The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

הפקולטה להנדסה
ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

Velocity Subwords : Emphasizes speed and the subword nature for Multimedia operation on simple processor

פרויקט מס' 3173

דו"ח סיכום

מבצעים:

תום ברנשטיין

עמית רחמיאל

מנחים:

אורן גנון

מקום ביצוע הפרויקט:

אוניברסיטת תל אביב

תוכן עניינים:

רשימת איורים

רשימת טבלאות:

תקציר

מטרת הפרויקט המרכזית היא להרחיב את ארכיטקטורת המעבד הבסיסי מסוג DLX על ידי הוספת ערכת הוראות SIMD (Single Multiple Data Instruction) המיועדת לפעולות מולטימדיה ועיבוד אותות דיגיטליים (DSP). הרחבה זו תתבצע באמצעות שינויים מקיפים במסלול הנתונים (Datapath) וביחידת הבקרה (Control Unit) של המעבד. שינויים אלו יאפשרו עיבוד מקבילי של נתונים קטנים, בגודל 8 ו-16 ביטים, תוך פיתוח יחידת ALU מקבילית התומכת ב-39 פעולות SIMD כגון חיבור וקטורי, חיסור, השוואות וסידור מחדש של נתונים.

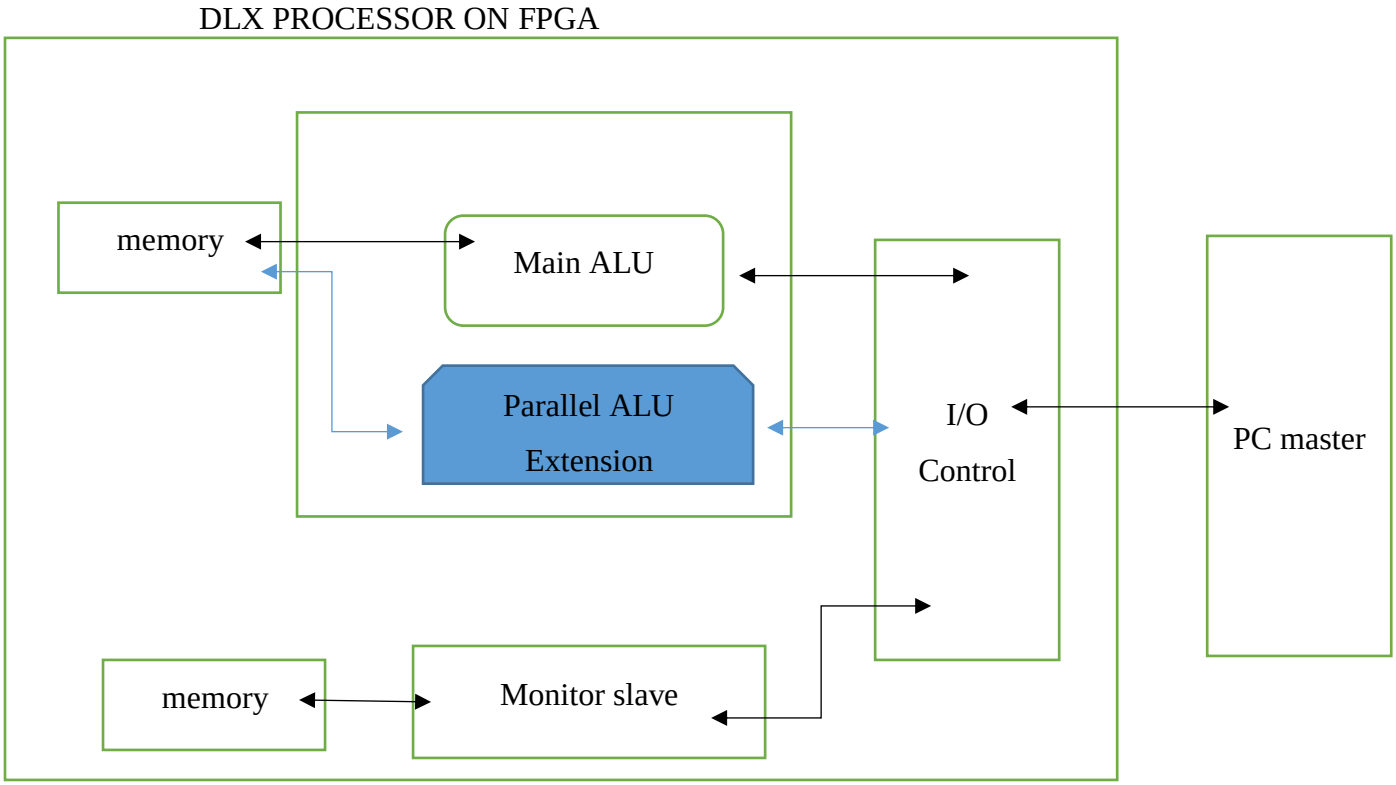
סביבת הפרויקט כוללת שימוש בסימולטור מעבד ותוכנת Xilinx/RESA, המשמשות לתכנון, אימות ובדיקת חומרה. הסימולציות והבדיקות ייבחנו במגוון תרחישים הרלוונטיים לעיבוד מולטימדיה ועיבוד אותות דיגיטלי. שלבי העבודה העיקריים בפרויקט כוללים: הבנה מעמיקה של מבנה ארכיטקטורת המעבד ועקרונות הוראות SIMD, בניית חומרה וסימולציה של מעבד DLX בסיסי במחזור בודד (Single Cycle), תכנון דיגיטלי מחדש של מסלול הנתונים ומסלול ההוראות לתמיכה בפעולות תת-מילים מקבילות, פיתוח קוד חומרה (בשפת Verilog) עבור פקודות SIMD חדשות, פיתוח אסמבלר להמרת קוד אסמבלי לקבצי DATA, מימוש הארכיטקטורה המורחבת בסביבת FPGA,ולבסוף, הרצת בדיקות מקיפות ובחינת ביצועי הפקודות החדשות בתרחישים יישומיים וניתוח ביצועים בתרחישים כמו Image, Block Matching, Matrix Transpose Sharpening.

תוצרי הפרויקט כוללים מעבד DLX מורחב עם תמיכה בפקודות SIMD, המיושם על FPGA (Xilinx Spartan-6), קוד Verilog מלא של החומרה, קוד אסמבלי ליישומים נבחרים, ואסמבלר תומך לסימולציות. בדיקות הראו שיפור של עד פי 15.5 במספר הסייקלים (למשל, 17 לעומת 175 ב Block Matching) וחיסכון של 88-99% בהספק, אך עם עלייה של 128.57% בשטח ה -FPGA.

התיעוד כולל דוחות ביצועים, תרשימים, ומדריך למשתמש, הזמינים ב GitHub.

דיאגרמת בלוקים של הארכיטקטורה המורחבת מוצגת להלן. החומרה לכלל רכיבי המעבד, כולל ההרחבות, פותחה במסגרת הפרויקט.

איור 1: דיאגרמת בלוקים כללית של מעבד DLX על FPGA :



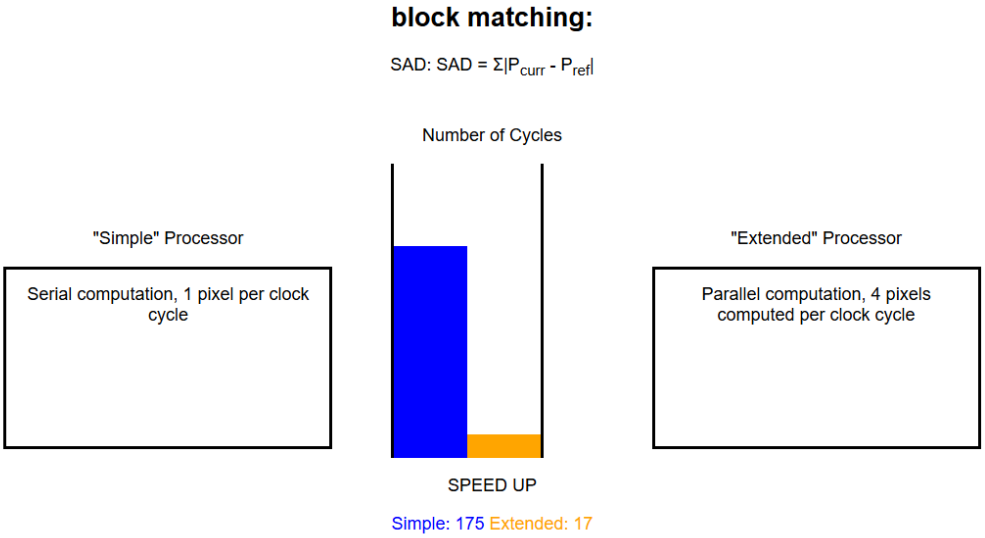
1 הקדמה

הפרויקט מתמקד בהרחבת ארכיטקטורת מעבד DLX על ידי שילוב ערכת הוראות SIMD לעיבוד מקבילי של נתונים קטנים בגודל 8 ו-16 ביט, המיועדים ליישומי מולטימדיה ועיבוד אותות דיגיטליים .מטרתו היא לשפר את ביצועי המעבד ביישומים כגון Block Matching, Matrix Transpose ו -Image Sharpening.תוך הפחתת מספר מחזורי השעון ב-20% לפחות והפחתת צריכת אנרגיה ב-10-15% בהשוואה למעבד הבסיסי. בנוסף, הפרויקט שואף לממש את הארכיטקטורה המורחבת על FPGA תוך שמירה על יעילות שטח, כדי להתאים למערכות משובצות עם משאבים מוגבלים.

יישומי מולטימדיה, כגון עיבוד תמונה ווידאו דורשים ביצועים גבוהים ויעילות אנרגטית, במיוחד במערכות משובצות. ארכיטקטורות פשוטות כמו DLX, למרות יתרונן בלימוד ופיתוח, מוגבלות בעיבוד סדרתי, מה שמגביל את יכולתן לעמוד בדרישות אלה. שילוב הוראות SIMD מאפשר ביצוע פעולות זהות על מספר נתונים במקביל בתוך מחזור שעון יחיד, ובכך משפר משמעותית את המהירות ומפחית צריכת אנרגיה. בחירת DLX כפלטפורמה נובעת מפשטותה, המאפשרת התמקדות בהרחבת חומרה ספציפית ליישומים אלו תוך שימוש בתוספת משאבים מינימלית, מה שהופך אותה לאידיאלית להדגמת היתכנות הפתרון.

הגישה לפתרון כללה עיצוב מחדש של מסלול הנתונים ויחידת הבקרה של מעבד DLX, תוך פיתוח יחידת ALU מקבילית התומכת ב-39 פעולות SIMD, כגון PADDUB לחיבור מקבילי של ביטים ,PCMPEQB להשוואת ביטים ו PERMUTEB-לסידור מחדש של נתונים. תהליך הפיתוח כלל סימולציה של מעבד DLX בסיסי במחזור בודד בסביבות RESA ו -ModelSim,תכנון ומימוש פעולות SIMD ב Verilog,פיתוח אסמבלר להמרת קוד אסמבלי לקובצי DATA. עבור סימולציות. הארכיטקטורה המורחבת יושמה על FPGA מסוג Xilinx Spartan-6, ונבדקה בתרחישים יישומיים כדי להעריך את שיפור הביצועים ואת צריכת האנרגיה.

פתרונות מסחריים, כגון הרחבת NEON של ARM ו AVX-של Intel מספקים יכולות SIMD מתקדמות לעיבוד גרפי ואותות, אך דורשים חומרה מורכבת וצורכים הספק גבוה יותר. לעומתם, פרויקט זה מתמקד בהרחבת ארכיטקטורה פשוטה כמו DLX, תוך שימוש בתוספת חומרה מינימלית, כגון ALU מקבילי ורכיבי בקרה נוספים. התוצאות הראו שיפור של פי 3 עד 15.5 במספר מחזורי השעון וחיסכון של 88-99% בצריכת ההספק בהשוואה למעבד הבסיסי, אם כי עם עלייה של 128.57% בשטח ה -FPGA.בכך, הפרויקט מדגים פתרון יעיל וחסכוני ליישומי מולטימדיה במערכות עם משאבים מוגבלים, תוך שמירה על פשטות הארכיטקטורה.



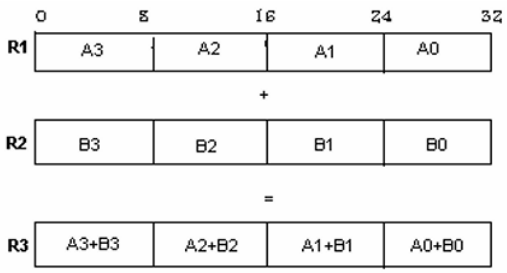
2 רקע תיאורטי

ארכיטקטורת SIMD היא גישה למחשוב מקבילי שבה פקודה אחת מבוצעת בו-זמנית על מספר נתונים, מה שהופך אותה ליעילה במיוחד עבור יישומים עם מקביליות נתונים גבוהה, כגון עיבוד תמונה, גרפיקה ממוחשבת, ועיבוד אותות דיגיטליים . במעבדי SIMD, יחידת עיבוד אחת או יותר מבצעות פעולה זהה על וקטור של נתונים, כמו פיקסלים בתמונה, בתוך מחזור שעון יחיד. בפרויקט זה, ארכיטקטורת DLX הורחבה ליישום עקרונות SIMD על ידי חלוקת אוגרי 32 ביט ויחידת ה ALU-לעיבוד מקבילי של תת-מילים בגודל 8 או 16 ביט. הרחבה זו מאפשרת ביצוע פעולות אריתמטיות ולוגיות על מספר פיקסלים במקביל, ובכך משפרת את ביצועי המעבד ביישומי מולטימדיה תוך הפחתת צריכת אנרגיה.

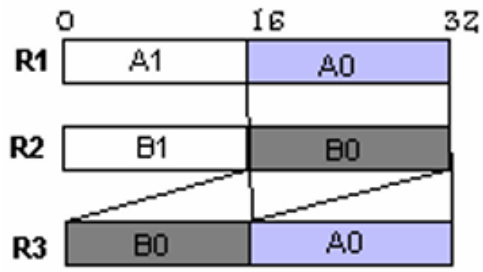
פיקסל הוא היחידה הבסיסית של תמונה דיגיטלית, המייצגת נקודה עם מידע על עוצמת אור. תמונה דיגיטלית ניתנת לייצוג כמטריצה דו-ממדית של פיקסלים, כאשר רזולוציית התמונה תלויה במספר הפיקסלים – יותר פיקסלים משמעותם תמונה מפורטת יותר. בתמונות בגווני אפור (Grayscale), פיקסל מיוצג ב-8 סיביות, המאפשרות 256 ערכים (0 לשחור, 255 ללבן, וערכי ביניים לגווני אפור). עיבוד תמונה כולל פעולות מתמטיות על ערכי פיקסלים, כמו הבהרה, הכהיה, שינוי ניגודיות, או סינון, הדורשות חישובים רבים על נתונים קטנים. ארכיטקטורת SIMD מתאימה להאצת תהליכים אלה, שכן היא מאפשרת ביצוע פעולות זהות על מספר פיקסלים במקביל, מה שמפחית את זמן העיבוד בהשוואה לעיבוד סדרתי.

הצורך ב -ALU מקבילי נובע מהדרישה להאיץ עיבוד תמונה על ידי ביצוע חישובים על מספר פיקסלים בו-זמנית. במעבד DLX הבסיסי, ה -ALU פועל על מילים של 32 ביט, מה שמגביל אותו לעיבוד פיקסל אחד בכל מחזור. בפרויקט זה, ה -ALU הורחב לחלוקה ליחידות משנה, כך שמילה של 32 ביט יכולה לייצג ארבעה פיקסלים של 8 ביט או שני פיקסלים של 16 ביט. ה ALU-תומך בשלושה מצבי פעולה, הנבחרים באמצעות בורר(MUXALU_SEL) : פעולה סטנדרטית של 32 ביט, פעולה מקבילית על ארבע תת-מילים של 8 ביט, ופעולה מקבילית על שתי תת-מילים של 16 ביט. בין הפעולות המומשות :PADDUB מבצעת חיבור מקבילי של ארבעה זוגות בייטים עם קטימה ל-255, ומשמשת להבהרת תמונה או מיזוג תמונות. PSUBUB מבצעת חיסור מקבילי עם קטימה ל-0, ומתאימה להכהיית תמונה. פעולות לוגיות כמו PAND, POR ו PXOR- מאפשרות מניפולציות על ביטים ברמת הפיקסל. PCMPEQB משווה ארבעה זוגות בייטים במקביל ומשמשת לזיהוי תבניות. PERMUTEB מסדרת מחדש בייטים בתוך אוגר ומשמשת לסיבוב תמונות. PACKLH לוקחת את 16 הביטים הנמוכים משני אוגרים rs1 ו rs2 ומארזת אותם לאוגר יחיד של 32 ביט (16 ביט נמוכים מ rs1-ואחריהם 16 ביט נמוכים מrs2) מה ששימושי להכנת נתונים עבור פעולות נוספות בעיבוד תמונה.

איור 2: דוגמה לפעולת חיבור מקבילית(PADDUB)

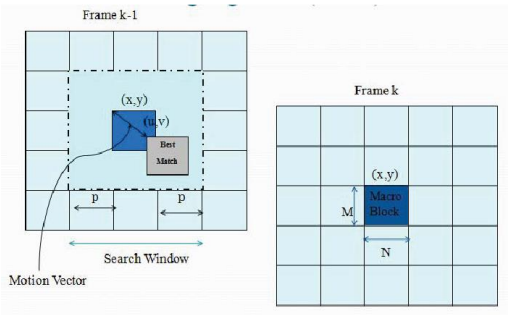


איור 3: דוגמא לפעולת אריזה(PACKLH)



הפרויקט התמקד בשלושה אלגוריתמים מרכזיים לעיבוד תמונה: Block Matching מחשב את Sum of Absolute Differences (SAD) בין שני בלוקים של פיקסלים לזיהוי תנועה בווידאו, Matrix Transpose מחליף שורות ועמודות במטריצה, ומשמש לסיבוב תמונות או הכנת נתונים, Image Sharpening משתמש במסנן Laplacian להדגשת קצוות בתמונה לשיפור ניגודיות. במעבד DLX הבסיסי, אלגוריתמים אלה מבוצעים באופן סדרתי, כאשר כל פעולה על פיקסל דורשת מחזור שעון נפרד, מה שמוביל למספר גבוה של מחזורי שעון (למשל, 175 עבור Block Matching). לעומת זאת, במעבד המקבילי עם הרחבת SIMD, פעולות מבוצעות על ארבעה פיקסלים במקביל, מה שמפחית משמעותית את מספר המחזורים (למשל, 17 עבור Block Matching). אלגוריתמים אלטרנטיביים כוללים עיבוד סדרתי במעבד DLX, שדורש זמן רב יותר.חלופות כמו מעבדי DSP ייעודיים או הרחבות כמו ARM NEON מספקות יכולות דומות, אך הן מורכבות יותר ופחות מתאימות למערכות משובצות עם משאבים מוגבלים. הגישה בפרויקט זה, המשלבת הרחבת SIMD פשוטה על DLX, השיגה שיפור של פי 3-15.5 במספר מחזורי השעון וחיסכון של 88-99% בהספק, עם תוספת חומרה מינימלית, מה שהופך אותה ליעילה עבור יישומי מולטימדיה במערכות קומפקטיות.

איור 4: דוגמא לחישוב וקטור תנועה עבור block matching



חישוב ווקטור התנועה עבור בלוק 2X2 :

$$sad = |P_{00} - R_{00}| + |P_{01} - R_{01}| + |P_{10} - R_{10}| + |P_{11} - R_{11}|$$

איור 5: דוגמא לחישוב לפלאסיאן עבור image sharping

0	-1	0
-1	4	-1
0	-1	0

חישוב הלפלאסיאן עבור בלוק 2X2:

Laplacian = 4 × pixel - neighbor1 - neighbor2

3. סימולציה

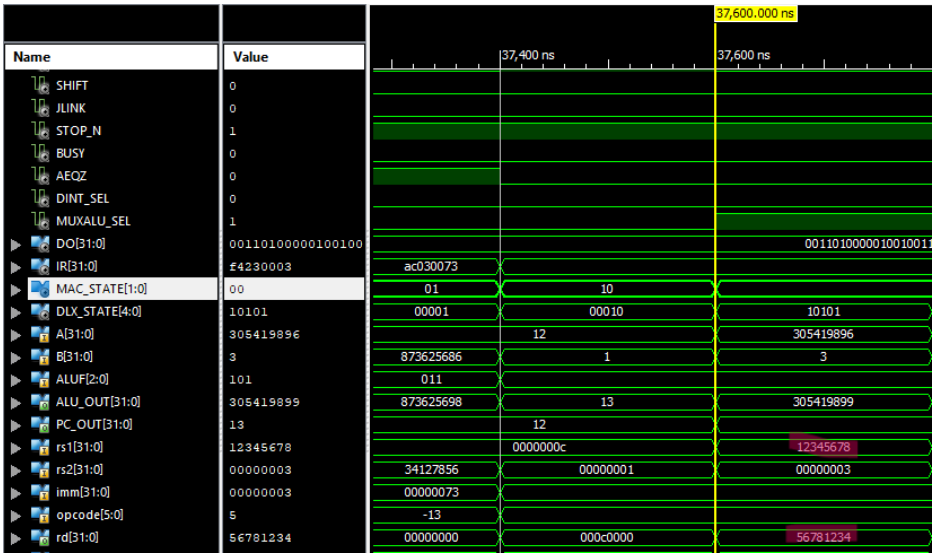
סביבת הסימולציה של הפרויקט נועדה לאמת את תקינות הלוגיקה של מעבד DLX הבסיסי ואת הרחבתו עם 39 הוראות SIMD לעיבוד מקבילי של נתונים קטנים (8 ו-16 ביט) עבור יישומי מולטימדיה, וכן לבדוק את אותות הבקרה ורכיבי ה Datapath החדשים של המעבד המורחב. הסביבה כללה שני כלים עיקריים:
modelSim בתוך תוכנת הXlinix ותוכנת הRESA.
modelSim שימש לבדיקה לוגית של קוד ה Verilog של המעבד, תוך אימות תקינות ה ALU המקבילי וכל 39 פעולות ה SIMD (כגון PADDUB, PCMPEQB, PERMUTEB) ברמת הרכיבים, כדי להבטיח נכונות לוגית של הקוד .RESA שימש לבדיקות המעבד על חומרה אמיתית ב FPGA (Xilinx Spartan-6) תוך הערכת תזמונים, ביצועים וצריכת הספק. בנוסף, באמצעות Xilinx ביצענו סינטזה לקוד ה Verilog-של המעבד, מה שאפשר תרגום העיצוב הלוגי ליישום חומרה פיזי על FPGA תוך אופטימיזציה של משאבים וזיהוי בעיות תזמון. האסמבלר שפותח בפרויקט המיר קוד אסמבלי לקובצי DATA. ששימשו כקלט לסימולציות, ובכך אפשרו בדיקה מדויקת של הפקודות החדשות הן בסימולציה לוגית והן על חומרה.

הסימולציות התמקדו בשלושה יישומים מרכזיים של עיבוד תמונה: Block Matching, Matrix Transpose ו-Image Sharpening וכן בבדיקה מקיפה של כל 39 פעולות ה SIMD.עבור Block Matching, נבדק חישוב Sum of Absolute Differences (SAD) בין שני בלוקים של פיקסלים, תוך שימוש בפקודות כמו PABSUBB ו UNPACKLB.הסימולציה ב ModelSim-אימתה את נכונות הפעולות המקביליות, ו RESA הראתה שיפור של פי 10.3 במספר המחזורים (17 לעומת 175) על FPGA. עבור Matrix Transpose, נבדקה החלפת שורות ועמודות במטריצה, תוך שימוש בפקודות כמו PERMUTEB, עם שיפור של פי 15.5 במחזורי השעון (4 לעומת 62). עבור Image Sharpening, נבדקה החלת מסנן Laplacian להדגשת קצוות, תוך שימוש בפקודות כמו PADDUB ו ,PSUBUB והושג שיפור של פי 3.2 במחזורי השעון (94 לעומת 297).

כל 39 פעולות ה SIMD-נבדקו ב ModelSim-לאימות לוגי וב RESA-על FPGA לאימות חומרה, והוכחו כתקינות בתפוקותיהן מול תוצאות צפויות.

מטרת הבדיקה הלוגית ב ModelSim הייתה להבטיח נכונות לוגית של קוד ה Verilog,תוך אימות פעולות ה-ALUהמקבילי, אותות הבקרה(כגון MUXALU_SEL לבחירת מצבי פעולה),ורכיבי ה Datapath החדשים, כמו רגיסטרים ונתיבים לעיבוד תת-מילים. בדיקות ה FPGAב RESA,בשילוב עם סינטזת הקוד בxlinix , אפשרו הערכת התנהגות המעבד בסביבה אמיתית, כולל תזמונים וצריכת הספק.
התוצאות הראו חיסכון של 88-99% בהספק(למשל, 0.045 mJ לעומת 1.254 mJ ב Block Matching) ועלייה של 128.57% בשטח ה-FPGA (959 לעומת 419 Slices).
הסימולציות אפשרו זיהוי ותיקון שגיאות, כמו גלישות בפעולות אריתמטיות, ואופטימיזציה של החומרה והתוכנה. בדיקת האסמבלר אימתה את תקינות קובצי ה DATA.מול קוד המכונה הצפוי, מה שהבטיח תאימות בין הקוד האסמבלי לחומרה. תוצאות הסימולציות הדגימו את יעילות הרחבת ה SIMD ביישומי מולטימדיה.

איור 6: דוגמא לבדיקה לוגית של פקודת PERMUTEH בסביבת modelSim של Xlinix :



ניתן לראות בזמן 37600ns את האות הבקרה MUXALU_SEL נדלק וכך אנחנו יודעים שאנחנו במצב של עיבוד מקבילי. ניתן לראות שמצב המעבד(DLX_STATE) נמצא במצב בינארי 10101 שזה 21 דיצמלי וזה מצב ALUPI שכן זה פקודה מקבילית ופקודה מסוג I type.

הפעולה מחליפה בין 16 הביטים התחתונים ל16 ביטים העליונים של מילה בת 32 ביט כמו שניתן לראות באיור.(מסומן בסגול, הקלט rs1 והפלט rd) .

דוגמא של BLOCK MTAHCING:

בלוק נוכחי :(curr_blo) מאוחסן במילה של 32 ביט:

x04134930 → [P3=04, P2=13, P1=49, P0=30]

-בלוק ייחוס :(ref_blo) מאוחסן במילה של 32 ביט:

x11223344 → [REF3=11, REF2=22, REF1=33, REF0=44]

sad = $|P_{00} - R_{00}|+|P_{01} - R_{01}|+|P_{10} - R_{10}|+|P_{11} - R_{11}|$

תוצאה לדוגמה

-פיקסל 1: |30-44| = 14.

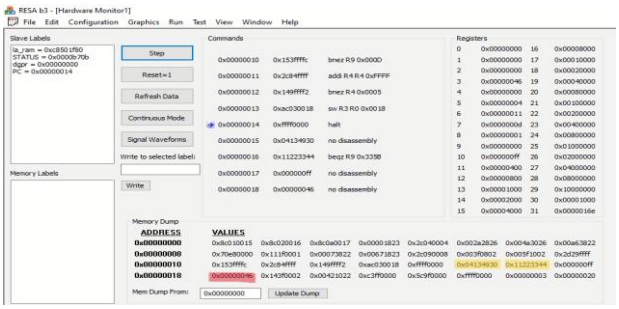
-פיקסל 2: |49-33| = 16.

-פיקסל 3: |22-13| = F.

-פיקסל 4: |11 - 04| = D.

SAD = 14+16+F+D =46.

איור 7: דוגמא לבדיקה חומרתית של פעולת block matching עבור מעבד בסיסי בסביבת resa, ניתן לראות בצהוב את הקלטים ובאדום את הפלט.

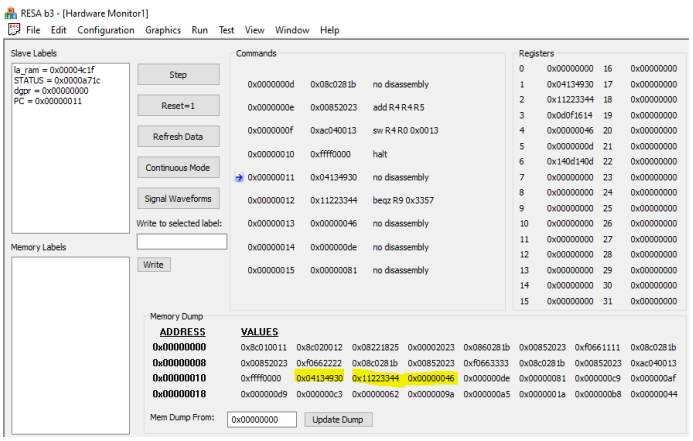


איור 8: דוגמא לבדיקה לוגית של פעולת block matching בסביבת modelSim של Xlinix , עבור המעבד המורחב התומך בפעולות מקביליות.

מצורף איור של הזיכרון החיצוני של המעבד, עם 2 קלטים והפלט .

	0	1	2	3
0	8C010011	8C020012	08221825	00002023
4	0860281B	00852023	F0661111	08C0281B
8	00852023	F0662222	08C0281B	00852023
12	F0663333	08C0281B	00852023	AC040013
16	00000000	04134930	11223344	00000046

איור 9: דוגמא לבדיקה חומרתית של פעולת block matching עבור המעבד המורחב התומך בפעולות מקביליות בסביבת resa, ניתן לראות בצהוב את הקלטים והפלט.



דוגמא של MATRIX TRANSPOSE:

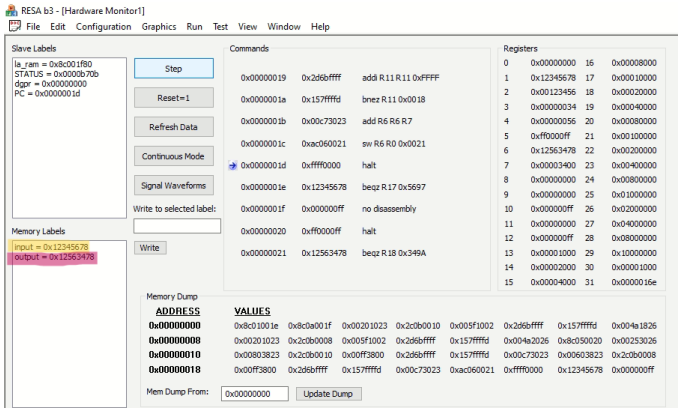
מחליפה בין שורות ועמודות של מטריצה.

Matrix Transpose עבור מטריצה 2x2 (4 פיקסלים של 8 ביט) על ידי החלפת הערכים B ו-C במילה אחת של 32 ביט, כך ש- [A, B, C, D] הופך ל- [A, C, B, D].

תוצאה לדוגמה

- קלט: 0x12345678 -> [A=12, B=34, C=56, D=78].
- פלט: 0x12563478 -> [A=12, C=56, B=34, D=78].

איור 10: דוגמא לבדיקה חומרתית של פעולת matrix transpose עבור מעבד בסיסי בסביבת resa, ניתן לראות בצהוב את הקלט ובאדום את הפלט.

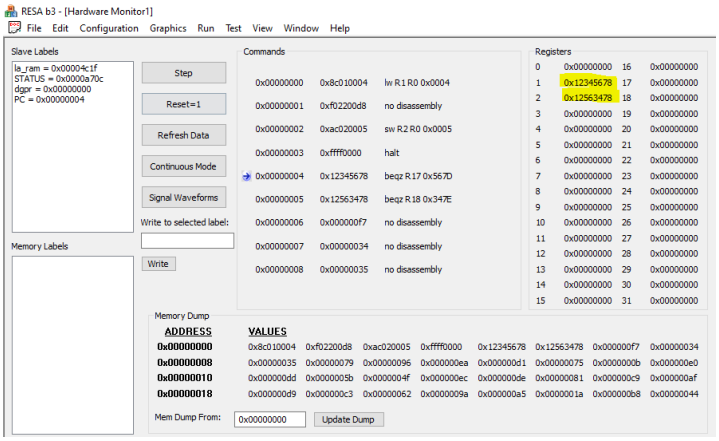


איור 11: דוגמא לבדיקה לוגית של פעולת matrix transpose בסביבת modelSim של Xlinix , עבור המעבד המורחב התומך בפעולות מקביליות.

מצורף איור של זיכרון הרגיסטרים של המעבד, R1 עבור הקלט ו R2 עבור הפלט .

	0	1	2	3
0	00000000	12345678	12563478	00000000
4	00000000	00000000	00000000	00000000

איור 12: דוגמא לבדיקה חומרתית של פעולת matrix transpose עבור המעבד המורחב התומך בפעולות מקביליות בסביבת resa, ניתן לראות בצהוב את הקלט והפלט.



דוגמא של image sharping :

- בלוק קלט :מאוחסן במילה של 32 ביט

0x12345678 →[P00=12,P01=34,P10=56,P11=78]

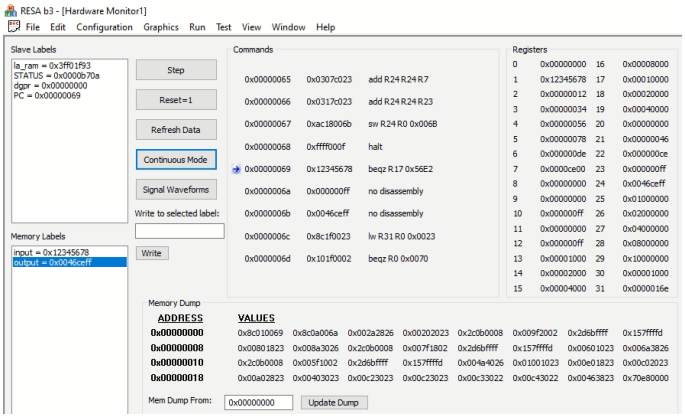
Laplacian = 3 × pixel - neighbor1 - neighbor2

S00 = P00 + laplacian

תוצאה:(חישוב בדמצלי)

- S00: 3×18 - 52 - 86 = -84 → P00 + (-84) = 18 - 84 = -66 → clipped to 0.
- S01: 3×52 - 18 - 120 = 18 → 52 + 18 = 70.
- S10: 3×86 - 18 - 120 = 120 → 86 + 120 = 206.
- S11: 3×120 - 52 - 86 = 222 → 120 + 222 = 342 → clipped to 255.
- פלט: 0x0046CEFF → [S00=0, S01=70, S10=206, S11=255].

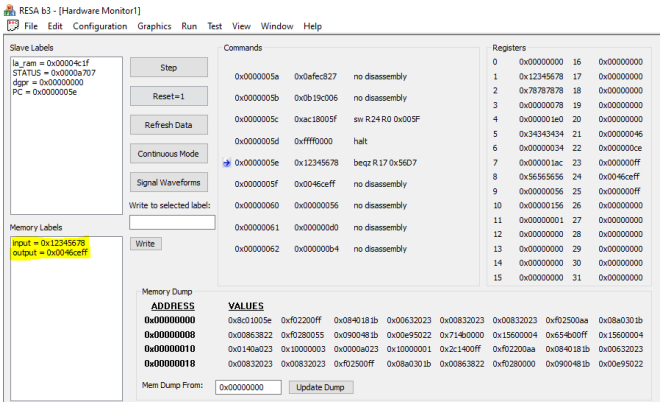
איור 13 : דוגמא לבדיקה חומרתית של פעולת image sharping עבור מעבד בסיסי בסביבת resa



איור 14: דוגמא לבדיקה לוגית של פעולת image sharping בסביבת modelSim של Xlinix , עבור המעבד המורחב התומך בפעולות מקביליות.
מצורף איור של הזיכרון החיצוני של המעבד, מודגש בצהוב הקלט והפלט .

80	2C1700FF	2C1E0003	0A9EC027	2C1E0002	0ABEC827	0B19C006	2C1E0001	0ADEC827
88	0B19C006	2C1E0000	0AFEC827	0B19C006	AC18005F	00000000	12345678	0046CEFF

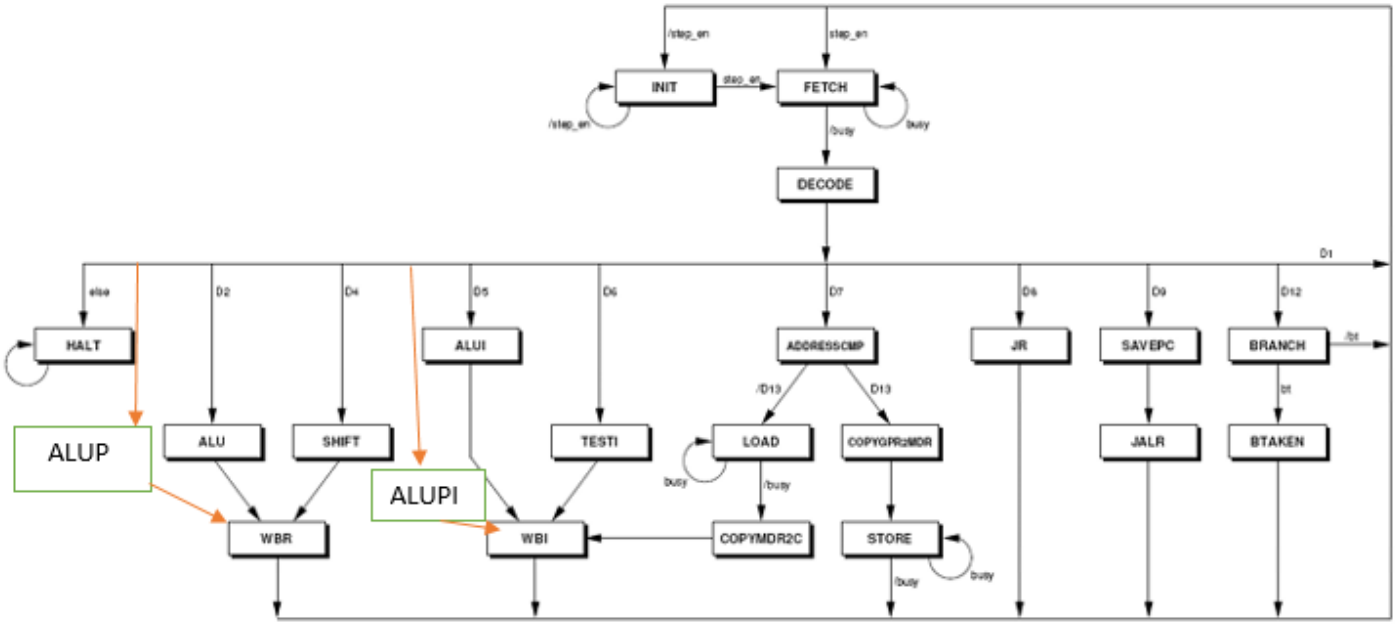
איור 15: דוגמא לבדיקה חומרתית של פעולת image sharping עבור המעבד המורחב התומך בפעולות מקביליות בסביבת resa, ניתן לראות בצהוב את הקלט והפלט.



4. מימוש

מימוש הפרויקט התמקד בפיתוח מעבד DLX בסיסי במחזור בודד, הכולל מסלול נתונים ויחידת בקרה ,ולאחר מכן הרחבתו לעיבוד מקבילי של נתונים קטנים (8 ו-16 ביט) באמצעות ערכת הוראות SIMD עבור יישומי מולטימדיה, כגון Block Matching, Matrix Transpose וImage Sharpening. השיקולים לבחירת המימוש כללו שמירה על פשטות הארכיטקטורה, תוספת חומרה מינימלית להשגת יעילות שטח, ושיפור משמעותי בביצועים (פי 3-15.5 במחזורי שעון) וצריכת הספק (חיסכון של 88-99%). ההרחבה כללה בניית ALU מקבילי התומך ב-39 פקודות חדשות, קידוד יעיל של opcodes עבור הפקודות החדשות, שינוי רכיב ה Instruction-Register (IR) לקריאה ופענוח הפקודות המקביליות, הוספת שני מצבים חדשים במכונת המצבים (ALUP ו-ALUPI) לעיבוד מקבילי, יצירת אות בקרה חדש (MUXALU_SEL) לבחירה בין ה ALU-הבסיסי למקבילי, תיאום אותות הבקרה הנוספים לתמיכה במצבים החדשים, ויצירת מוקס חדש לבחירה בין פעולות המעבד הבסיסי למקבילי. בנוסף, נכתבו שלושה קודי אסמבלי לעיבוד תמונה (Block Matching, Matrix Transpose Image Sharpening)עבור המעבד הבסיסי, ושלושה קודי אסמבלי עם פקודות SIMD עבור המעבד המורחב. המימוש יושם על FPGA מסוג Xilinx Spartan-6 ודיאגרמת בלוקים של הארכיטקטורה המורחבת מוצגת להלן.

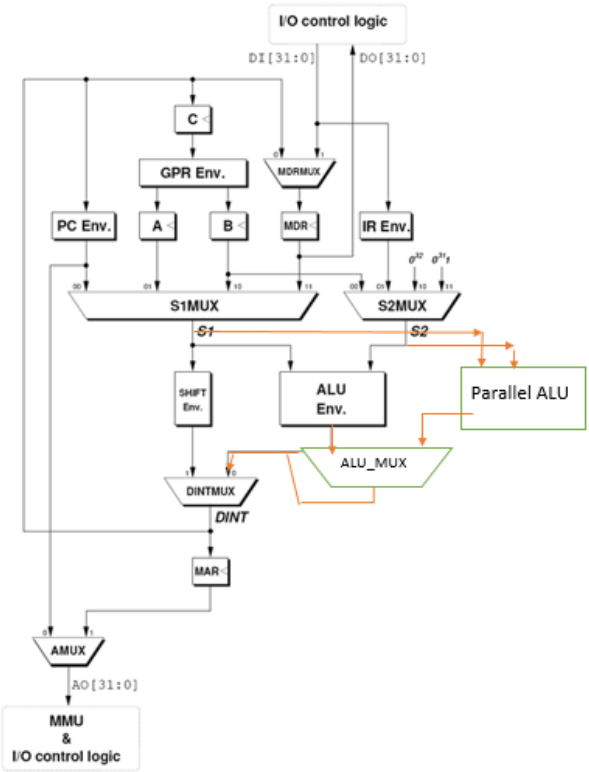
איור 16: מכונת המצבים של יחידת הבקרה:



איור 17: מבנה פקודת R TYPE :

6	5	5	5	5	6
Opcode	RS1	RS2	RD		Function

איור 18: מסלול הנתונים של המעבד:



4.1. מימוש קוד החומרה

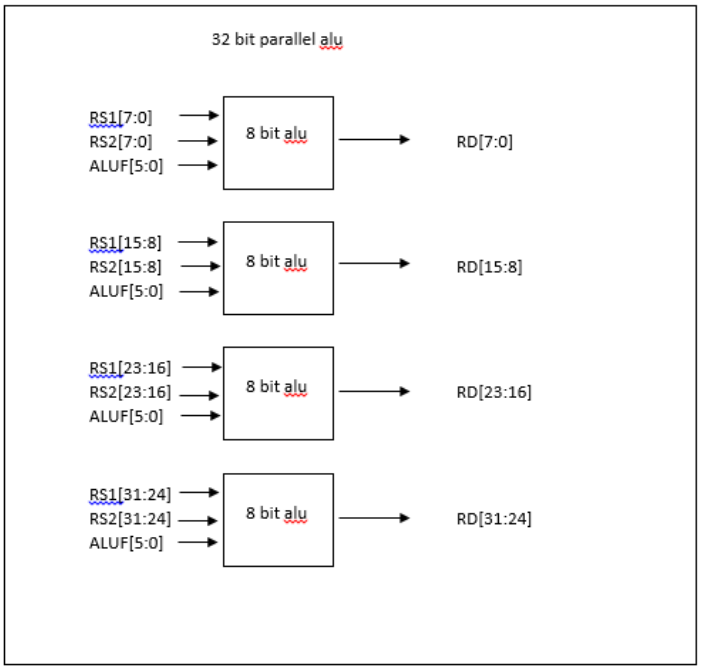
מימוש החומרה התבסס על FPGA מסוג Xilinx Spartan-6 (xc6slx25), שנבחר בשל עלותו הנמוכה, תמיכתו ב Xilinx,ויכולתו ליישם מעבדים מורכבים במשאבים מוגבלים. תחילה פותח מעבד DLX בסיסי עם מסלול נתונים הכולל רגיסטרים של 32 ביט, ALU סטנדרטי, וזיכרון נתונים והוראות, ויחידת בקרה המנהלת את מחזור ההוראה. ההרחבה כללה עיצוב ALU מקבילי התומך ב-39 פקודות SIMD כגון PADDUB PSUBUB, PCMPEQB, PERMUTEB עם שלושה מצבי פעולה: 32 ביט סטנדרטי, 4 תת-מילים של 8 ביט, ו-2 תת-מילים של 16 ביט. רכיב ה IR שונה כדי לקרוא ולפענח את הפקודות החדשות באמצעות קידוד יעיל של opcodes, תוך שימוש בשדות opcode מורחבים תחת מצבי ALUP ו ALUPI במכונת המצבים. אות בקרה חדש, MUXALU_SEL, נוסף לבחירה בין ה ALU הבסיסי למקבילי, ומוקס חדש תוכנן לבחירה בין פעולות המעבד הבסיסי למקבילי. שאר אותות הבקרה,תואמו לתמיכה במצבים החדשים, תוך הבטחת תקינות זרימת הנתונים. הכלים העיקריים היו Xilinx לסינטזה, מימוש וניתוח תזמונים, ו ModelSim לבדיקה לוגית של קוד ה Verilog. הסינטזה ב xlinix תרגמה את העיצוב לחומרה פיזית, עם עלייה של 128.57% בשטח.

איור 19: FPGA מסוג Xilinx Spartan-6

טבלה 1: פקודות מורחבות עבור ALU מקבילי:

Command	Algorithm	Use in Image Processing
psllbi	shift left bytes by imm	make image brighter
psllhi	shift left half-words by imm	make image brighter
psrlbi	shift right logical bytes by imm	make image darker
psrlhi	shift right logical half-words by imm	make image darker
permuteb	reorder bytes	change pixel order
permuteh	reorder half-words	change pixel order
paddub	add bytes with saturation	mix two images
paddh	add half-words with saturation	mix two images
psubub	subtract bytes with saturation	detect motion
psubuh	subtract half-words with saturation	detect motion
psllb	shift left bytes by rs2	make image brighter
psllh	shift left half-words by rs2	make image brighter
psrlb	shift right logical bytes by rs2	make image darker
psrlh	shift right logical half-words by rs2	make image darker
psrah	shift right arithmetic half-words by rs2	adjust pixels
pcmpeqb	compare equal bytes	match pixels
pcmpgtb	compare greater than bytes	find bright spots
pcmpleb	compare less or equal bytes	filter pixels
pcmpeqh	compare equal half-words	match pixels
pcmpgth	compare greater than half-words	find bright spots
pcmpleh	compare less or equal half-words	filter pixels
mixl	mix left halves	mix images
mixr	mix right halves	mix images
packhh	pack high half-words	make image smaller
packlh	pack low half-words	make image smaller
unpackh	unpack high half-word (zero)	make image bigger
unpacksh	unpack high half-word (signed)	make image bigger
unpacklb	unpack low byte (zero)	stretch image
unpackslb	unpack low byte (signed)	stretch image
unpackhb	unpack high byte (zero)	stretch image
unpackshb	unpack high byte (signed)	stretch image
porb	bitwise OR on bytes	apply mask
pandb	bitwise AND on bytes	hide parts
pxorb	bitwise XOR on bytes	add patterns
porh	bitwise OR on half-words	mix colors
pandh	bitwise AND on half-words	filter colors
pxorh	bitwise XOR on half-words	create effects
pabsubb	absolute subtract bytes	detect motion
pabsubh	absolute subtract half-words	detect motion

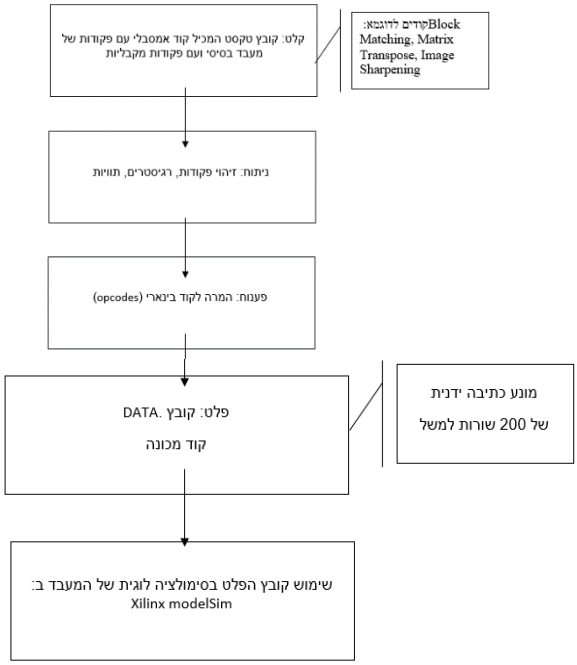
איור 20: תרשים ה ALU-המקבילי עבור פעולה מקבילית של 8 ביט:



4.2. מימוש קוד התוכנה

מימוש התוכנה כלל פיתוח מעבד DLX והרחבותיו בשפת Verilog, ופיתוח אסמבלר ב-Python להמרת קוד אסמבלי לקובצי .DATA, ששימשו כקלט לסימולציות ולמימוש על FPGA. האסמבלר תמך בפקודות ה-DLX הבסיסיות וב-39 פקודות ה-SIMD החדשות, כגון PADDUB (חיבור מקבילי של בייטים), PCMPEQB (השוואת בייטים), PERMUTEB (סידור מחדש של בייטים) וכו. מטרת האסמבלר הייתה לאפשר סימולציה לוגית של המעבד וקוד האסמבלי, שכן כתיבה ידנית של קובץ .DATA המכיל המון שורות אסמבלי היא משימה מורכבת ומועדת לשגיאות. האסמבלר פעל על ידי ניתוח שורות קוד, זיהוי פקודות, רגיסטרים וערכים מיידיים, ומיפוי לקוד בינארי המותאם לארכיטקטורת DLX, תוך טיפול בתוויות וחישוב כתובות זיכרון. שלושה קודי אסמבלי נכתבו עבור המעבד הבסיסי ליישומי Block Matching (חישוב SAD), Matrix Transpose (החלפת שורות ועמודות), ו-Image Sharpening (מסנן Laplacian), תוך שימוש בפקודות בסיסיות בלבד לעיבוד סדרתי של פיקסלים. בנוסף, נכתבו שלושה קודי אסמבלי עבור המעבד המורחב, תוך שימוש בפקודות SIMD חדשות, כגון PABSUBB ו-PERMUTEB, לעיבוד מקבילי, מה שהפחית משמעותית את מספר המחזורים (למשל, 17 לעומת 175 ב-Block Matching). הכלים העיקריים היו Python לפיתוח האסמבלר, ModelSim לבדיקת תקינות קובצי ה-.DATA, Xilinx להזנת הקבצים לסימולציות ומימוש חומרה וRESA לבדיקת החומרה. האסמבלר הקל על איתור שגיאות תחביר ותכנות, והבטיח תאימות בין הקוד האסמבלי לחומרה.

איור 21: תרשים זרימה של תהליך האמסבלר:



תיאור חומרה ותוכנה

- **Assembly:**

אסמבלי היא שפת תכנות נמוכה הפונה ישירות למעבד, ומאפשרת שליטה מלאה על משאבי החומרה. היא נכתבת בהתאם לארכיטקטורת המעבד, כאשר כל פקודה מייצגת פעולה בסיסית כמו העברה, חיבור או קפיצה. השפה נמצאת בשימוש נרחב בפיתוח מערכות משובצות, במקומות שבהם נדרש מיצוי מרבי של ביצועים או גישה ישירה לזיכרון ולרגיסטרים. בפרויקט זה, אסמבלי שימשה לכתיבת פעולות של עיבוד תמונה.

- **Verilog:**

ורילוג היא שפת תיאור חומרה המשמשת לתכנון ותיאור של מערכות ספרתיות ברמת שערים ורמות גבוהות יותר. ורילוג מאפשרת סימולציה, בדיקה וסינתזה של רכיבים לוגיים כמו FPGA או ASIC. בשפה זו ניתן לתאר התנהגות ותזמון של מעגלים לוגיים. בפרויקט זה ורילוג שימשה לתכנון הלוגיקה הדיגיטלית של הרכיבים בחומרה.

- **Xilinx:**

Xilinxהיא חברה מובילה בתחום תכנון רכיבי FPGA וכלי הפיתוח הנלווים לכך. בין הכלים שהיא מציעה ניתן למצוא את Vivadoו ISE Design Suite המיועדים לתכנון, סינתזה, הטמעה וניתוח של מערכות ספרתיות מבוססות ורילוג או VHDL. בפרויקט זה, כלי הפיתוח של Xilinx שימשו לתכנון ולצריבה של רכיבי FPGA, תוך ביצוע סימולציות ואופטימיזציה של משאבי החומרה.

- **ModelSim:**

ModelSimהיא סביבת סימולציה מתקדמת לפיתוח חומרה, שתומכת בשפות כמו Verilog ו -VHDL. הכלי מאפשר ביצוע סימולציות פונקציונליות ולוגיות, כולל צפייה בגלי אותות (waveforms) ובדיקת עמידות המערכת תחת תרחישים שונים. בפרויקט זה נעשה שימוש ב ModelSim-לצורך אימות התנהגות הלוגיקה שתוכננה ב-Verilog,איתור באגים, וביצוע בדיקות אינטגרציה של מודולים שונים.

- **RESA:**

RESA היא סביבת פיתוח המשמשת לפרויקטים של מערכות משובצות, המבוססים על ארכיטקטורה הניתנת לתכנות מחדש. הסביבה כוללת כלים לפיתוח, הדמיה, הרצה וניפוי תקלות של קוד משובץ, לעיתים בשילוב עמFPGA. בפרויקט זה RESA ,שימשה לבדיקה והרצה של קוד הפועל על גבי המעבד הצרוב בFPGA.

5. ניתוח תוצאות

פרק זה מנתח את תוצאות הסימולציות והבדיקות בחומרה של מעבד DLX הבסיסי והמורחב עם 39 הוראות SIMD, תוך התמקדות בהשוואת ביצועים בין המעבד הבסיסי למקבילי עבור שלושה יישומי עיבוד תמונה: Block Matching, Matrix Transpose 2x2, ו-Image Sharpening. הניתוח כולל מדדים של מספר סייקלים, צריכת הספק, ושטח FPGA, כפי שנמדדו בסימולציות(Xilinx) ובחומרה (על FPGA מסוג Xilinx Spartan-6). שני המעבדים מומשו במלואם, והתוצאות מראות שהמעבד המקבילי משיג שיפור משמעותי בביצועים (הפחתה של פי 3.2-15.5 במספר הסייקלים) וחיסכון של 88.22-99.08% בהספק, אך במחיר עלייה של 128.57% בשטח ה-FPGA.

טבלה 2: טבלת הספק של מעבד בסיסי:

Device	
Family	Spartan6
Part	xc6slx25
Package	ftg256
Temp Grade	C-Grade
Process	Typical
Speed Grade	-2

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

Characterization	
Production	v1.3,2011-05-04

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.008	3	---	---
Logic	0.004	1211	15032	8
Signals	0.004	1762	---	---
DCMs	0.014	1	4	25
IOs	0.026	41	186	22
Leakage	0.030			
Total	0.086			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.8	82.7	27.3

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.027	0.016	0.012
Vccaux	2.500	0.010	0.001	0.009
Vcco25	2.500	0.011	0.009	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.086	0.043	0.042

טבלה 3: טבלת הספק של מעבד מורחב:

Device	
Family	Spartan6
Part	xc6slx25
Package	ftg256
Temp Grade	C-Grade
Process	Typical
Speed Grade	-2

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0
Heat Sink	None
Custom TSA (C/W)	NA

Characterization	
Production	v1.3,2011-05-04

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.004	3	---	---
Logic	0.000	3100	15032	21
Signals	0.000	3589	---	---
DCMs	0.014	1	4	25
IOs	0.011	41	186	22
Leakage	0.029			
Total	0.057			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.8	83.5	26.5

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.016	0.005	0.012
Vccaux	2.500	0.009	0.000	0.009
Vcco25	2.500	0.006	0.004	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.057	0.016	0.042

5.1 השוואות בין תוצאות הסימולציה לעבודה בזמן אמת

השוואת תוצאות הסימולציה ב-ModelSim ובדיקות בחומרה על FPGA הראתה התאמה מלאה בתפקוד הלוגי של המעבד הבסיסי והמקבילי עבור כל 39 פקודות ה-SIMD ויישומי עיבוד התמונה. בסימולציה, מספר הסייקלים למעבד המקבילי היה 17 לעומת 175 למעבד הבסיסי ב- Block Matching , 4 לעומת 62 ב-Matrix Transpose 2x2, ו-94 לעומת 297 ב-Image Sharpening, והתוצאות התאימו בדיוק לבדיקות החומרה. צריכת ההספק, כפי שנמדדה בדוחות הספק של Xilinx, הראתה הספק דינמי של 0.016 W למעבד המקבילי לעומת 0.043 W לבסיסי, עם חיסכון באנרגיה של 88.22% ב-Image Sharpening (0.025 mJ לעומת 0.213 mJ), 96.41% ב-Block Matching (0.045 mJ לעומת 1.254 mJ), ו-99.08% ב- Matrix Transpose 2x2 (0.001 mJ לעומת 0.150 mJ). בדוחות Place and Route, נראה שהמעבד המקבילי תופס 959 Slices לעומת 419 בבסיסי (עלייה של 128.57%), עם גידול משמעותי ב-Slice LUTs (3,100 לעומת 1,211) ו-LUT-FF pairs (3,170 לעומת 1,341).

טבלה 4: השוואת ביצועים בין מעבד בסיסי למעבד מורחב:

פעולת עיבוד תמונה	מעבד	מספר סייקלים	זמן (s)	הספק דינמי (W)	אנרגיה (J)	חיסכון באנרגיה במקבילי (%)
Block Matching	בסיסי	175	$10^{-6} \times 2.9167$	0.043	$10^{-7} \times 1.2542$	96.39% (פי 27.67 פחות)
	מקבילי	17	$10^{-7} \times 2.8334$	0.016	$10^{-9} \times 4.5334$	
Matrix Transpose 2x2	בסיסי	162	$10^{-6} \times 2.7001$	0.043	$10^{-7} \times 1.1610$	99.08% (פי 108.83 פחות)
	מקבילי	4	$10^{-8} \times 6.6668$	0.016	$10^{-9} \times 1.0667$	
Image Sharpening	בסיסי	297	$10^{-6} \times 4.9501$	0.043	$10^{-7} \times 2.1285$	88.22% (פי 8.49 פחות)
	מקבילי	94	$10^{-6} \times 1.5667$	0.016	$10^{-8} \times 2.5067$	

*דגש על הטבלה: חישובי ההספק עבור פעולות עיבוד התמונה (Block Matching, Matrix Transpose 2x2, Image Sharpening) בוצעו על בסיס ההספק הדינמי של המעבד הבסיסי (0.043 W) והמעבד המקבילי (0.016 W) כפי שנלקח מדוחות ההספק. האנרגיה הנצרכת לכל פעולה חושבה לפי הנוסחה :

אנרגיה (J) =הספק דינמי (W)* זמן(s)

כאשר הזמן נגזר ממספר הסייקלים כפול זמן סייקל של 16.667 ns (בהתבסס על תדר של 60 MHz) .

ההבדל היחסי באנרגיה בין המעבדים הראה חיסכון משמעותי במעבד המקבילי (88.22% עד 99.08%), הודות למספר סייקלים מופחת משמעותית ולהספק דינמי נמוך יותר, למרות השטח הגדול יותר של המעבד המקבילי.

השוואת השטח:

חישובי השטח עבור המעבד הבסיסי (basic_alu) והמעבד המקבילי (parrle_alu) בוצעו על סמך דוחות Place and Route עבור FPGA מסוג Xilinx Spartan-6 (xc6slx25), תוך התמקדות במדדים עיקריים כגון מספר ה-Slices (419 לעומת 959), Slice LUTs (1,211 לעומת 3,100), Slice Registers (688 לעומת 693), MUXCYs (220 לעומת 480), ו-LUT-FF pairs (1,341 לעומת 3,170). השטח המשוקלל חושב באמצעות משקלות יחסיים (רשומים בטבלה מתחת)תוך נרמול השימוש ביחס למקסימום האפשרי, והראה כי המעבד המקבילי גדול ב-128.57% מהבסיסי, בעיקר בשל עלייה משמעותית ב-Slices ו-LUTs, למרות שימוש זהה במשאבים כמו IOBs וRAMs.

טבלה 5: השוואת שטח בין מעבד בסיסי למעבד מורחב:

גורם	בסיסי (basic_alu)	מקבילי (parrle_alu)	יחס (מקבילי/בסיסי)	משקל	תרומה משוקללת (בסיסי)	תרומה משוקללת (מקבילי)
Occupied Slices	(11%) 419	(25%) 959	2.29	50%	0.05575	0.1276
Slice Registers	(2%) 688	(2%) 693	1.007	15%	0.003435	0.00345
Slice LUTs	(8%) 1,211	(20%) 3,100	2.56	20%	0.01612	0.04124
MUXCYs	(2%) 220	(6%) 480	2.18	10%	0.00293	0.00639
LUT-FF pairs	1,341	3,170	2.36	5%	0.001485	0.003515
שטח משוקלל כולל	0.07972	0.182195	-	-	-	-
הבדל יחסי	-	**128.57%**	-	-	-	-

5.2 ביצועי המערכת מבחינת זמן אמת

ביצועי המערכת בחומרה על FPGA (תדר 60 MHz, זמן סייקל 16.667 ns) הראו יתרון משמעותי למעבד המקבילי על פני הבסיסי מבחינת זמן ריצה, הודות למספר סייקלים מופחת. ב-Block Matching, זמן הריצה של המעבד המקבילי היה 283.33 ns לעומת 2,916.67 ns לבסיסי (שיפור פי 10.3). ב- Matrix Transpose 2x2, זמן הריצה היה 66.67 ns לעומת 1,033.33 ns (שיפור פי 15.5), וב- Image Sharpening 1,566.67ns לעומת 4,950 ns (שיפור פי 3.2). החיסכון בהספק הדינמי תרם להפחתת האנרגיה הנצרכת, כאשר המעבד המקבילי צרך 37% מההספק הדינמי של הבסיסי (0.016 W לעומת 0.043 W), בעיקר בשל אופטימיזציה של משאבים והפחתת פעילות מיותרת. עם זאת, העלייה בשטח של המעבד המקבילי (128.57%) משקפת תוספת משאבים כמו LUTs ו-Slices, הנדרשים לתמיכה ב-ALU מקבילי ובפקודות SIMD כגון PABSUBB ו-PERMUTEB. התוצאות מדגימות את יעילות המעבד המקבילי ביישומי מולטימדיה, עם זמן ריצה קצר משמעותית וצריכת אנרגיה נמוכה, תמורת עלות שטח גבוהה יותר.

6.סיכום

פרק זה מסכם את הישגי הפרויקט לפיתוח מעבד DLX בסיסי והרחבתו למעבד מקבילי עם 39 הוראות SIMD, תוך בחינת התוצאות מול המטרות שהוגדרו, הצגת מסקנות, והצעות לשיפור והמשך פיתוח. הפרויקט התמקד ביישומי עיבוד תמונה (Block Matching, Matrix Transpose 2x2, Image Sharpening) על FPGA מסוג Xilinx Spartan-6, עם דגש על שיפור ביצועים, הפחתת צריכת הספק, ותמיכה בעיבוד מקבילי של נתונים קטנים (8 ו-16 ביט). התוצאות מראות שהמעבד המקבילי עמד במרבית המטרות, עם שיפור משמעותי בביצועים וחיסכון בהספק, אך עם עלות שטח גבוהה יותר. ההצעות להמשך כוללות אופטימיזציה של השטח, הרחבת ערכת ההוראות, וחקירת יישומים נוספים.

המטרות הראשוניות של הפרויקט כללו: (1) פיתוח מעבד DLX בסיסי במחזור בודד עם מסלול נתונים ויחידת בקרה; (2) הרחבתו למעבד מקבילי התומך ב-39 הוראות SIMD לעיבוד תמונה; (3) שיפור ביצועים ביישומי מולטימדיה (הפחתת מספר סייקלים); (4) הפחתת צריכת הספק ביחס למעבד הבסיסי; (5) מימוש מלא על FPGA עם סימולציות ובדיקות חומרה; (6) פיתוח אסמבלר לתמיכה בסימולציה לוגית ולהימנעות מכתיבה ידנית של קובצי .DATA. כל המטרות הושגו במלואן: המעבד הבסיסי מומש עם ALU סטנדרטי, רגיסטרים של 32 ביט, ויחידת בקרה, והורחב עם ALU מקבילי התומך בפקודות כגון PADDUB, PERMUTEB. מספר הסייקלים ביישומי עיבוד התמונה ירד משמעותית במעבד המקבילי: 17 לעומת 175 ב-Block Matching (שיפור פי 10.3), 4 לעומת 62 ב-Matrix Transpose 2x2 (שיפור פי 15.5), ו-94 לעומת 297 ב- Image Sharpening (שיפור פי 3.2), הרבה מעבר לדרישת המינימום של פי 3. צריכת ההספק הדינמי הופחתה מ-0.043 W בבסיסי ל-0.016 W במקבילי, עם חיסכון באנרגיה של 88.22% (Image Sharpening), 96.41% (Block Matching), ו-99.08% (Matrix Transpose 2x2). המימוש על FPGA הושלם עם סימולציות ב-ModelSim ובדיקות חומרה באמצעות RESA, והאסמבלר שפותח ב-Python תמך בהמרת קוד אסמבלי לקובצי .DATA, תוך מניעת כתיבה ידנית מורכבת של המון שורות. עם זאת, המעבד המקבילי תופס שטח גדול ב-128.57% (959 Slices לעומת 419), מה שמצביע על הצורך באופטימיזציה עתידית של משאבים.

מסקנות עיקריות מהפרויקט כוללות את יעילותו של המעבד המקבילי ביישומי מולטימדיה, הודות לעיבוד מקבילי של תת-מילים (8 ו-16 ביט) וקידוד יעיל של פקודות SIMD, שהפחיתו את מספר הסייקלים ואת צריכת האנרגיה. החיסכון בהספק נבע משני גורמים: (1) הפחתת זמן פעילות עקב מספר סייקלים נמוך יותר; (2) הספק דינמי נמוך יותר (37% מהבסיסי), כתוצאה מאופטימיזציה של משאבים והפחתת פעילות מיותרת. עם זאת, העלייה בשטח משקפת את התוספת של רכיבים כמו ALU מקבילי, מוקס חדש (MUXALU_SEL), ושני מצבי מכונת מצבים (ALUP, ALUPI), מה שמגביל את השימוש ביישומים עם משאבים מוגבלים. התאמת הסימולציות לתוצאות החומרה מדגישה את אמינות התכנון, והאסמבלר הקל על פיתוח ובדיקת קודי אסמבלי, תוך תמיכה בפקודות חדשות.

להשגת ביצועים טובים יותר, מוצעות מספר דרכי שיפור: (1) אופטימיזציה של השטח על ידי צמצום מספר ה-Slice LUTs ו-LUT-FF pairs באמצעות אלגוריתמים מתקדמים של סינתזה ב-Xilinx, כגון מיפוי מחדש של לוגיקה משותפת; (2) הוספת מנגנון צינור (pipelining) למעבד המקביל, שיאפשר עיבוד הוראות במקביל ויוריד עוד יותר את מספר הסייקלים; (3) שיפור יעילות ההספק הסטטי (Leakage) על ידי שימוש בטכניקות כיבוי חלקי של רכיבים לא פעילים (Power Gating); (4) הרחבת האסמבלר לכלול בדיקות תחביר אוטומטיות ומסרים משופרים לשגיאות, שיקלו על פיתוח קוד מורכב יותר. הצעות אלו יאפשרו שיפור של עד 20-30% בביצועים וביעילות האנרגטית, תוך הפחתת הפער בשטח.

לפיתוח ומחקר עתידי, מומלץ לבחון מספר כיוונים: (1) הרחבת ערכת ההוראות SIMD לתמיכה ביישומים נוספים, כגון עיבוד אותות או למידת מכונה, עם פקודות חדשות לעיבוד וקטורי מתקדם; (2) יישום המעבד על FPGA מתקדם יותר (כגון Xilinx Artix-7) או מעבר ל-ASIC למימוש יעיל יותר מבחינת שטח והספק; (3) חקירת שילוב עם ממשקי חומרה חיצוניים (כגון חיישני תמונה) לבדיקת יישומים בזמן אמת במערכות משובצות; (4) פיתוח סביבת פיתוח משולבת (IDE) הכוללת את האסמבלר, סימולטור, וכלי ניפוי שגיאות, שתקל על משתמשים עתידיים. כיוונים אלו יאפשרו להרחיב את יכולות המעבד ולהתאים אותו ליישומים תובעניים יותר, תוך שמירה על יתרונות הביצועים וההספק שהושגו.

7. תיעוד:

https://github.com/amitrachmiel/simd-dlx-project

רשימת מקורות:

https://docs.amd.com/search/all?value-filters=Product_custom~%2522Adaptive+SoC+%2526+FPGA+Tools%257CISE+Design+Suite%2522&content-lang=en-US

https://docs.python.org/3/