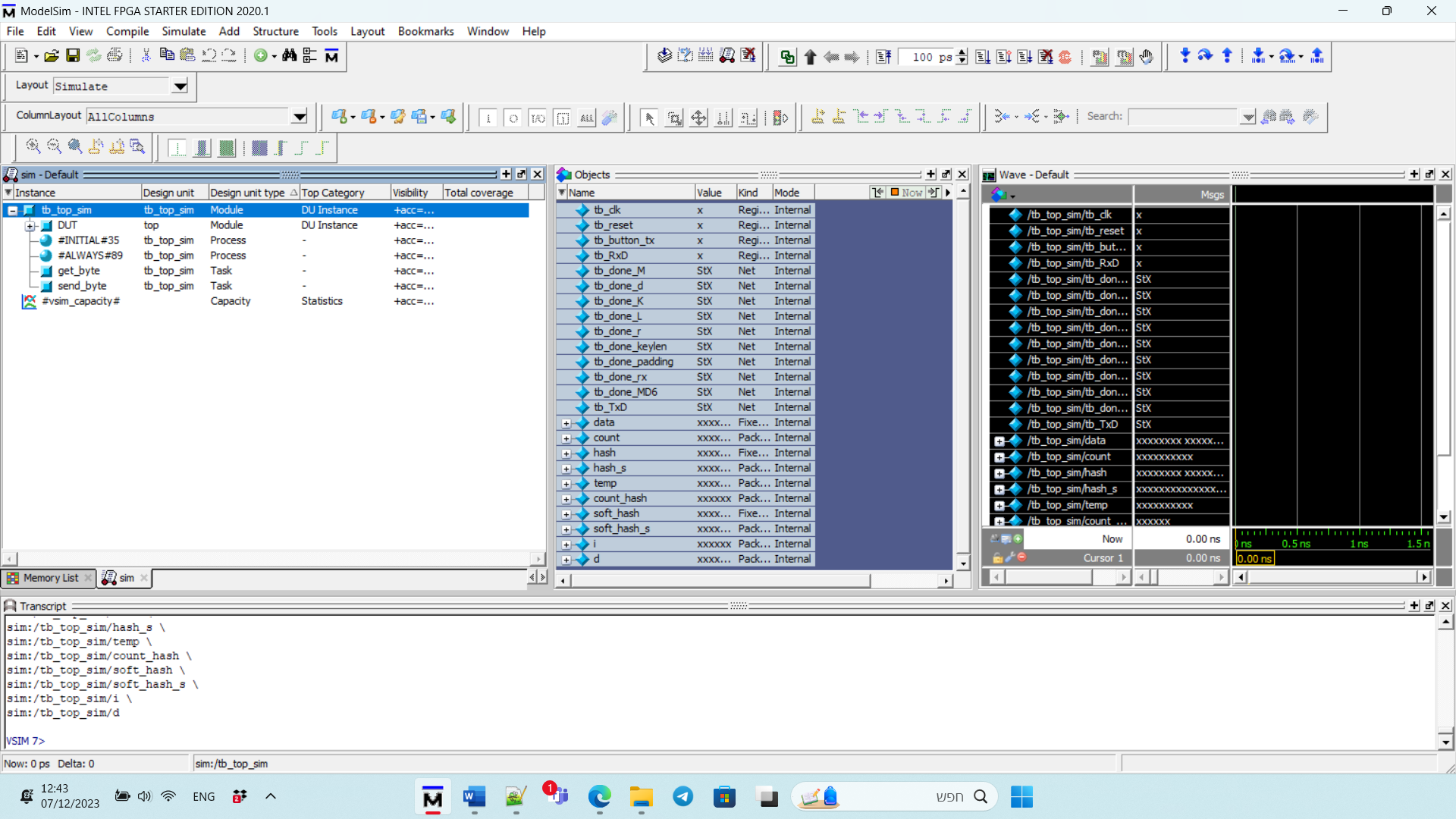
איור ‎4.29 – Change directory בכלי התוכנה ModelSim.

* ב-Transcript ניצור תיקיית עבודה ע"י שימוש בפקודה הבאה:
* לאחר יצירת התיקייה יש לקמפל את קבצי ה-Source ע"י שימוש בפקודה הבאה:

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

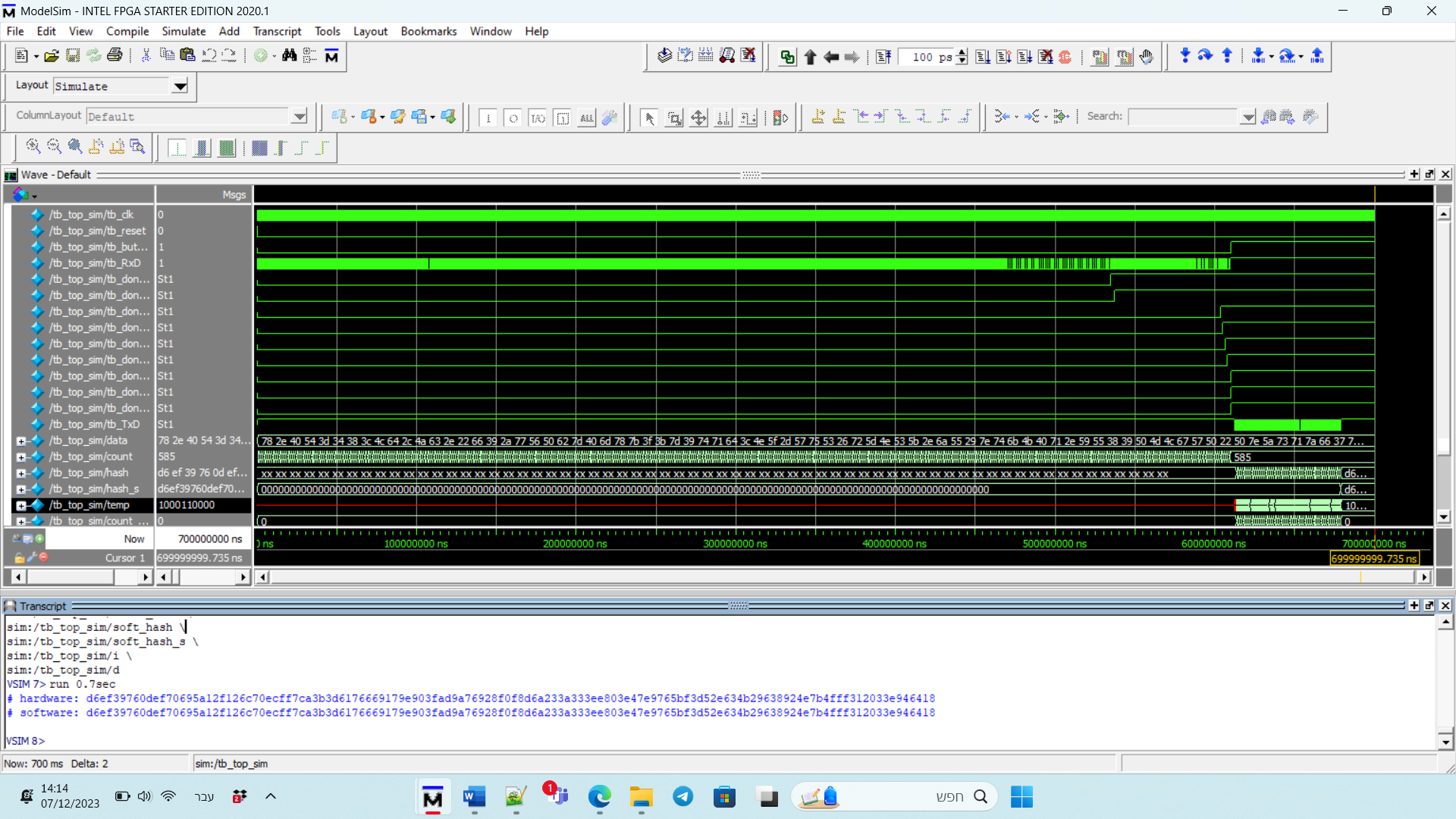
יש לשים לב שאין צורך לקמפל את קבצי ה-mem ואת קובץ ההידור (.h), אלא רק לוודא שיימצאו באותה התיקיה.

* לאחר שהקימפול של קבצי ה-Source עבר בהצלחה, יש לקמפל את קובץ ה-test bench:
* לאחר שקימפול קובץ ה-test bench עבר בהצלחה, יש להפעיל את הסימולציה ע"י הפקודה הבאה:
* לאחר הפעלת פקודת ה-vsim יפתח החלון הראשי של הסימולציה כמוצג באיור 4.30.



איור ‎4.30 – 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:
* *אחרי סיום הרצת הסימולציה יתקבלו במסך ה-*Transcript *את תוצאות הגיבוב בחומרה ובתוכנה אחד אחרי השני כמוצג באיור 4.31.*



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

## אימות תכנון החומרה על ערכת הפיתוח

### 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.

### 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](#_נספח_א_–).

* בתחילה יש להטמיע את התכנון על רכיב החומרה. ניתן למצוא את התהליך המפורט [בפרק 4.1.1](#_הטמעת_אלגוריתם_ה-MD6).
* 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.

תמונה שמכילה צילום מסך, טקסט, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

איור ‎4.32 – הכנסת מספר כניסת ה-COM.

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

תמונה שמכילה טקסט, צילום מסך, תוכנה, מחשב

התיאור נוצר באופן אוטומטי

איור ‎4.33 – 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.

תמונה שמכילה טקסט, צילום מסך, תוכנה, מחשב

התיאור נוצר באופן אוטומטי

איור ‎4.34 – 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.

תמונה שמכילה טקסט, צילום מסך, תוכנה, מחשב

התיאור נוצר באופן אוטומטי

איור ‎4.35 – 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.

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטי

איור ‎4.36 – קובץ csv להשוואה בין התוצאות.

# דיונים

בפרק זה נשווה בין תכנון אלגוריתם MD6 שתוכנן במסגרת הפרויקט, לבין התוצאות בדו"ח האלגוריתם של יוצר האלגוריתם Ronald Linn Rivest [[3]](#c).

יתר על כן, נשווה בין יעילות תכנון האלגוריתם בחומרה ובתוכנה.

## השוואה לספרות

בפרק זה נשווה את תכנון הפרויקט לדו"ח אלגוריתם MD6 של Ronald Linn Rivest [[3]](#c). 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.

תמונה שמכילה טקסט, גופן, קבלה, צילום מסך

התיאור נוצר באופן אוטומטי

איור ‎5.1 – מידע הקלט של הדוגמא הראשונה [[3]](#c).

תוצאת הגיבוב של האלגוריתם לפי הדוגמא הראשונה בדו"ח מופיעות באיור 5.2.

תמונה שמכילה טקסט, גופן, צילום מסך, לבן

התיאור נוצר באופן אוטומטי

איור ‎5.2 – תוצאת הגיבוב של הדוגמא הראשונה [[3]](#c).

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

טבלה 23: 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 |

### 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 5.1.

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

התיאור נוצר באופן אוטומטי

איור ‎5.3 – תוצאת הגיבוב של הדוגמא הראשונה [[3]](#c).

תמונה שמכילה צילום מסך, טקסט, קו, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

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

באיור 5.2 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.](#_הפעלת_אימות_תכנון)

### 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.

תמונה שמכילה טקסט, צילום מסך, תוכנה, דף אינטרנט

התיאור נוצר באופן אוטומטי

איור ‎5.5 – התאמת שאלות ההגדרה לדוגמא הראשונה.

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

תמונה שמכילה טקסט, צילום מסך, תוכנה, דף אינטרנט

התיאור נוצר באופן אוטומטי

איור ‎5.6 – התאמת ההודעה לדוגמא הראשונה.

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

תמונה שמכילה טקסט, צילום מסך, תוכנה, מערכת הפעלה

התיאור נוצר באופן אוטומטי

איור ‎5.7 – התאמת המפתח לדוגמא הראשונה.

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

תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

התיאור נוצר באופן אוטומטי

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

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

תמונה שמכילה טקסט, צילום מסך, גופן

התיאור נוצר באופן אוטומטי

איור ‎5.9 – תוצאת הגיבוב של הדוגמא הראשונה בחומרה.

סרטון המראה את הרצת הדוגמא הראשונה של דו"ח האלגוריתם ניתן למצוא [בנספח א.](#_נספח_א_–)

## 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](#משוואה2):

*כאשר:*Crystal Oscillator *– זמן המחזור של מתנד הקריסטל של רכיב החומרה השווה ל-*10ns, *כמפורט בטבלה 5* [*בפרק 3.1*](#_ערכת_הפיתוח_ורכיב)*.*

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 3.30.*

*לפי נתונים אלו התדר המקסימלי יוצא: .*

*לאחר מציאת התדר המקסימלי יש לחשב את קצב עיבוד המידע (*throughput*) של האלגוריתם לפי* [*משוואה 3*](#משוואה3)*.*

*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:*

*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:*

* חישוב קצב עיבוד המידע בתוכנה:

*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:*

*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.*

# 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.

# מקורות מידע ומאמרים

[1] Follow, G. (2018, November 2). Cryptography introduction. GeeksforGeeks. [Cryptography Introduction - GeeksforGeeks](https://www.geeksforgeeks.org/cryptography-introduction/)

[2] Cryptography Hash functions. (n.d.). Tutorialspoint.com. Retrieved December 16, 2023, from [Cryptography Hash functions (tutorialspoint.com)](https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm)

[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)](https://people.csail.mit.edu/rivest/pubs/RABCx08.pdf)

[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)](https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf)

[5] Merkle, R. C. (n.d.). Ralphmerkle.com. Retrieved December 16, 2023, from [Certified1979.pdf (ralphmerkle.com)](https://www.ralphmerkle.com/papers/Certified1979.pdf)

[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)](https://www.hypr.com/security-encyclopedia/salt)

[7] Secret key. (n.d.). Hypr.com. Retrieved December 16, 2023, from [What is a Secret Key? | Security Encyclopedia (hypr.com)](https://www.hypr.com/security-encyclopedia/secret-key)

[8] Basys 3TM FPGA Board Reference Manual. (2016). Digilent.com. [basys3:basys3\_rm.pdf (digilent.com)](https://digilent.com/reference/_media/basys3:basys3_rm.pdf)

[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)](https://ece353.engr.wisc.edu/serial-interfaces/uart-basics/)

[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)](https://www.researchgate.net/figure/Communication-Between-Alice-and-Bob-intercepted-by-Eve-Here-channel-is-insecure-which_fig1_348427077)

[11] (N.d.). Mouser.Co.Il. Retrieved December 11, 2023, from [102050644.png (600×436) (mouser.co.il)](https://www.mouser.co.il/images/marketingid/2015/img/102050644.png?v=070223.0350)

[12] Customisable design - UART. (n.d.). Tinytapeout.com. Retrieved December 11, 2023, from [Customisable Design - UART :: Documentation in English (tinytapeout.com)](https://tinytapeout.com/digital_design/design_uart/)

# נספח א – קישורים למסמכי וקבצי הפרויקט

* קישור לתיקיית קבצי הפרויקט:

[aviel207/Final\_Project (github.com)](https://github.com/aviel207/Final_Project)

* קישור לקובץ ה- README של תיקיית הפרויקט:

[Final\_Project/README.md at main · aviel207/Final\_Project (github.com)](https://github.com/aviel207/Final_Project/blob/main/README.md)

* קישור לדו"ח הפרויקט המלא:

[Final\_Project/Documents/Project reports/FPGA Implementation of MD6 Hash.pdf at main · aviel207/Final\_Project (github.com)](https://github.com/aviel207/Final_Project/blob/main/Documents/Project%20reports/FPGA%20Implementation%20of%20MD6%20Hash.pdf)

* קישור 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)](https://github.com/aviel207/Final_Project/blob/main/Documents/Project%20reports/FPGA%20Implementation%20of%20MD6%20Hash%20-%20RTL%20Code%20and%20Simulation.pdf)

* 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)](https://github.com/aviel207/Final_Project/blob/main/Documents/Project%20reports/FPGA%20Implementation%20of%20MD6%20Hash%20-%20Implementation%20and%20Executable%20Code.pdf)

* קישור לסרטון של הפעלת אלגוריתם MD6:

[Full Demonstration Of Implementing The CF MD6 On FPGA (youtube.com)](https://www.youtube.com/watch?v=ZC4aacu0pbw)

# 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 Xilinx-AMD 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)](https://www.openhw.eu/2023-results-gallery)