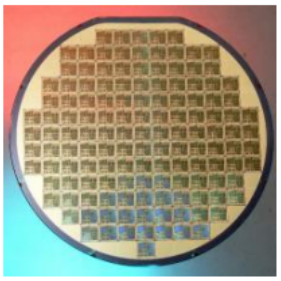
הטכניון - מכון טכנולוגי לישראל

מרכז מחקר למערכות VLSI

המעבדה ל-VLSI

הפקולטה להנדסת חשמל ומחשבים



מאיץ חומרתי לתיקון שגיאות בזכרונות מבוססי ד.נ.א

המבצעים

דור אביב
212778179

שגיא פריעד
212669394

מנחה
ד"ר עמית ברמן

תוכן עניינים

רשימת תרשימים

רשימת קיצורים

הסבר	פירוש מילולי	קיצור

DNA/ד.נ.א	DeoxyriboNucleic Acid	מולקולת אחסון מידע ביולוגית
CRISPR	clustered Regularly Interspaced Short Palindromic Repeats	טכניקת עריכה מבוקרת של תוכן סלילי ד.נ.א
PCR	Polymerase Chain Reaction	טכניקת קריאה של תוכן סלילי ד.נ.א על בסיס שכפול
IDS	Insertion Deletion Substitution	משפחת שגיאות מידע נפוצות בעולמות הביו-כימיה
VT	Varshamov-Tenengolts	קוד זוגיות המקבע את סכום מיקומי ביט "1" באמצעות ביטי זוגיות בחזקות של 2
SISO	Soft-Input-Soft-Output	הסקת המסקנות אינה בינארית אלא ערך רציף המבטא וודאות. הרצף מבוסס פענוח על סמך מספר רב של כניסות.
LLR	Log Likelihood Ratio	מודל סבירות סטטיסטי מבוסס לוגריתם בין $\pm\infty$ כאשר 0 מעיד על שוויון
BER	Bit-Error-Rate	יחס השגיאה של רצף מידע. מוגדר להיות היחס בין מספר הביטים הלא נכונים חלקי מספר הביטים הנכונים.

תקציר

בפרויקט זה מומשה מערכת לתיקון שגיאות ברצפי מידע המותאמת לאחסון בד.נ.א, המבוססת על מאמר קיים בתוכנה וכן כמאיץ חומרתי. במהלך הפרויקט

היה צורך לענות לאתגרים השונים של כל אופן פיתוח, הנוגעים למגבלות החישוב של המערכת. בפרויקט הוצגו תוצאות הסימולציות התוכנתיות וכן נכונות הפעולה של המאיץ החומרתי החומרה.

מבוא

זיכרונות ד.נ.א

ד.נ.א היא מולקולה פולימרית אשר משמשת מנגנונים ביולוגיים לצרכי שמירת זיכרון, ומשמשת את התא החי בתור "ספר המתכונים" להרכבת חלבוני התא.
עם גילוי טכנולוגיית הCRISPR בשנת 2012, קיבלו בני האדם יכולת שליטה מלאה במידע הנמשר במולקולה, הן לצרכי קריאה והן לצרכי כתיבה, כך שהן יכולות לשמש בתוך מרכיבי זיכרון עבור עולמות המחשוב והמידע. בהשוואה לטכניקות זיכרון דיגיטליות אחרות, מולקולות הד.נ.א הינן בעלות עמידות לזמן ארוך בסדר גודל מטכנולוגיות הזיכרון העמידות ביותר, ובעלת צפיפות גבוהה בסדר גודל מהמגבלות הפיזיקליות על טכנולוגיות הזיכרון המודרניות. עם זאת, מידת השגיאה וזמן הגישה בד.נ.א כה גבוהים שאין היום היתכנות מעשית לשימוש הזכרונות אלו לצרכים מסחריים. בכדי להתגבר על מגבלות אלו יש צורך בפיתוח טכנולוגיות יעילות יותר לצד אוטומציה של מנגנוני הגישה לזיכרון.
בפרויקט זה נתמקד בבעיית השגיאות המרובות ונממש מאמר הנוגע לאלגוריתם תיקון שגיאות המותאם ל-DNA בתוכנה וכן בחומרה.

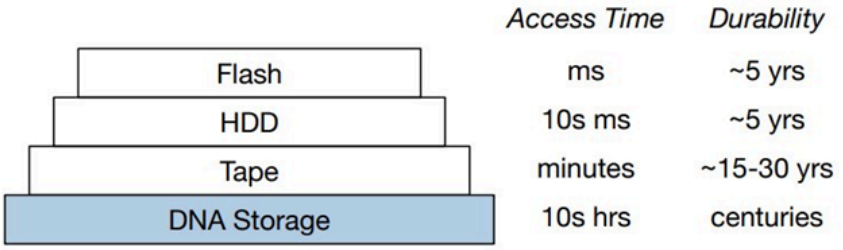


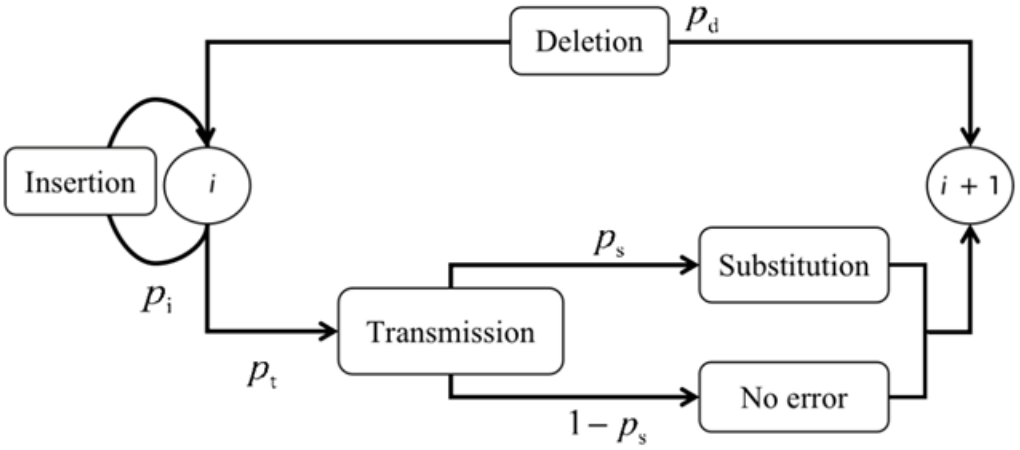
Figure 2. DNA storage as the bottom level of the storage hierarchy

תרשים 1 - בהשוואה לזכרונות דיגיטליים מוכרים, זכרונות ד.נ.א הם בעלי עמידות יוצאת דופן כמו זמן גישה ארוך במיוחד, אופי המתאים לשימוש בתחתית היררכיית הזיכרון עבור זכרונות גדולים מאוד בשימוש שאינו תדיר

מולקולת הד.נ.א מורכבת מסליל הכורך שני גדילים זה מול זה. על כל גדיל מצויים גרעין אשר נקשר בקשר כימי חלש לגרעין בעל קשר משלים המצוי על גבי הגדיל השני. כל גרעין יכול להיות אחד מארבעה סוגים, בשני זוגות תואמים: אדנין (A)

ותימין (T), לצד גואנין (G) וציטוזין (C). שנקדד לסט הביטים 00, 01, 10, ו-11 בהתאמה, ללא תלות במידע בין שני הרצפים.

הטכניקה המקובלת לקריאת סלילי ד.נ.א הינה PCR בה נלקח מיכל המכיל מספר סלילים שונים ומוכנס אליו אנזים הגורם להכפלת מספר הסלילים מסוג מסוים. לאחר ביצוע הפעולה מספר פעמים נדגמים מספר סלילים מתוך המיכל, בהנחה שאחוז הסלילים שמקורם בסליל הרצוי מתוך כלל הסלילים בדגימה גבוהה מאוד. עם זאת, במהלך תהליך השכפול מתרחשות שגיאות אשר לצרכי זיכרון בינארי מאופיינות בתור שגיאות IDS כלומר שגיאות בהן מוכנס ביט עודף, מוסר ביט רצוי או מוחלף ביט רצוי. מאחר ושגיאות אלו מתרחשות במהלך סריקה פיזית של הגדיל על ידי האנזים תהליך זה קורה גרעין לאחר גרעין, לכן ניתן לאפיין אותו לפי המודל ההסתברותי המוצג בתרשים:



תרשים 2 - תיאור איטריציית הרכבת ביט עבור שגיאות IDS. בכל שלב בהתאם להסתברות המתוארת מתבצע שימוש בביט במקום הi.

עבור כל ביט ברצף - בהסתברות p_i מתקיימת הכנסה של ביט רנדומי, בהסתברות p_d נמחק הביט במיקום הנוכחי. אחרת, (בהסתברות $p_t = 1 - p_i - p_d$) משדרים את הביט, ובהסתברות p_s מתהפך ערכו.

לפיכך בעת קריאת הזיכרון נקבל מספר דגימות שונות, כאשר רובן גרסאות משובשות של המידע המקורי ומיעוטן מידע שאינו המידע הרצוי.

קידוד VT לרצפים

שגיאות IDS הינן מובחנות במיוחד עבור מידע המקודד לפי קוד VT (הומצאו על ידי Varshamov-Tenengolts). קידוד זה מתקין ביטי זוגיות במיקומים הבינאריים של

המידע כדי לשמר בין כל הביטים את ה**סינדרום** (a) שלהם,כך שכל הרצפים התקניים מוגדרים באופן הבא

$$VT_{a,m}(n) = \left\{ \boldsymbol{v} \in \{0,1\}^n : \sum_{i=1}^{n} i v_i \equiv a \pmod{m} \right\} \quad (1)$$

כלומר כל המילים התקניות בעלות אותו סינדרום אשר מוגדר כשארית סכום מיקומי כל ביטי ה"1". מאחר והמיקומים הבינאריים נשמרים לביטי זוגיות, קל לקודד כל מידע לקוד VT תקני. קודים אלו נוחים במיוחד לטיפול בשגיאות IDS, מאחר והשפעת השגיאה על הסינדרום מעידה על מיקום השגיאה. נציין כי m הינו 2n+1 (אורך המאפשר אבחון שגיאה), כאשר n הוא אורך הקוד לאחר קידוד.

SISO + פענוח רצף

אלגוריתם פענוח הרצפים אשר מאפשר שחזור נלקח מתוך המאמר [1]. הפענוח מסתמך על אלגוריתם BCJR (הומצא על ידי Bahl - Cocke - Jelinek - Raviv) שביסודו חישוב הסתברות לביט מסוים כתלות בערכים לוגיים המבטאים את התלות הסטטיסטית של אותו הביט בביטים סמוכים לו ובמידע המשובש, ובפרט תלות Forward המייצגת תלות בביטים קודמים ותלות Backwards המייצגת תלות בביטים הבאים ברצף.

לצרכי ביצוע האלגוריתם, מוגדרים פרמטרי מעקב עבור כל אינדקס t ברצף המתקבל (שנסמן r_t, בעל **אורך N**). אלה הם הפרמטר s עבור **סינדרום** (סכום מיקומי ביטי ה"1" כפול המיקום שלהם ברצף) עד כה בתהליך הסריקה קדימה ופרמטר d עבור **סחיפה** (מוגדר להיות מספר ההכנסות פחות מספר המחיקות).

עבור המעבר מהביט ה-t-1 לביט t מוגדרים הפרמטרים המתאימים s', d' אשר עומדים בתנאי:

$$s' + t v_t \equiv s \pmod{2n+1} \qquad d' \le d+1, \quad (2)$$

כאשר v_t הוא הרצף המתוקן (בעל **אורך n**). ניתן להגדיר באמצעותם את הסתברות המעבר γ:

$$\gamma_t(s',d',s,d) = Pr[\boldsymbol{r}_{d'+t}^{d+t}, s, d|s',d'] \\ = Pr[s|s']Pr[\boldsymbol{r}_{d'+t}^{d+t}, d|s',d',s],$$

where

$$Pr[s|s'] = \begin{cases} 1 & s'+jb \equiv s \pmod{2n+1}; \\ 0 & \text{otherwise}, \end{cases}$$

and for $l \geq 0$,

$$Pr[\boldsymbol{r}_{d'+t}^{d+t}, d|s',d',s] \\ = \begin{cases} p_{\mathrm{d}} & d = d'-1; \\ \left(\frac{1}{2}p_{\mathrm{i}}\right)^l (p_{\mathrm{t}}p_{\mathrm{s}} + \frac{1}{2}p_{\mathrm{i}}p_{\mathrm{d}}) & \boldsymbol{r}_{d+t} \neq b, d = d'+l \\ \left(\frac{1}{2}p_{\mathrm{i}}\right)^l (p_{\mathrm{t}}(1-p_{\mathrm{s}}) + \frac{1}{2}p_{\mathrm{i}}p_{\mathrm{d}}) & \boldsymbol{r}_{d+t} = b, d = d'+l \\ 0 & \text{otherwise}. \end{cases} \tag{3}$$

באמצעות הסתברות המעבר ניתן להגדיר תלות forward לביטים קודמים ותלות backward לביטים הבאים, α ו-β בהתאמה:

$$\alpha_t(s,d) = \sum_{(s',s)\in\Sigma_t,(d',d)\in\mathcal{D}_t} \gamma_t(s',d',s,d)\alpha_{t-1}(s',d'); \\ \beta_t(s',d') = \sum_{(s',s)\in\Sigma_{t+1},(d',d)\in\mathcal{D}_{t+1}} \gamma_{t+1}(s',d',s,d)\beta_{t+1}(s,d) \tag{4}$$

<u>הערה</u>: α_0 מאותחל להיות 1 עבור s,d=0 ו- β_n מאותחל להיות 1 עבור $s = a, d = N - n$, כדי לעגן בקצות הרקורסיה את המגבלות הרצויות אלגוריתם זה מתבצע באופן רקורסיבי לקבלת α, β המלאים. באמצעותם ניתן לחשב את ההסתברות לביט מסוים ברצף המקורי להיות 0/1 על סמך סכום ההסתברות עבור כל המעברים האפשריים.

$$Pr[\boldsymbol{r}, \mathcal{S}|v_t = b] = \sum_{(d',d)\in\mathcal{D}_t,(s',s)\in\Sigma_t^b} \alpha_{t-1}(s',d')\gamma_t(s',d',s,d)\beta_t(s,d) \tag{5}$$

כאשר הסכימה מתבצעת לכל זוגות הסינדרום והסחיפה העונים לתנאי במשוואה (2).

בפענוח קשיח הסתברויות אלו הינן מספיקות לשיערוך השחזור. בעבור פענוח ממספר מקורות ניתן להשתמש בטכניקות מסוג SISO, אשר כפי שצוין מתאימה לבעיית תיקון השגיאות ב ד.נ.א. על ידי סכימה של ה- Log-Likelihood עבור כל אחד מהרצפים הנדגמים ניתן לקבל מדד משוקלל (בהנחת התפלגות 0/1 אחידה):

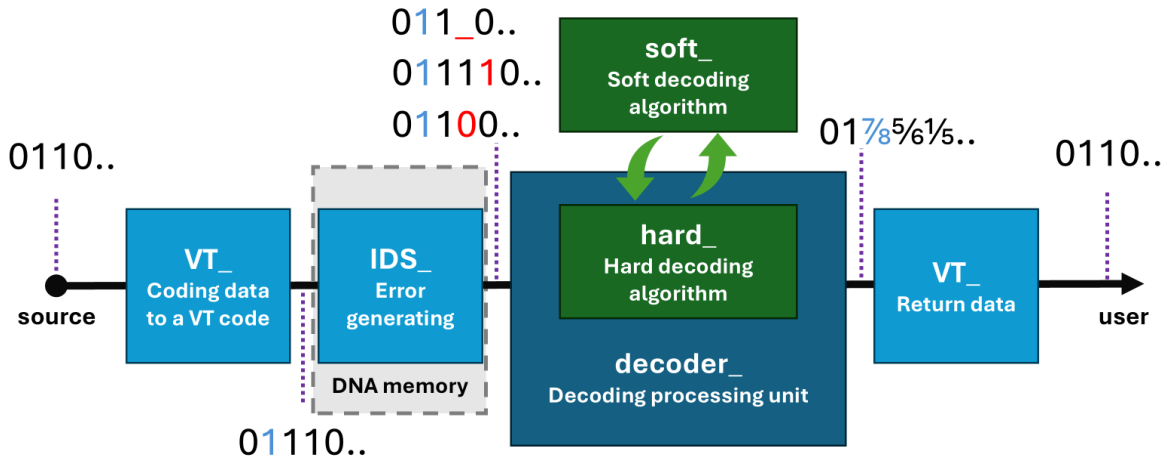
$$L(v_t) = \sum_{i=1}^{c} \log \frac{Pr[\boldsymbol{r}_i | v_t = 0]}{Pr[\boldsymbol{r}_i | v_t = 1]} \quad (6)$$

מתוך הגדרת Log-Likelihood עבור $L(v_t) > 0$ ההנחה היא שהביט הינו 0, ועבור $L(v_t) < 0$ ההנחה היא שהוא 1.

תוכן הפרויקט

תכנון לוגי

ברצוננו לבנות מאיץ ליישום האלגוריתם כחלק ממערכת זיכרון ד.נ.א אוטומטית, לשם כך נמדל את מנגנון המידע באופן הבא:



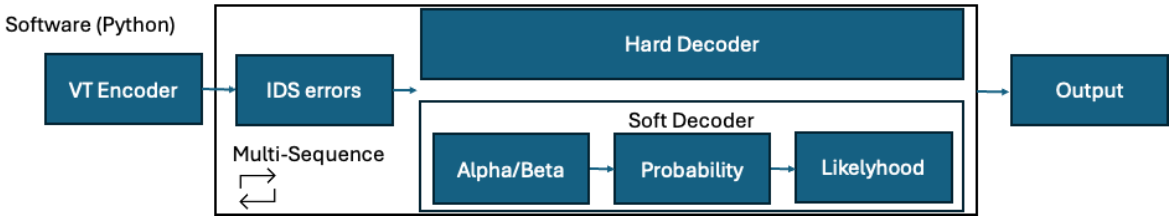
תרשים 3 - תיאור "נתיב הנתונים" במערכת, המידע מקודד לקודי VT, לאחר מכן מחולל שגיאות IDS ממוחשב מדמה את תהליך קריאת הזיכרון באמצעות PCR (שכפול ויצירת שגיאות), רצפי המידע המשובש מוזנים למפענח אשר מכיל יחידת עיבוד ניתנת להחלפה בין אלגוריתם פענוח קשיח והאלגוריתם הרך, ולאחר מכן מקודד חזרה לקוד בינאי וקוד הVT מפוענח.

תחילה נקודד את המידע לקוד VT, לאחר מכן נדמה את זיכרון הד.נ.א על ידי יצירת שגיאות IDS במידע באמצעות מחולל ייעודי, בסופו נקבל מספר שיבושים שונים של אותו קוד המקור. את המידע המשובש נזין למפענח אשר יבצע את האלגוריתם. בכדי למדל SISO אנחנו נדרשים לביצוע אותו אלגוריתם מספר רב של פעמים (כמספר העותקים המשובשים). לשם כך ומתוך רצון לייעל שימוש בחומרה בחרנו למדל את המפענח בתור **מיקרו-בקר** כאשר העותקים המשובשים השונים נשמרים בזיכרון, ובאמצעות מכונת מצבים בכל שלב נשלח עותק אל יחידת עיבוד מרכזית אשר מבצעת את האלגוריתם ומזינה את תוצר הLLR בזיכרון, וכן תשלוט על כתיבה לזיכרון והודעה על סיום החישוב עבור קבוצת העותקים שנטענה. בנוסף על

הבקר לבצע הטלה של תוצר ממוצע הLLR לערך בינארי. לאחר מכן המידע הבינארי מתורגם חזרה מקוד הVT המשוחזר למידע המקורי שקודד.

תכן תוכנתי

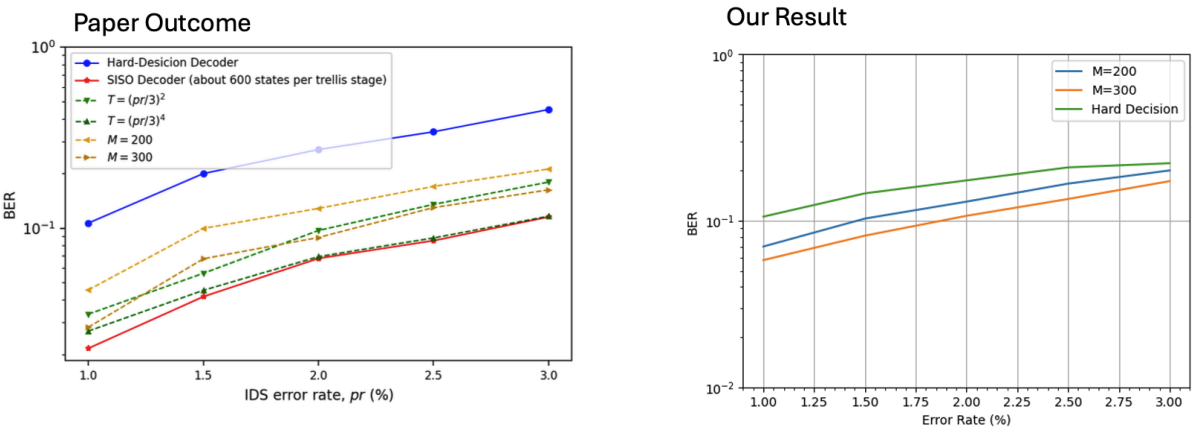
על מנת לוודא את נכונות מימוש האלגוריתם, נבנתה סימולציה תוכנתית של המערכת. המערכת נכתבה כקובץ Jupyter Notebook. למען מימוש הסימולציה, המערכת חולקה למספר בלוקים אשר רצים באופן סדרתי.



תרשים 4 - ארכיטקטורת התוכנה לחישוב הקידוד+פענוח של רצפי מידע. לאחר קידוד המידע, נוצרים מלאכותית מספר רצפים שונים עם שגיאות. עבור כל שגיאה מחשבים את ה- Log-Likelihood, והסכימה הסופית מייצרת את השחזור.

על מנת לענות על מגבלות זמן המחשוב של המערכת (גם לאחר הייעול שייצוין, הסימולציות חושבו במשך 8 ו-16 שעות בהתאמה), בוצעה התאמה לאלגוריתם בהשראה ממאמר [1]. שיטה זאת היא M-Reduced Decoding Strategy, בה מחשבים את M המצבים s, d בעלי ערכי α המשמעותיים ביותר בכל אינדקס. יתר הערכים "נזרקים" כך שנדרש לנרמל את M המצבים ל-1. ערכי β נקבעים על סמך M ערכי s, d שנקבעו על פי α. שיטה זו מורידה במקצת את דיוק המודלים והשחזור, שכן ייתכן ו"נאבד" הסתברות לאחד הביטים כך ש- Log-Likelihood יישאף לאינסוף (במודל התוכנתי זה נלקח בחשבון).

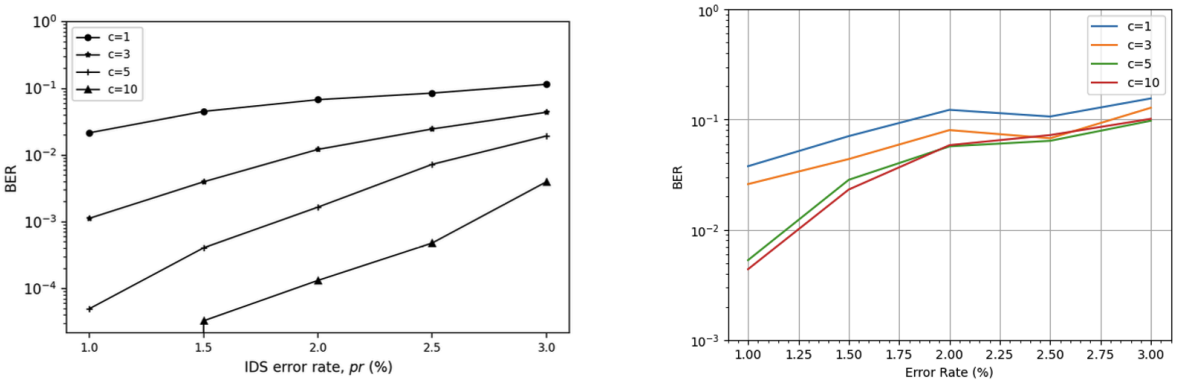
היה נסיון לשחזור תוצאות המאמר בסימולציה. בשלב הראשון חושב ה- BER כתלות באחוז השגיאה $p_r = 3p_i = 3p_d = 3p_s$ הכולל. לאחר הרצה של 100 סימולציות עבור רצף באורך מקורי של $n = 68$ התקבלה התוצאה הבאה:



תרשים 5 - משמאל: גרף השגיאה המופיע במאמר. מימין: גרף השגיאה כפי שחושב בסימולציה שלנו. נקודות הדגימה הם (1, 1.5, 2, 2.5, 3). המדידות עבור T נוגעות לשיטה אחרת לצמצום יחידות החישוב שלא נבדקה בסימולציה שלנו.

תוצאות המדידה דומות וכך גם מגמת השינוי ביחס לשגיאה. עם זאת, ישנו היסט מסוים בכל הדגימות הרלוונטיות. ההערכה בנוגע לשגיאה נובעת ממספר דגימות קטן באופן יחסי (בהרצות שונות ישנם הפרשים בתוצאות, אך עקב מגבלות החישוב לא היה ניתן להגדיל בסדר גודל את מספר הנסיונות). בנוסף, במאמר לא צוין כיצד חושב ה -BER: ביחס לאות המקורי, ביחס לאות המשוחזר, או רק ביחס לביטים ששונו. עם זאת, ניתן לראות כי ככל שמגבילים את המודל יותר דיוק השחזור יורד.

בנוסף, ביצענו סימולציה של ה BER ביחס למספר הרצפים שאיתם מבצעים את פענוח ה SISO (משתנה c).

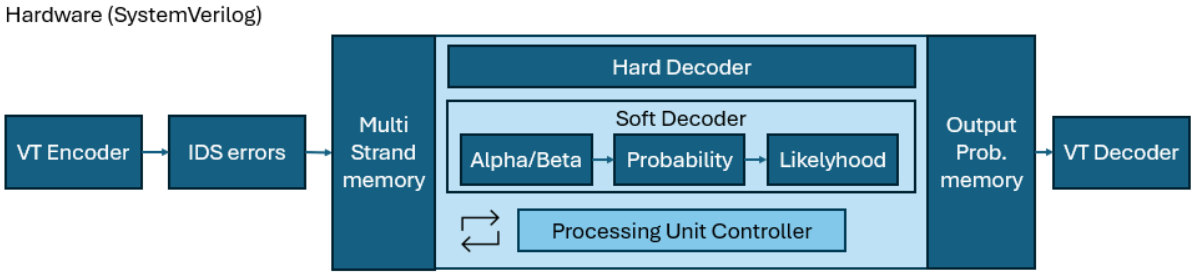


תרשים 6 - משמאל: גרף השגיאה המופיע במאמר. מימין: גרף השגיאה כפי שחושב בסימולציה שלנו. נקודות הדגימה הם (1, 1.5, 2, 2.5, 3).

מגמת המדידות תואמת לתוצאות המאמר אך תוצאות הסימולציה לא מדויקות. מקור השגיאות הוא גם מהסיבות המצוינות לעיל, אך בעיקר מכך שהגרף שהתקבל בסימולציה שלנו חושב עבור מערכת M-Reduced עם M=200, וזה עקב מגבלות המחשוב (נדרשו 16 שעות חישוב על מנת להפיק 100 מדידות). ניתן לראות שככל שמסתמכים על יותר רצפי מידע בכדי להסיק את הפענוח דיוק המודל עולה.

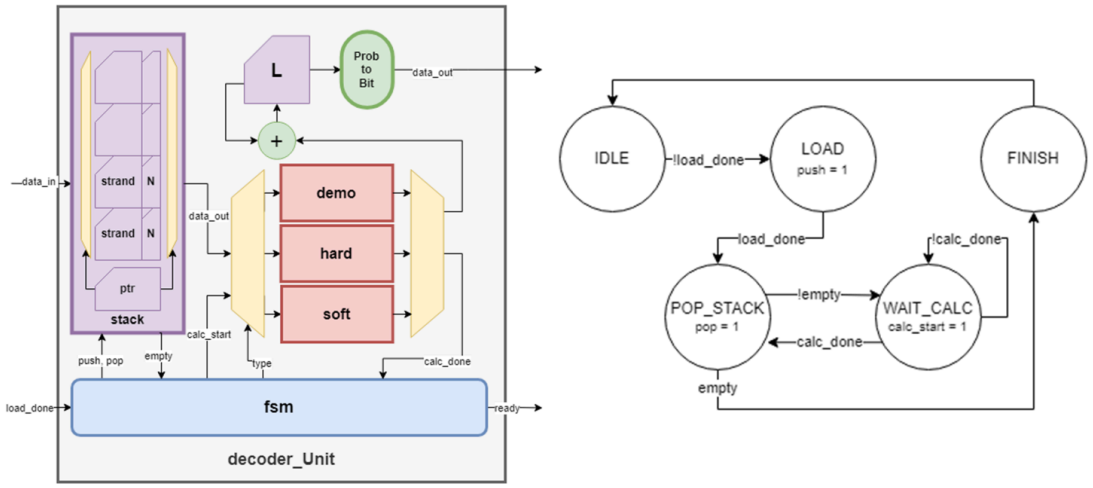
תכן חומרתי

כחלק ממטרות הפרויקט, ישנו צורך ליישם את התכנון הלוגי לחומרה מעשית. לשם כך נבנו מודלים מתאימים בSystemVerilog, מתוך מטרה לתרגם את התכנון הלוגי שבוצע בקוד לתכנון מעשי בחומרה:



תרשים 7 - ארכיטקטורת המערכת החומרתית, החל מייצור וקידוד הרצפים השונים בעלי השגיאות, הפענוח (המסוגל להיות "קשיח" או "רך") וקבלת רצף ה log-likelihoods שאיתם משחזרים. הפענוח מתבצע בתוך מעבד בעל רכיבי זיכרון לכניסות ויציאות האלגוריתם, לצד בקר המנהל את שלבי ביצוע החישוב.

מודל זה מסתמך חלקית על התכנון הלוגי ועל עיבוד המידע התוכנתי כדי לבצע את פעילות המאיץ. לצורכי ניהול זמנים תקין נקבעו אינדיקטורים לארך הדרך וכל תת-מקטע שיכול להיות קומבינטורי נקבע ככזה. במפענח נבנה בקר שליטה ומסלולי מידע אפשריים שונים כך שכתלות בבחירת המצב המפענח יוכל לתפקד בתור מפענח "רך", מפענח "קשה" או מפענח דמה לצרכי בדיקת תקינות המעבד. המעבד תוכנן לעבוד עם מכונת המצבים לפי התיאור הבא:



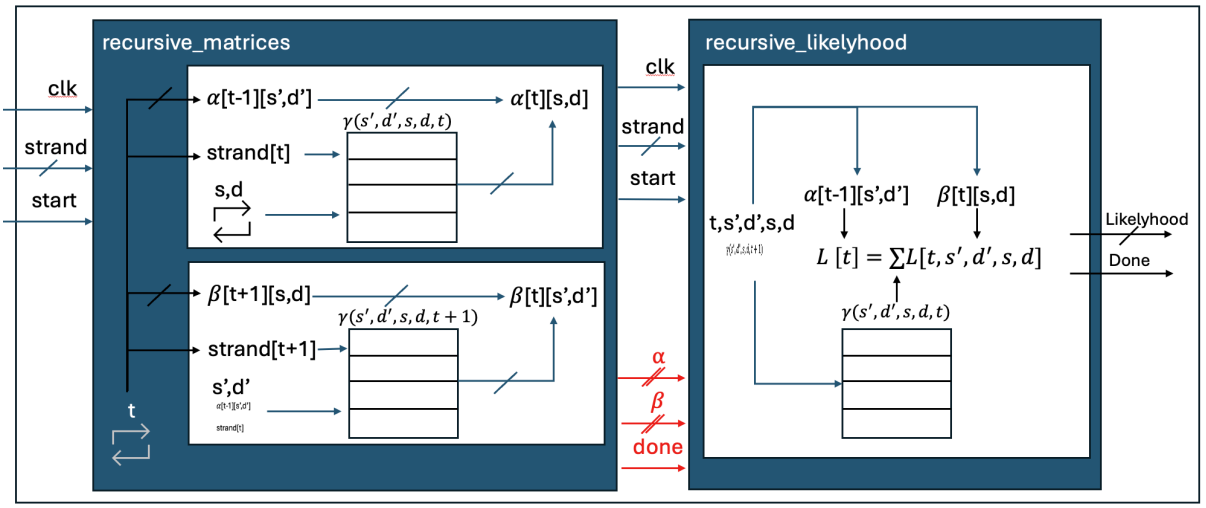
תרשים 8 - תיאור מבנה המעבד, למעבד זיכרון ערמה ברוחב קבוע השומר את אורך המידע, זיכרון מתעדכן לשמירת הסבירות, מקודד הסתברויות לרצף בינארי, רכיבי אלגוריתם סגורים נשלטי מכונת מצבים ובורר לבחירה ביניהם. מכונת המצבים מנהלת את טעינת המידע לזיכרון, מזינה מידע מהזיכרון ליחידת החישוב וממתינה לסיום החישוב, וממשיכה את המעבר על הזיכרון עד לסיומו.

מאחר והשיבושים המקורות גורמים לשינוי בגודל המידע המשובש, היה צורך להתמודד עם שאלת ניהול הזיכרון. נבחר להשתמש בזיכרון בגודל קבוע לכל רצף

משובש, כאשר גודלו גדול בהרבה מהגודל הממוצע של רצפים משובשים. שיטה זו מאפשרת ניהול זיכרון מהיר מאוד ומקלה על שילוב נוח למשתמש של המאיץ לתוך מנגנון זכרונות ד.נ.א אוטומטי. ואמנם, לבחירה זו מספר מגרעות משמעותיות - כל תחום זיכרון קבוע מחזיק איתו שיירים ריקים של זיכרון שניתן היה לנצל ביעילות עם ניהול זיכרון דינמי, והיא חותכת מקרי קיצון בהם ריבוי במקרי הכנסה גורר מידע משובש מעבר לגודל הקבוע. אנחנו חושבים שבעבור צרכי המאיץ יתרונות המהירות של זיכרון בגדלים קבועים עדיפים על חסרונותיו, ומומש זיכרון מחסנית. כמו כן לצד כל מידע משובש נשמר גודלו האמיתי של המידע המשובש מתוך גודל הזיכרון הקבוע כדי להימנע מקריאה של זיכרון מחוץ לתחום החוקי. שיטה זו מאפשרת הזנה מהירה של רצף משובש לתוך ערוץ בגודל קבוע בקצב של רצף כל מחזור שעון.

שיקול נוסף הינו מהירות האלגוריתם אשר הינה O(1) עבור אלגוריתם "קשה" ו O(n^2) עבור אלגוריתם "רך". בשקלול הפרמטרים נמצאה הפשרה של בקר עם יחידת חישוב בודדת וזיכרון בגודל קבוע להיות המתאימה ביותר לצרכינו, מאחר והיא חסכונית, פשוטה, ורסטילית וסקיילבילית לשינויים בפרמטרי החישוב (אורך המידע המוצפן, מספר עותקי הסלילים הזמינים לכל מידע).

ביצוע האלגוריתם הרך תוכנן באמצעות שני מודלים עם תוצרים דו מימדיים המייצגים את קומבניציות [s,d] האפשריות, כאשר המודל הראשון מייצר באופן מחזורי את תלות הα, β באיטרציות במחשבות את התלות קדימה ואחורה לצד שימוש בתלות מעבר γ באופן קומבינטורי, והבלוק השני משתמש במטריצות לחישוב הlikelihood



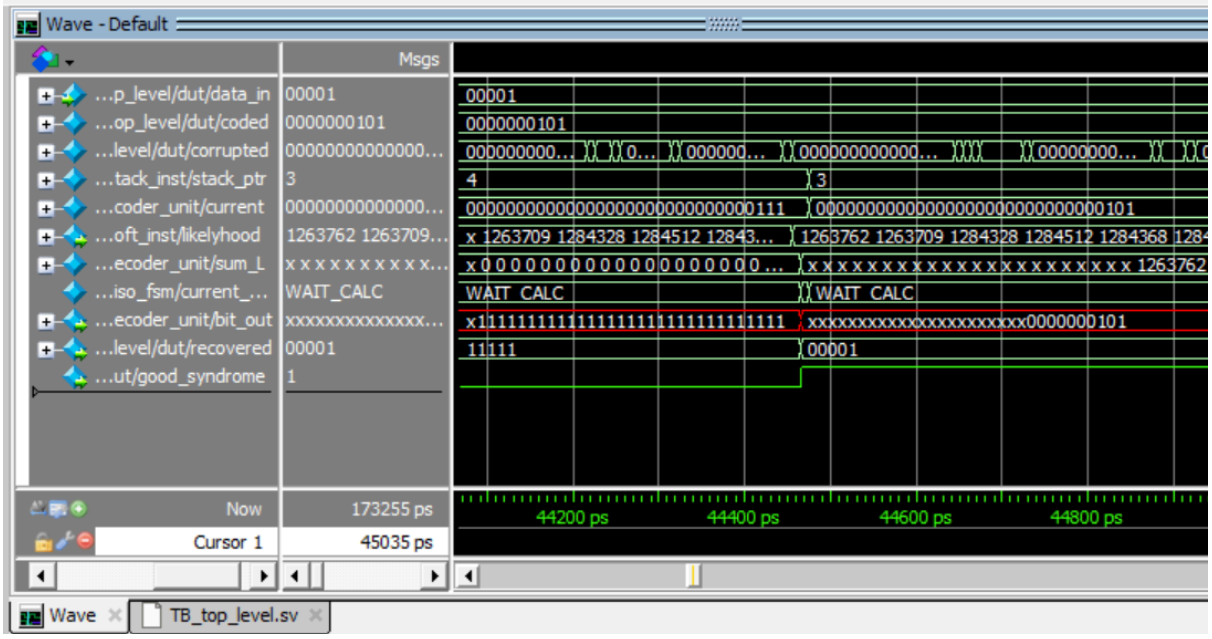
תרשים 9 - המבנה החומרתי של הפענוח ה-"רך" ביחידת החישוב. ישנם 2 שלבים סדרתיים, הנוגעים לחישוב המטריצות α, β, וחישוב ה likelihood של כל ביט ברצף.

מודל זה משמש בגודל קבוע לצרכי החישוב, כאשר כל שכבת T הינה ברוחב 2n+1 במימד הS מאחר וחישוב בסינדרום מבוסס %(2n+1), ורוחב n + WIDTH למימד ה-D כאשר n נובע מהסחיפה המינימלית האפשרית של מחיקה מוחלטת וWIDTH הינו גודל קבוע (כמפורט לעיל) אשר גדול מN, גודל המידע המשובש.בכל איטרציה מיוצרת חומרה באמצעות generate שמייצרת מטריצת מעבר γ מתאימה. לאחר ייצור ה α, β מבוצע חישוב הLikelyhood באמצעות המידע ומטריצות מעבר.

החומרה מומשה לפי ההיררכיה הבאה (סימון ° למודל אשר נבדק בתור מודל ראשי בTestBench):

Top_level° - המערכת המשולבת
- VT_encode° - קידוד כניסה לקוד
- IDS_generator° - מחולל שגיאות בכניסה מקודדת
    - Xorshift_prng° - מחולל מספרים רנדומיים
- Decoder_unit° - מעבד פענוח שגיאות
    - Decoder_stack° - זיכרון מחסנית של מידע משובש
    - Decoder_fsm - מכונת מצבים של המעבד
    - Soft_decoder° - חישוב אלגוריתם רך
        - Soft_gamma° - יחידת עזר לחישוב מטריצות מעבר
        - soft_Recursive_matrices - ניהול חישוב תלות
        - Soft_forward° - חישוב איטרציית תלות קדימה
        - Soft_backward° - חישוב איטרציית תלות אחורה
        - soft_Recursive_likelyhood - ניהול חישוב הסתברות
        - Soft_likelyhood - חישוב איטרציית הסתברות
    - hard _container - מתאם חישוב קשיח למעבד
        - Hard_decoder° - חישוב אלגוריתם קשיח
- VT_decode° - פענוח יציאה מקוד

דוגמא לסימולציית בדיקת TB עבור top_level:



תרשים 10 - סימולציית TestBench של המודל הראשי top_level. מידע 0001 מוזן מ-data_in, מקודד לקוד VT באמצעות VT_encode אל coded לערך 0000000101, המידע המקודד משובש על ידי IDS_generator אל corrupted, מוזן באמצעות אינדיקטור לתוך מערך הזיכרון decoder_stack אשר מוציא את הרצף הנוכחי current ואורכו, במקרה זה המידע משובש להיות 0000000111. יחידת החישוב soft_decoder מפענח את השגיאה ומוציא likelyhood לוגריתמי אשר נסכם לתוך sum_L (במקרה זה הוא הראשון בזיכרון לכן sum עדיין מאופס) ומתורגם לbit_out לפי סימנו, נבחין כי השגיאה אותרה ותוקנה, וvt_decoder משיב את המידע המקורי. כמו כן נבדק שהקוד המשוחזר עומד בתנאי הסינדרום.

במהלך כתיבת החומרה נדרשה התחשבות בקו ההנדסי הכללי לצד התמודדות עם אתגרים מקומיים. כדי לתאם בין המקטעים השונים נעשה שימוש באינדיקטורים סינכרוניים ונדרש ניהול זמנים בין המודלים כדי להבטיח מעבר טוב ומדויק של האינדיקטורים. מספר אתגרים ומוסכמות מתוארים בנספחים.

סיכום

ניתן לאפיין את הפרויקט כהצלחה - הצלחנו לממש מערכת תיאורטית כן בתוכנה וכן בחומרה כפי שהתבקש, ולבנות תשתית לשיפורים וייעולים לפענוח המערכת. המערכת עצמה מהווה Proof of Concept של השימוש באלגוריתם המוצע על מנת לבצע תיקון שגיאות ברצפי DNA.

במהלך ביצוע הפרויקט היו מספר אתגרים אשר נגעו לכל אחד מהשלבים בתכנון. אתגר משמעותי ביותר היה בהירות המאמר שעליו התבסס הפרויקט : חסר פירוט עבור חלקים שלמים ממימוש המאמר, ישנם החלפות לא מוסברות באינדקסים ואף חלקים שאינם נכונים (במיוחד באלגוריתמיקה במימוש המפענח הקשיח). היה צורך בפיתוח ופרשנות עצמאית של חלק מן האלגורתמים, אשר ייתכן והובילו להבדלים בין הסימולציה לתוצאות המאמר (כפי שמופיע בחלק התוכנטי).

אתגר משמעותי נוסף נבע מההבדל בין תכנון של מערכות תוכנה מול חומרה - השיקול של זמן מחשוב (בעיקר בתוכנה) אל מול גודל וסיבוכיות המערכת (בעיקר בחומרה). לפיכך, השיטה שבה מימשנו את המערכת בתוכנה ובחומרה היו שונות לחלוטין - בתכנה השתמשנו בשיטת M-reduced והסתמכנו על מיון האיברים בכדי לסנן אותם. בחומרה מצד אחד קשה יותר לממש מיון שכזה אך מצד שני איננו מוגבלים עד כדי כך בזמן הריצה ולכן פרסנו את כל המצבים והסתמכנו על מערכת גדולה יותר.
לבסוף - עקב האבסטרקציה הנמוכה של המימוש החומרתי היה צורך בבנייה של מודולים ופונקציות המאפשרות לממש את האלגוריתם. ניתן לראות בנספחים פירוט של האופן בו מימשנו מספרים שאינם שלמים, מערכת סמי-אקראית לשגיאות, וקירוב לפונקצית הלוגריתם שבה משתמשים ב- log likelyhood. הרחבה עליהם מופיעה בנספחים.

ניתן לאפיין מספר כיווני שיפור משמעותיים. שיפור מרכזי עוסק בהקטנת מספר המידע/זכרון הנדרשים למימוש החומרתי. מכיוון שלא ידענו לצפות מהו מספר המצבים, נלקח "מרווח ביטחון" של כמות המידע המוקצה שמוגבל בגודל הרצף המקסימלי במערכת (שרירותית הוא 32). אפיון יותר מורכב של המערכת, ואף שימוש ב M-reduced כפי שנעשה בתוכנה יורידו את גודל החומרה (לשם כך יהיה צורך לממש מערכת מיון חומרתית). שיפור נוסף נוגע לשימוש חוזר באותם רכיבים שכבר קיימים - למשל חישוב מראש של מצבי gamma הרלוונטיים ושימוש חוזר בהם, במיוחד במעבר מחישוב מטריצות לחישוב likelihood כפי שמופיע בתרשים המפענח הרך.

נספחים

מקורות

[1] Z. Yan, G. Qu and H. Wu, "A Novel Soft-In Soft-Out Decoding Algorithm for VT Codes on Multiple Received DNA Strands," 2023 IEEE International Symposium on Information Theory (ISIT), Taipei, Taiwan, 2023, pp. 838-843, doi: 10.1109/ISIT54713.2023.10206446.

[2] Sousa, R. (n.d.). *From BCJR to Turbo Codes*. Universidade do Porto. Available at https://paginas.fe.up.pt/~sam/textos/From%20BCJR%20to%20turbo.pdf.

[3] Marsaglia, G. (2003). *Xorshift RNGs*. The Florida State University.

מוסכמת ייצוג מספרי ואריתמטיקה

במהלך החישוב נדרש שימוש בשברים ובמספרים מעורבים. כדי להקל על מימוש חומרתי יעיל וחסכוני בחרנו לצרכי סינתזה להשתמש במודל נקודה מקובעת (fixed point) על פני שימוש בנקודה צפה (floating point), לפי פורמט Q

בלי הגבלת הכלליות בהינתן מספר בינארי ברוחב 32 ביטים נגדיר נקודה לאורכו המגדירה את הנקודה הבינארית, למשל Q16.16 כך ש16 הביטים הLSB מייצגים שבר בינארי ו16 הביטים הMSB מייצגים מספר בינארי שלם.בפרט הביט ה0 (LSB) מייצג 2^{-16} , הביט ה15 מייצג 0.5, הביט ה16 מייצג 1 וכן הלאה. באופן שקול המספר בפורמט Q16.16 שקול למספר הבינארי בצורה שלמה אשר מוזז ימינה 16 פעמים או מחולק ב2^{16}, כלומר תחת סימון a למספר בפורמט Q16.16 וA למספר אשר נקרא מפורמט Q16.16 כאשר מניחים שהוא מספר בינארי שלם (Q32.0) יתקיים $a \cdot 2^{16} = A$.

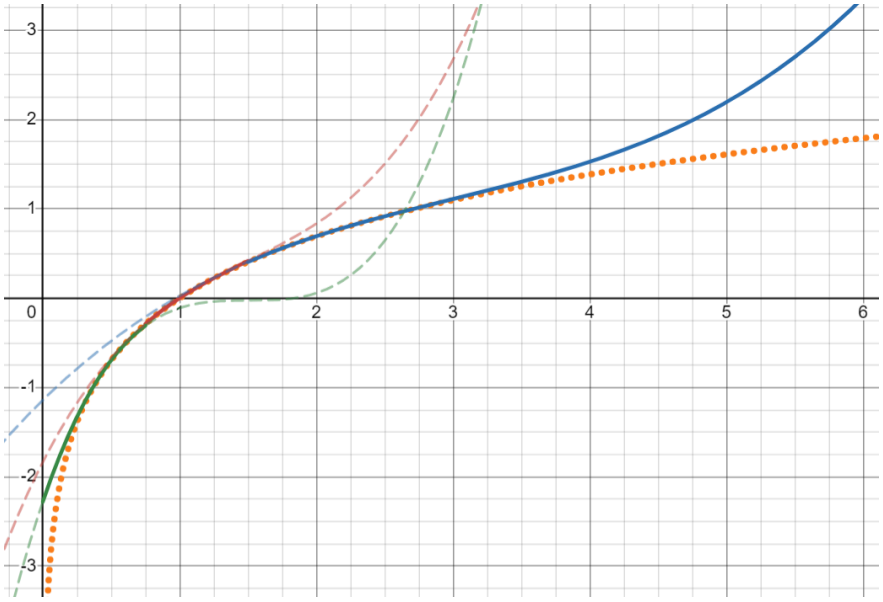
לפורמט זה מספר הבדלים משימוש קלאסי במספרים בינאריים שלמים, למשל עבור כפל נסמן a כמספר בפורמט Qx.y וA עבור קריאת הערך כבינארי רגיל, אזי מאחר וכל מספר מביא עימו את הנקודה, בעת מכפלה יתקיים

$$A * B = a * 2^y * b * 2^y = [(a * b) * 2^y] * 2^y$$

לכן כדי לקבל את תוצר המכפלה ab יש צורך לחלק את תוצר מכפלת AB ב2^y או להזיזו ימינה y פעמים. נציין כי הזזות אלו משפיעות על תחום השימוש האפשרי מאחר ולאחר ההזזה נשמר המידע רק מx ביטי הMSB .

כמו כן, עבור חישוב פונקציית לוגריתם, נדרשת פונקציה ייעודית לפורמט.
פונקציה זו צריכה להיות מבוססת על קירוב טיילור. בכדי להקל על כמו החישובים הנדרשים וכדי להתאים את קירוב הטיילור הן עבור ערכים קטנים מ1 והן עבור ערכים גדולים מ1 בוצעו שלושה קירובי טיילור שונים, סביב 0.5, 1 ו2, כך שכל ערך

מקורב טיילור לפי הקירוב אליו הוא סמוך, לכן ניתן להשתמש בקירוב מסדר נמוך (למשל 3) ולקבל קירוב קל חומרתית וקרוב באופן הבא:



מחולל מספרים רנדומליים

לצרכי דימוי התנהגות שגיאות, נדרשים מספרים רנדומיים מחוללים באופן חומרתי תקני. מאחר וSystemVerilog שקולה לחומרה אידיאלית אין ביכולתנו להסתמך על רעש פיזיקלי (למשל חשמל סטטי חיווט צף המחובר למגבר) ועלינו לייצר אלגוריתם מחולל מספרים פסודו-רנדומי, כלומר כזה המתפלג באופן דומה למספר רנדומי. לאחר סקר ספרות קצר בחרנו לממש את אלגוריתם XORSHIFT באופן מותאם לצרכינו. זהו אלגוריתם יעיל וקצר שאינו מאוד רנדומי ביחס לאלגוריתמים מוכרים אחרים אך הוא נוח מאוד למימוש, במיוחד בחומרה בכך שהוא דורש רק פעולות XOR וSHIFT ולא דורש פולינומים גבוהים, והוא פועל על מנגנון חוזר של ביצוע הזזות הXOR עם עצמו טרם ההזזות, כאשר התוצר של המנגנון מהווה נקודת ההתחלה לאיטרציה הבאה. לדוגמא עבור 32 ביטים מותאם הפסודו-קוד:

```
loop:
    x ^= x << 13;
    x ^= x >> 17;
    x ^= x << 5;
    Out = x;
```

נציין כי פורמט זה מאותחל להכיל ביטי 1 בכולו והוא אינו יכול להתמודד עם רצף אפסים, אם כי גרסאות מתקדמות יותר של האלגוריתם פותרות את בעיה זו.

כדי להתאים את האלגוריתם לצרכינו בנינו מחולל שמזין בכל איטרציה 32 ביט רנדומיים לתוך תור (Queue). מאחר ונדרשים לנו בכל מחזור שעון מספר גבוהה של מספרים רנדומיים לצורך חישוב שגיאות IDS המתרחשות במהלך יצירת הסליל, לקחנו תור ארוך לכל אחד מגורמי השגיאות (IDS) וייחסנו להם שבר ספורמט Q0.32, באמצעות מצביע שזז עם כל דגימה. כדי לשמור על רנדומיות נפרדת בין המחוללים קבענו שכל מחולל בכל מחזור שעון יחזור על הזנת 32 ביטים רנדומיים נוספים כSEED פעמים, כאשר SEED הינו מספר ראשוני גבוהה ושונה לכל מחולל.