דו״ח סיכום לפרויקט גמר בקורס ארכיטקטורות מחשבים מתקדמות

פרויקט 9 – RiscV with systolic array accelerator

שמות הסטודנטים ות.ז:

דור בר שי 205661739  
שחר כדורי 205442544

# מבוא

* **תיאור הבעיה ורקע כללי**

מעבד risc-v היא ארכטיקטורה בעלת סט פקודות סטנדרטי open-source platform

במעבד risc-v כפל המטריצות לוקח הרבה זמן מכיוון שחומרת המעבד מוגבלת בכמות רכיבי הALU ופעולת כפל המטריצות דורשת מס' פעולות רב מרכיבים אלה. לשם כך מעוניינים לבנות רכיב שיאפשר לעשות זאת ביעילות רבה יותר ככל האפשר על מנת להאיץ את פעולת כפל המטריצות.  
ישנו רכיב שיודע לעשות זאת ביעילות והוא מערך סיסטולי, מערך זה יודע לבצע כפל מטריצות ע"י רכיבי DPUs המחוברים זה לזה באופן מטריציוני כך שמתקבלת מטריצה של רכיבים אלו. כל רכיב שומר בתוכו סכום של תוצאות המכפלה ומעביר הלאה את גורמי הכפל של שתי המטריצות הן להמשך השורות של ה DPU והן לעמודות. בסיום הפעולה כל "תא" DPU מחזיק בתוכו את הסכום המהווה את התוצאה של התא הרלוונטי במטריצת התוצאה.

* **מטרות**

מטרת הפרויקט היא לשלב את המערך הסיסטולי כחלק ממערך החישוב של המעבד risc-v

* **הדרך והמצעים להשגת המטרה**

מימוש המערך יעשה בשפת חומרה – ורילוג.

המימוש יעשה בשלבי פיתוח שונים על מנת לחלק את הבעיה לבעיות קטנות.

בעצם הגדרת מטרות קטנות בדרך להשגת מטרה גדולה (בניית המערך הסיסטולי)

בסוף תכנון ומימוש הבעיות הקטנות נצטרך לבצע אינטגרציה של כל החלקים השונים.

# תיאור הפתרון

* תיאור **מפורט** של הפתרון שבחרתם למימוש הפרויקט:
* יצירת רכיב המערך הסיסטולי:

כידוע לנו רכיב המערך הסיסטולי נועד לקבל בכניסותיו את איבר המטריצות לפי הסדר המיוצג להם וחישובם בצורה אשר תיארנו ברקע הכללי.  
לשם כך, אנו נדרשים להציג את התת-יחידות של המערך עצמו, יחידות הdpu, יחידת הבקרה אשר מתזמנת, מאפשרת ומפסיקה את תהליך החישוב של המערך הסיסטולי באמצעות תקשורת למול המעבד, יחידת סידור אשר מקבלת את הרגיסטרים של מטריצות הכניסה ומסדרות אותן ע"ב צורת החישוב של המערך ויחידת mux אשר מעבירה הלאה את השורה לפי הסדר ע"ב מס' סיבוב השעון הנוכחי. נפרט כעת על הרכיבים:

* יחידת dpu:

כניסות:  
data\_in\_a, data\_in\_b – שהם האיברים של המטריצות a ו-b שאיתם הרכיב מחשב את האיבר במטריצת המכפלה אשר נדרש למצוא.

Enable – אפשור ליחידה לבצע את תהליך החישוב.  
clk – כניסת שעון.  
rst\_n – כניסת ריסט אסינכרוני.  
יציאות:  
data\_out\_sum – התוצאה הסופית המתקבלת לאיבר המחושב, מועברת הלאה לרכיב הtop.  
data\_out\_a, data\_out\_b – האיברים אשר נשלחים הלאה ליחידות הdpu העוקבות במערך בהתאמה ע"ב צורת החישוב של המערך הסיסטולי (a לרכיב מימינו, b לרכיב מתחתיו)

* יחידת הבקרה:

יחידה זו נועדה לאפשר תהליך חישוב כפל המטריצות באמצעות המערך הסיסטולי ולהודיע על סיומו.

כניסות:  
clk – כניסת שעון.  
rst\_n – כניסת ריסט אסינכרוני.  
go\_bit – רגיסטר איפשור המתקבל מהמעבד לביצוע תהליך

יציאות:

sys\_arr\_ena – איפשור הנשלח היחידת הבקרה למערך הסיסטולי וליחידת הmux לצורך אפשור תהליך החישוב של המערך הסיסטולי.  
clk\_count – מונה סיבובי שעון אשר נועד לתזמן את שליחת הdone\_bit והעברת הכניסה לפי הסדר בתור sel ביחידת הmux.

Done\_bit – נועד להודיע על תהליך סיום החישוב של המערך הסיסטולי, נשלח למערך עצמו ולרכיב הtop.

* יחידת המערך הסיסטולי:  
  המערך מרוכב למעשה מ-16 יחידות dpu אשר נועדו לחשב כל איבר במטריצת המכפלה.

כניסות:

RowA,rowB – כניסות אלה מכילות 4 מספרים המסודרים כניסות עבור כל שורה של מטריצה A ועבור כל עמודה של מטריצה B, כל מספר בשורות אלה נכנסות לרכיבי הdpu בהתאמה: rowA מכניס ליחידות dpu 1,4,7,10 וrowB מכניס ליחידות dpu 1,2,3,4.  
systolic\_clk – כניסת שעון.  
rst\_n – כניסת ריסט אסינכרוני  
enable – אפשור לתהליך החישוב אשר מתקבל מהבקרה, אפשור זה נשלח הלאה לרכיבי הdpu.  
done – כניסת סיום התהליך מיחידת הבקרה, נועד לעדכן את התוצאה הסופית ברגיסטרים של היציאה.

יציאות:  
rowC0-3 – 4 רגיסטרים של יציאה באורך 32 ביטים אשר כוללים את שורות מטריצת המכפלה, אשר מחוברים ליציאות רכיבי הdpu בהתאמה.  
חיבורים:  
רכיבי הdpu מחוברים כל אחד לרכיב שמימינו ומתחתיו מלבד לרכיבי הקצה.

* יחידת mux עבור הכניסות למערך:  
  מתואר בתור רכיב mux10to1.  
  כניסות:  
  i0-9 – כל כניסה מהווה שורת קלט עבור המערך הסיסטולי בהתאם לסדר לפי מס' סיבובי השעון בתהליך החישוב.  
  sel – מעבירה את השורה הנוכחית לפי המונה של יחידת הבקרה.  
  ena – אפשור מיחידת הבקרה להעביר שורות הלאה כחלק מתהליך החישוב.  
  יציאות:  
  out – שורת היציאה.
* יחידת סידור הכניסות בהתאם לערכי המטריצות:  
  כניסות:  
  i0-15 – כל כניסה מהווה איבר במטריצה אשר מקבלת מרכיב הtop אשר מקבל מהמעבד  
  יציאות:  
  out0-9 – יציאה זו מסדרת את האיברים כשורות של 4 איברים עבור הסדר שלהם בכל סיבוב שעון ע"ב צורת החישוב של המערך הסיסטולי.
* יחידת הtop:

יחידת הtop מורכב מרכיב המערך הסיסטולי, יחידת הבקרה, 2 יחידות סידור שורות עבור כל מטריצה ו-2 יחידות mux עבור כל מטריצה.  
כניסות:  
clk – כניסת שעון.  
rst\_n – כניסת ריסט אסינכרוני.  
rowA0-3, colB0-3 – קלטים באורך 32 ביטים המתארים את שורות המטריצה A ועמודות המטריצה B (8 ביטים לכל איבר), נשלחים ע"י המעבד ומוכנסים אל יחידות סידור השורות של המטריצות בהתאמה.  
go\_bit – קלט האפשור המתקבל מהמעבד לצורך יישום פעולת חישוב כפל המטריצות, מחובר אל יחידת הבקרה.  
יציאות:

rowC0-3– פלטים (רגיסטרים היציאה) באורך 32 ביטים המתארים את שורות מטריצת המכפלה (8 ביטים לאיבר), מתקבלים מרכיב המערך הסיסטולי ונשלחים אל המעבד.  
done\_bit – רגיסטר הסיום והמועבר מיחידת הבקרה אל המעבד.  
  
חיבורים:  
arr\_ena – קישור של האפשור מיחידת הבקרה עם יחידת המערך הסיסטולי ויחידת הmux.  
rowA0-9,rowB0-9 – השורות המסודרות ע"י יחידת הסידור מחוברות עם הכניסות ליחידת הmux.  
clk\_count – קישור המונה ליחידת הmux בכניסת sel.  
acc\_rowA,acc\_rowB – קישור היציאה מיחידות הmux אל המערך הסיסטולי

* ניהול קשר בין הרכיב הtop למעבד:

אנחנו ביססנו את קישור הרכיב עם המעבד בתלות לתוכנית בשפת c אשר מתארת כפל מטריצות, מקומפלת ומתורגמת לפקודות MIPS ונצרבת על הרכיב.  
הגדרנו כתובות ספציפיות עבור המעבד אשר יגדירו את כתובות המצביעים של כל איבר במטריצות הכניסה והיציאה.  
16 רגיסטרים יהיו מקושרים עבור כל מטריצת כניסה ויאכסנו את איברי המטריצות כחלק מתהליך האיתחול של המטריצות. רגיסטרים אלה מקושרים בתור קלט אל הרכיב.  
שליחת הgo\_bit לאחר האתחול נשלחת מהמעבד לאחר זיהוי פקודת \_\_\_ אשר נקלטת במעבד מהתוכנית הצרובה. רגיסטר זה ישלח לרכיב בtpu כדי להתחיל את תהליך החישוב כפל המטריצות. יהיו 16 רגיסטרים אשר יאכסנו את ערכי מטריצת היציאה אשר התקבלו מרכיב הtpu ויהיה רגיסטר done\_bit אשר יקלוט מרכיב הtpu את סיום התהליך.

# אמצעי המחקר

* תיאור התשתית:
* הרכיבים נכתבו באמצעות שפת ורילוג, השתמשו בכלי סימולציה בשם modelsim והצריבה על המעבד באמצעות הquartus.
* שיטת ניתוח התוצאות:
* כחלק מתהליך הצריבה, אנחנו מכינים תוכנית בשפת c אשר מתארת כפל מטריצות, מקמפלים ומתרגמים אותה לפקודות MIPS וקובץ זה נקרא ע"י המעבד.  
  אנחנו נבצע זאת עבור תוכניות שונות ונציג את התוצאות על המעבד.
* הצג של המעבד יציג את מס' סיבובי השעון הנדרשו מרגע כתיבת go-bit לרכיב הבקרה לרגע של קריאת done-bit מרכיב הבקרה, המעיד את יעילות המאיץ.
* הצגת התוצאה של מטריצת המכפלה על הצג.

# תיאור וניתוח התוצאות הניסוייות

* תיאור התוצאות הניסוייות:

את ניתוח התוצאות של תהליך תפקוד הרכיב בדקנו עבור כמה מקרים:

* סימולציה של הרכיב עצמו בmodelsim עם שורות ועמודות שהגדרנו בסימולציה.
* צריבה של הרכיב על הFPGA ללא התחשבות במעבד והגדרת ערכים מראש.

בצריבה זו, הגדרנו ערכים לתוך הכניסות של רכיב הtpu בקוד ושלחנו go bit בעת לחיצת כפתור. לאחר קבלת done bit, המספרים מוצגים על המסך.

* צריבה של הרכיב על הFPGA עם קישור למעבד בהתחשבות לתוכניות C.

התוכניות מתוכננות מראש כאשר הכתובות של איברי המטריצה A ו-B מוגדרות מראש, האיתחול של האיברים מתבצע לאחר כתיבה לתוך המצביעים בעלי ערכי הכתובות, ושליחת הgo bit בde0 נשלח בעת כתיבה לתוך כתובת איברי המטריצה C, מאחר ותהליך כפל המטריצות דורש כתיבה לכתובת המצביע לאיברי המטריצה

* ניתוח התוצאות והסקת מסקנות
* עבור הסימולציות הבחנו ווידאנו כי הרכיב עובד כראוי ואכן מחשב כפל מטריצות עבור 11 סיבובי שעון מרגע שליחת הgo bit ועד קבלת done bit. קיים עיכוב של סיבוב שעון אחד מאחר והאפשור מתעכב בcontroller למשך סיבוב שעון אחד.
* כאשר צרבנו את הרכיב על הFPGA ללא התחשבות במעבד, הוצגו על הלדים המספרים לפי הסדר לאחר קבלת done bit. כמו בסימולציה, קיים עיכוב של סיבוב שעון אחד.
* ביססנו את התוכניות כאשר המשתנים מכילים בתוכם את הערכים שונים, לאחר קבלת done bit, האיברים הוצגו על הלדים לאחר החישוב בהצלחה.

# סיכום

* סיכום הפרויקט.

לסיכום, ניתן לראות את חשיבות רכיב המערך הסיסטולי אשר מאיצה את תהליך חישוב כפל המטריצות וחוסכת בזמן חישוב בתלות לגודל המטריצה. מניתוח התוצאות ניתן לראות כי לוקח למאיץ 10 סיבובי שעון עבור כפל מטריצות בגודל 4X4 .

* סיכום של התרומה של העבודה שביצעתם

הפרויקט עזר לנו מאוד להבין את צורת התפעול של מעבד risc-V וצורת התפעול של רכיב המערך הסיסטולי במקביל עם המעבד. לא יצא לנו לעבוד בשפת ורילוג בעבר, שזה הקשה עלינו, אבל באותה מידה הפרויקט עזר לנו ללמוד ולעבוד עם השפה.

* חשיבה ביקורתית:
  + מה היה טוב במימוש:

הצלחנו לממש את פעולות המערך הסיסטולי בצורה טובה, הוא עושה את חישוב כפל המטריצות כמצופה ובחיסכון בזמן החישוב מאשר כפל מטריצות בלעדיו.

* + מה ניתן לשפר:
    - התבססנו על מס' לא קטן של רכיבים, ייתכן וניתן לממש את הרכיב בעלות קטנה יותר ברכיבים.
    - התהליך עובד עבור תוכניות שהגדרנו מראש כאשר הכתובות של איברי המטריצות בזיכרון ידועות לנו מראש, אפשר לשפר כך שהתהליך לעבוד בלי תלות בכתובת.
    - כתיבה לזיכרון במקרה שלנו לא מומשה, נדרש ליישם כתיבה של ערכי האיברים לתוך הכתובות.