# Digital Design & Logic Synthesis

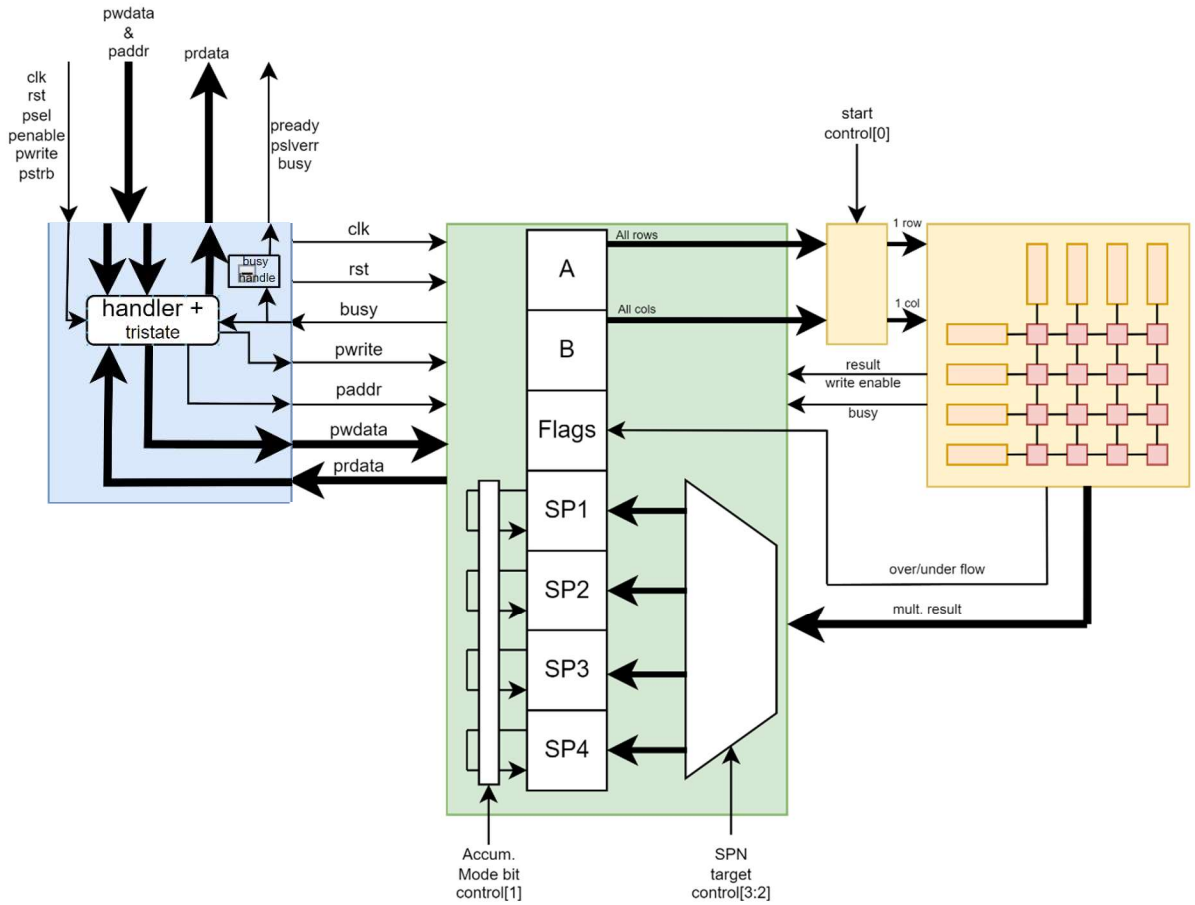
# Course Project: Matrix Multiplication


**מגישים:**

**יבגני יגודין 324432988**

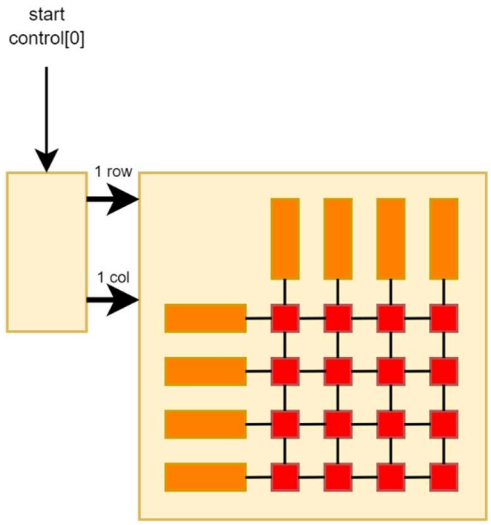
**ליאור פנקר 316506278**

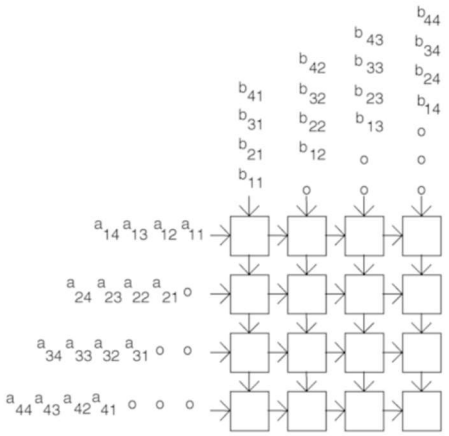

המשימה שקיבלנו היא לממש מכפל מטריצות.
המכפל כמובן לא עובד מעצמו ויש לתת לו זיכרון צמוד וגם גישה מסודרת לזיכרון הזה מעורק הראשי.
מה-bus שולטים ומכניסים מידע ל-APB slave, ממנו פונים לזיכרון לקריאה או למסירת נתונים שיחושבו במודול שמכפיל את המטריצות.
כאמור, כל המערכת ניתן לפרק ל-3 בלוקים עיקריים. נציג תחילה את הדיאגרמה הכללית של כל הדיזיין, ולאחר מכן נעבור על כל אחד משלושת הבלוקים העיקריים.



## בלוק המכפל HALF_MUL:

הבלוק הצהוב בשרטוט (יחד עם בלוק עזר קטן צהוב לידו) הוא בלוק שמבצע את הפעולה של הכפלת המטריצות עצמה. המימוש הוא בצורה סיסטולית ככה שכל איבר בווקטור מתפשט דרך כל הרכיבים בתזמון כזה שתואם את האלגוריתם של הכפלת מטריצות.

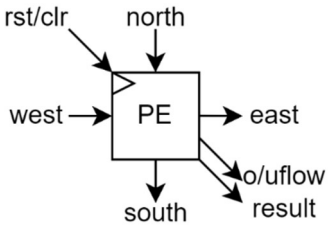



נתאר תחילה כל תת-בלוק של המכפל בנפרד ונתאר איך הכל עובד ואז נחזור אליו.

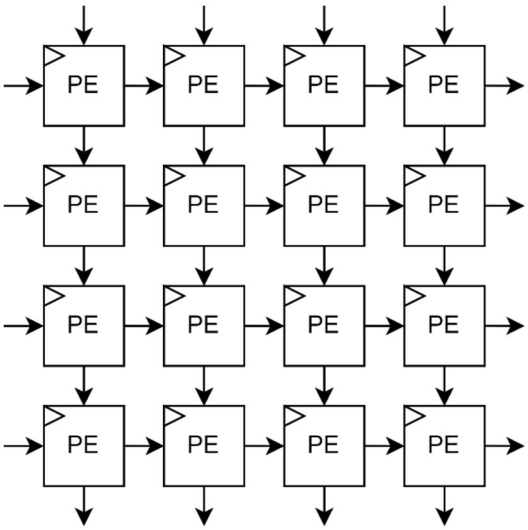
### מודול PE :

הבלוק הכי בסיסי שמקבל 2 קלטים (בציור הקלטים מלמעלה ומשמאל) , מחשב את המכפלה הסקלרית שלהם ומחבר לערך הצבור ממכפלה הקודמת.
את הקלטים, כמו שהם מקדם הלאה לבלוק PE הבא, אחד מימין ושני מלמטה: מה שנכנס מלמעלה – ממשיך למטה , ומה שנכנס משמאל – ממשיך ימינה, תנועה אחת לכל מחזור שעון.
היחידה פולטת את התוצאה המספרית שתואמת למיקום האיבר במטריצת התוצאה וביט של over/underflow .
קלט נוסף הוא כמובן שעון, איפוס וביט ניקוי שנדלק כאשר המכפל מסיים את העבודה ויש צורך לנקות את הערך הצבור. אם 2 איברי המכפלה באורך DATA אז המכפלה תהיה בגודל 2DATA , חיבור עם הערך הצבור שיכול לקרות עד 4 פעמים (אם מטריצה בגודל 4) זה גורם לאורך התוצאה להיות עד 2DATA+2 ביטים. במקרה כזה מוקדש לפלט רוחב BUS שלם שהוא פי 4 גדול מDATA. במקרה של מטריצה בגודל 2, רוחב BUS לא יספיק ונצטרך 2DATA+1 ביטים ולכן נשתמש בביט ה-OVF שישמר ברגיסטר FLAGS בקובץ הרגיסטרים.



**בלוק SYSTOLIC :**

הבלוק הזה משרשר את הבלוקים PE באופן כזה שיתאים לסדר ותזמון נכון של הכפלות סקלרים לפי אלגוריתם של הכפלת מטריצות. מיקום בלוק התוצאה יהיה תואם למיקומו במטריצה. בנוסף לזה, הבלוק סופר את כמות מחזורי השעון שלוקחת הפעולה שתלויה בגודל המטריצות ומוציא אות done שמאפס את ה-PE לפעולות עתידיות וגם מסמן לרגיסטר SP את write enable כדי לשמור את התוצאה רק לאחר שהפעולה הסתיימה ולא לפני הזמן.
כמות מחזורי שעון שייקח למכפל לסיים את העבודה יהיה כסכום של אורך שורה/עמודה מקסימלי + אורך האלכסון של מטריצת תוצאה + מספר קבוע קטן עבור תנועות ושמירה של התוצאה. בזמן הפעולה דולק סיגנל "started" שמסמל שהמכל עסוק באותו רגע, נשתמש באות הזה בהמשך.
הקלט של הבלוק הוא DIM איברים מ"צפון" + DIM איברים מ"מערב", שאותם דוחף לכיוון ה-PE והם כבר עושים חישובים ופרופורציה של איברים הלאה. את הצעד הראשון הוא יבצע כאשר יקבל "start_bit " שמחובר לרגיסטר הבקרה. הפלט הוא כל מטריצת התוצאה בצורות שטוחה שזורמת ישר לקובץ הרגיסטר המתאים, אותו דבר עם OVF שנוצרו בכל PE.




**בלוק PUSHER :**

מאחר והפעלה תקינה של המכפל דורשת תיזמון מדויק וריפוד באפסים (כפי שרואים בשרטוט בעמוד הקודם),

מימשנו רכיב עזר אשר מבצע זאת. (מלבן בשרטוט הצבעוני בעמוד קודם בצבע כתום).

הרכיב מזרים אפסים כדי לרפד ווקטורים/עמודות בהתחלה ובסוף בזמן שווקטור/שורה עובר דרך כל הרשת

הסיסטולית. קלט הבלוק הוא שורה/עמודה שלמה של מטריצת אופרנד וכל מחזור שעון פולט רק איבר אחד מתוך

השורה בסדר המתאים לאלגוריתם של המכפל.

לאחר שאיברים נגמרו, הרכיב מוציא אפסים כפי שנאמר מקודם.

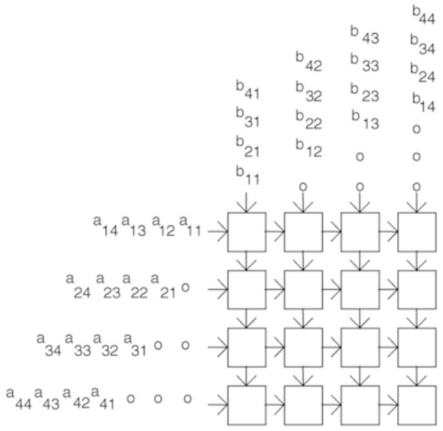
**בלוק HALF_MUL :**

מכיל את כל הבלוקים שהוזכרו עד עכשיו ומאחד את הפעולה של כולם, כלומר

מקבל שורות ועמודות של A ו-B בהתאמה בתזמון מתאים, זוג-זוג פר מחזור שעון. מזרים בתוכו את הווקטורים שמתפרקים לאיברים בודדים ומוזנים לתוך הרשת הסיסטולית. הפלט הוא כפלט הרשת הסיסטולית, כלומר מטריצת תוצאה בצורה שטוחה, מטריצת דגלים, אות busy ו-writenable לצורך שמירת התוצאה.

**בלוק flow_control :**

הבלוק הזה הוא בלוק עזר החוצץ בין רגיסטרים שמאחסנים את אופרנדים A , B לבין המכפל ומחובר עליו start_bit אשר נמצא ברגיסטר בקרה בביט ה-0 ( control[0] ). הביט הזה יתחיל את העברת השורות והעמודות בסדר נכון שיתאים לאלגוריתם הכפלה, כלומר שורה/עמודה 0 חייבת להיכנס ראשונה, במחזור שעון הבא חייב להיכנס שורה/עמודה1 וכך הלאה, כמו שניתן לראות בתמונה שראינו כבר :




הרכיב מחובר ישירות לזיכרון של B,A ולכניסה של מכפל, כלומר הקלטים שלו הם כל השורות של A וכל העמודות של B ,בעצם כל המטריצה בצורה שטוחה, הפלט שלו הוא זוג שורה+עמודה **בודד** מתוזמן שתואם לתזמון שמפעיל ביט הבקרה 0 (start_bit=control[0]).

## בלוק הזכרון MatrixRegisterFile :

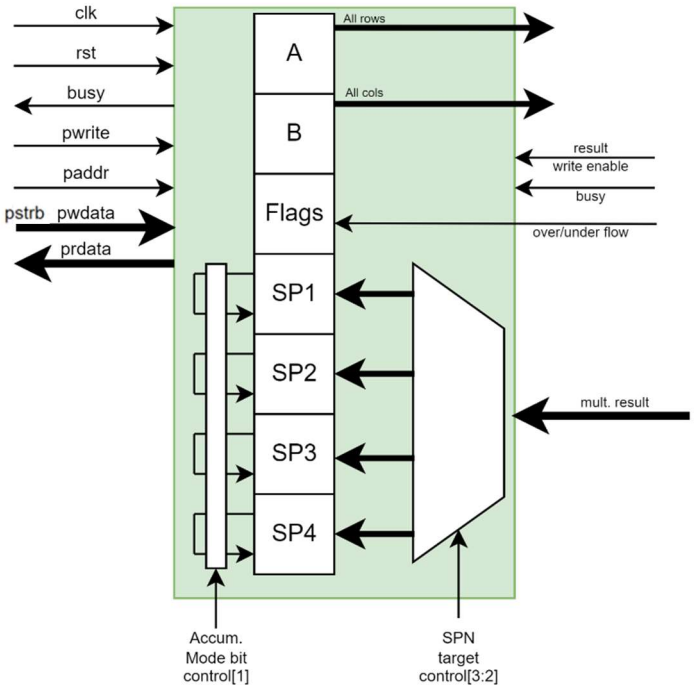
הבלוק של זיכרון שומר את מטריצות האופרנדים והתוצאות.

בנוסף לזה הוא מתקשר עם APB slave , מקבל ממנו מידע, כתובת וביט write enable ומאפשר לו כתיבה במקומות מותרים בלבד, לפי הכתובת.

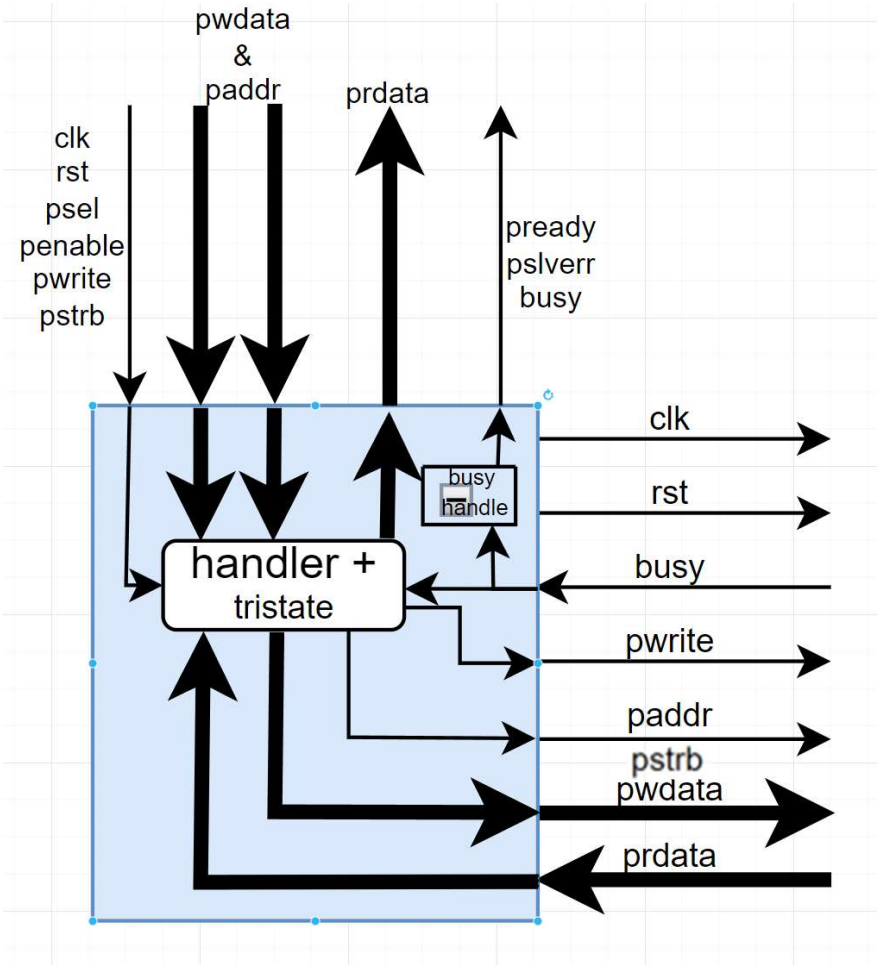
בנוסף לזה הוא מאפשר ל-APB slave לקרוא ממנו את התוצאות. הרגיסטר פיל יסמן busy כאשר התוצאה עדיין לא מוכנה ויחסום קריאה של מידע שעוד לא התייצב או כתובה למידע שמשתמשים בו על מנת למנוע דריסה לא רצויה.

בבלוק הזיכרון ממומש MUX שבורר את היעד של שמירת התוצאה לפי ביטי קונטרול שמוגדרים.

בנוסף לזה השמירה ברגיסטר SP היא בעלת אופציה של אקומולציה ועוד כל מיני אותות בקרה.



## בלוק APB slave :



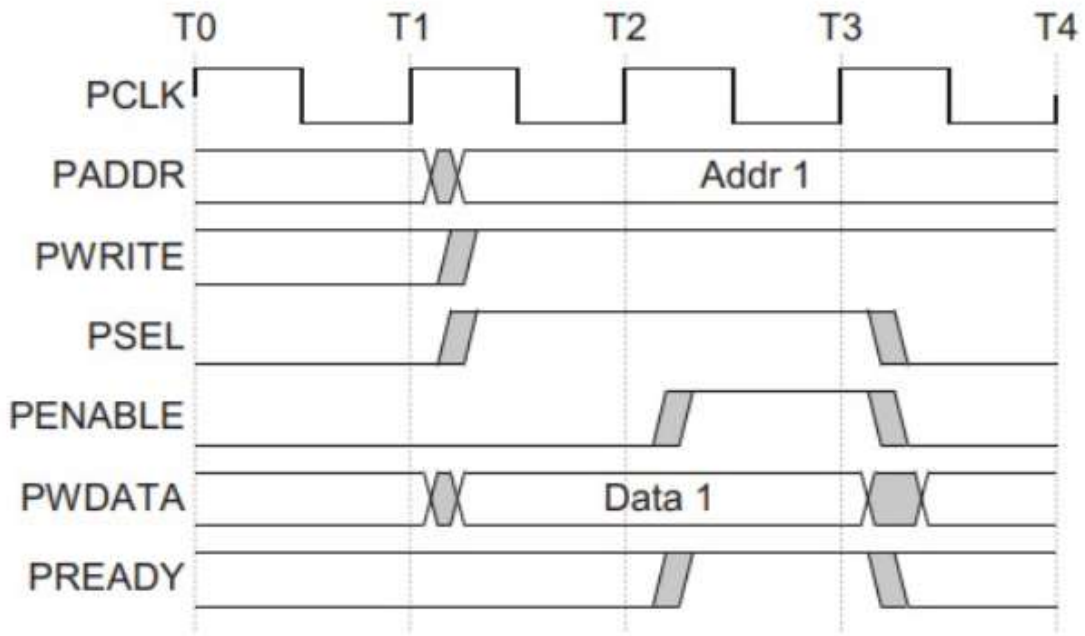

המערכת מממשת APB_slave לפי דף הנתונים שסופק.

המערכת מחכה לPSEL=1 כדי להתחיל בתהליך קריאה או כתיבה.

<u>תהליך הכתיבה:</u>

בתהליך הכתיבה ביחד עם PSEL המערכת מצפה לPWRITE=1 אחרת התהליך ייחשב כתהליך קריאה, לאחר מכן, אם המערכת אינה עסוקה מחכים לPENABLE כדי להעביר את המידע לקובץ הרגיסטרים.

הכתיבה לרכיב התבצעת בתצורת no wait state כך שאם המערת פנויה הכתיבה תתבצע כמעט במידיי.



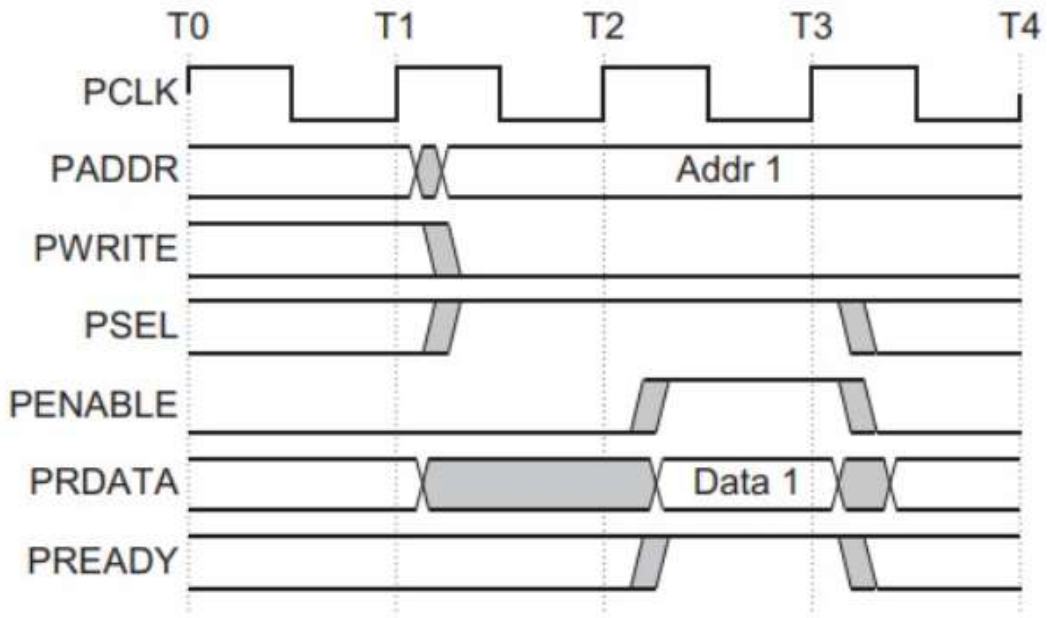

במידה והמערכת עסוקה PREADY=0 וגם BUSY=1 המערכת באמצע פעולת ההכפלה ולא נוכל לכתוב אליה נתונים חדשים, במקרה כזה אם המשתמש ינסה לכתוב למערכת יעלה דגל PSLVERR ל1 ויעיד על בעיה, בנוסף המידע הנכנס והכתובת לא יועברו למערכת שלנו כדי למנוע כתיבה בטעות לקובץ הרגיסטרים.

בנוסף במקרה כזה לא יישלח סיגנל write enable לקובץ הרגיסטרים.

<u>תהליך הקריאה:</u>

בתהליך הקריאה בדומה לתהליך הכתיבה אנו מחכים לPSEL=1 אך הפעם סיגנל PWRITE=0 כדי לסמן לנו שמדובר בתהליך הקריאה.

במצב הקריאה גם כן אנחנו משתמשים בתצורת no wait state וכל עוד המערכת מוכנה היא תשלח את המידע הדרוש החוצה.



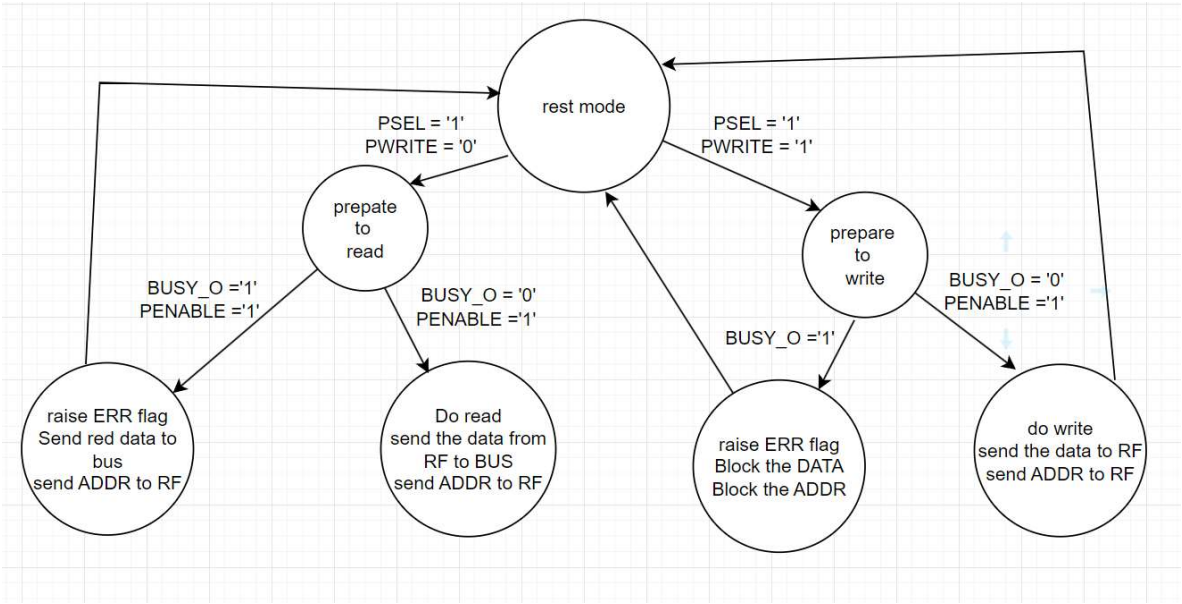

תהליך הקריאה קצת יותר מקל מתהליך הכתיבה מכיוון שאין לנו את הסכנה של דריסת המידע הקודם שלנו, לכן אם ננסה לקרוא בזמן שהמערכת עסוקה עדיין נקבל מידע שמוזרם לנו לבאס לשימושנו ביחד עם סיגנל error וזאת למקרה שבו אנו קוראים את מטריצות B,A או SP שאינן בשימוש.

<u>מצב מנוחה:</u>

במצב שלא פונים לרכיב שלנו הבאס והכתובת אינם מועברים לקובץ הרגיסטרים וגם המידע הנקרא מקובץ הרגיסטרים אינו מועבר החוצה, נשלח אות high z במקום כדי לא להפריע לרכיבים אחרים לעבוד.

<u>העברת הכתובות:</u>

העברת הכתובת לקובץ הרגיסטרים מתבצעת ע"י העברת 6 סיביות הLSB של הכתובת לשם מכיוון שלשאר הסיביות אין משמעות, המכשיר יכול להיות לכל כתובת ובלבד שהכתובות ב6 סיביות הLSB פנויות לעבודה.


rest mode
PSEL = '1'
PWRITE = '0'
PSEL = '1'
PWRITE = '1'
prepate
to
read
prepare
to
write
BUSY_O ='1'
PENABLE ='1'
BUSY_O = '0'
PENABLE ='1'
BUSY_O ='1'
BUSY_O = '0'
PENABLE ='1'
raise ERR flag
Send red data to
bus
send ADDR to RF
Do read
send the data from
RF to BUS
send ADDR to RF
raise ERR flag
Block the DATA
Block the ADDR
do write
send the data to RF
send ADDR to RF

# Excluded rules:

1) File header does not match the defined template – ALL MODULES -**BONUS?**
2) Use a separate line for each HDL statement – ALL MODULES
3) Module has comments density which is below the acceptable minimum ~~(50%)~~ -> (40%)
4) Do not use disallowed character: HT (decimal 9) (TAB)

שלושתם באישור של מייק.

5) Unconnected signals – **only on SYSTOLIC "loose ends"**

בגלל שימוש ביחידות PE, תמיד יישארו חיבורים לא מחוברים ביחידות שנמצאות בצד שמאל או למטה כי לא נצטרך את הפרופוגציה הלאה. אספנו את כל החיבורים לחיבור אחד כדי שבמקום 8 שגיאות נקבל רק בודדת ואותה נשאיר ונתעלם.

6) Nesting at an assignment statement exceeds the maximum of 3/5

```
if (done_i) begin
  for (i = 0; i < DIM; i = i + 1) begin
  for (j = 0; j < DIM; j = j + 1) begin
      case (control[5:1])
```

בגלל שכל הפרויקט עוסק במהותו עם מערכים דו ממדיים, ריבוי קייסים, פרמטריזציה, קשה מאוד ולא פרקטי, שלא להשתמש בקוד שכולל שימושים ב-if/case שנמצאים בתוך לולאה כפולה (דוגמא בצד) לכן ביטלנו את ההזהרה.

7) Avoid using hard coded numeric values such as "[4:0]" for specifying ranges of the multi-bit operand "address" **– only on RegisterFile**

בוטל רק עבור סיגנל של address/subadress . מאחר ומרחב כתובות הוא בהכרח 9 ביטים, כי נתון לפי טבלה שבמקרה הגרוע שבו spn=4 שיושב בכתובת 28, או במקרה שבו spn=1 שיושב בכתובת 16, בכל מקרה נצתרך 5 ביטים לכתובת ראשית , ועבור כתובת-משנה i,j כל אחד log2(MAX_DIM=4) , כלומר 2 ביטים, סה"כ נדרש 9 ביטים ומיקום הביטים הינו קבוע ולא משתנה.

8) Dimension/Range definition "[DIM*i+j +: 1]", for "data_out ", does not comply to descending order convention – **only on RegisterFile**

אין פה באמת ריצה בסדר הפוך כל המידע מועבר כמטריצה שטוחה בסדר רגיל. כנראה הנוטציה "+:" בשילוב אינדקס רץ מבלבל את ה-checker. כנראה כאשר i=j=0 הוא רואה **"[0+: 1]"** וחושב שהסדר של הביטים הפוך.

9) Bit widths differ on left (32) and right (33) of assignment. – **only on RegisterFile**

```
63    for (i = 0; i < DIM; i = i + 1) begin
64       for (j = 0; j < DIM; j = j + 1) begin
```

מדובר על אינדקס הלולאה בלבד ולא בסיגנל.

**המשך ->**

10) Register_has not been initialized using a reset. – **only on RegisterFile**

```
always @(posedge clk_i) begin: MEM_WRITE_AND_RESET
if (rst_i) begin
   data_out <= {BUS_WIDTH{1'bz}};
   control <= {CONTROL_WIDTH{1'b0}};
   for (i = 0; i < DIM; i = i + 1) begin
      for (j = 0; j < DIM; j = j + 1) begin
         matrix_A[i][j] <=0;
         matrix_B[i][j] <=0;
         flags[i][j] <= 0;
         SP1[i][j] <= 0;   // suppose to reset all the values of SP1
         //if(SP_NTARGETS >= 2)
         SP2[i][j] <= 0; // suppose to reset all the values of SP2
         //if(SP_NTARGETS == 4)
         SP3[i][j] <= 0;  // suppose to reset all the values of SP3
        //if(SP_NTARGETS == 4)
         SP4[i][j] <= 0; // suppose to reset all the values of SP4
      end
```

כנראה באג, האיפוס אכן קורה, כנראה בגלל שהאיפוס קורה בתוך לולאה הוא לא רואה זאת.

11) FSM states should not be Hardcoded. **-only on "flow control"**

לא מובן מה הסיבה לשגיאה, ניסינו לשנות את משתנה state לפרמטר/פרמטר לוקאלי ותמיד יש שגיאה אחרת.

12) Net 'paddr_i[15:9]' is unused. -**APV_SLAVE only**
13) Input port 'paddr_i[15:9]' is never used (read from) -**APV_SLAVE only**

הביטים לא בשימוש הפנימי של קובץ הגיסטרים. להבנתינו הם שמורים עבור מכשירים אחרים שמנהל הסלייב, מלבד המכפל.

14) Unresolved multiple drivers detected on signal 'current_state'**. -flow control ONLY**

ניסנו כל מיני קומבינציות אבל השגיאה לא נעלמת, הפונקציונליות של המודול תקינה. לא הצלחנו להבין מה גורם לכך.