Matrix Multiplication

Matrix Multiplication

MATMUL

Digital High Level Design

Version 0.1

by :

Yair Ross 207287889

Dan Bar 212404982

Table of Content

1. ***LIST OF FIGURES*** 3

2. ***LIST OF TABLES*** 7

3. BLOCKS FUNCTIONAL DESCRIPTIONS .....

3.1. << Apb_Slave functionality >> 14

3.2 << Memory functionality>> 15

3.3 << Control functionality>> 16

3.4 << Buffers functionality>> 17

3.5 << Systolic_Array functionality>> 18

3.6 << Adder functionality>> 19

3.7 << ScratchPad functionality>> 20

1. LIST OF FIGURES

Figure 1: view of the design.....4
Figure 2: view of the FSM of the design.....5
Figure 3: view of the FSM of the Apb_Slave 5
Figure 3: view of the Buffers/ Systolic Array 6

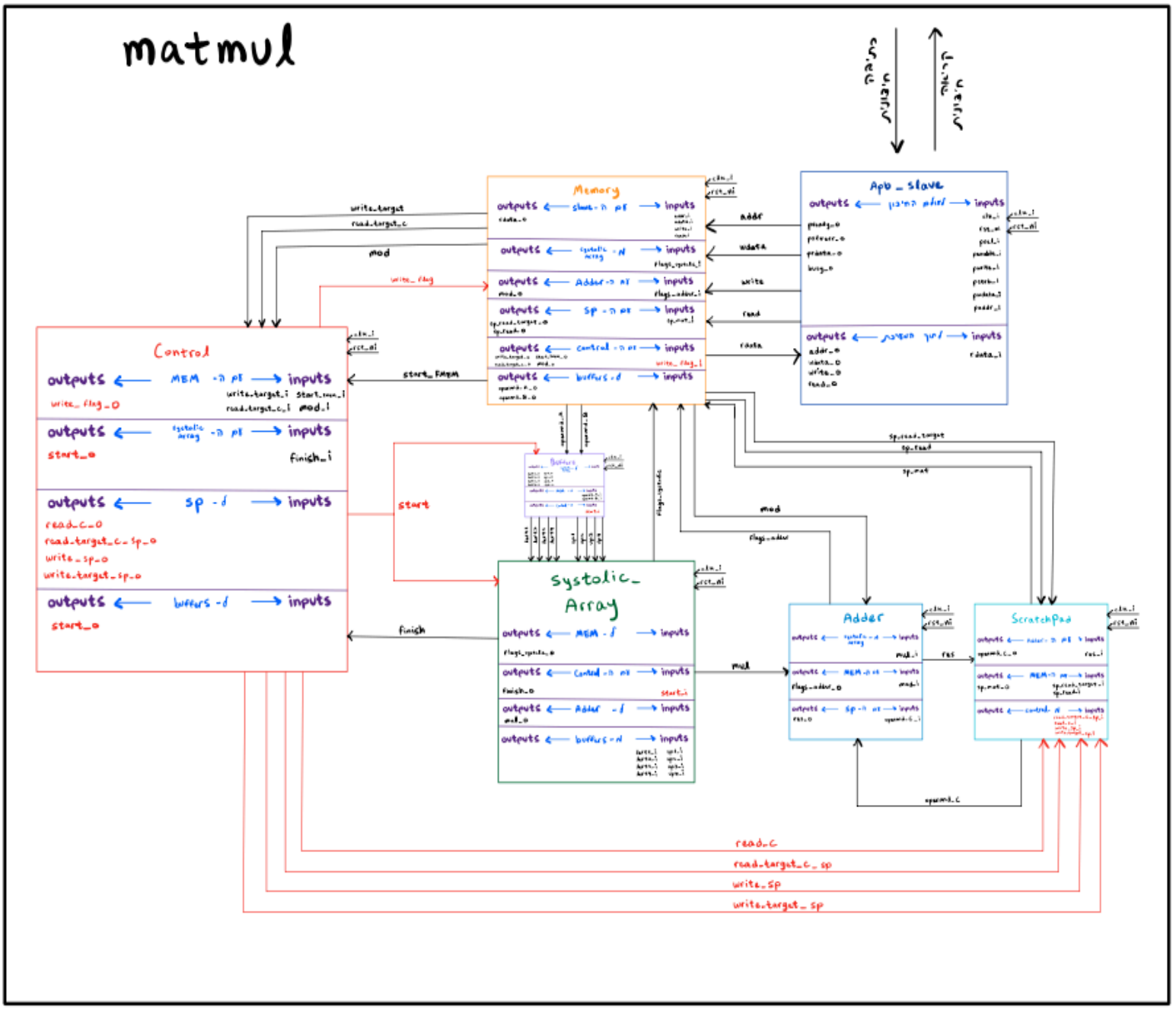


Figure 1: view of the design.

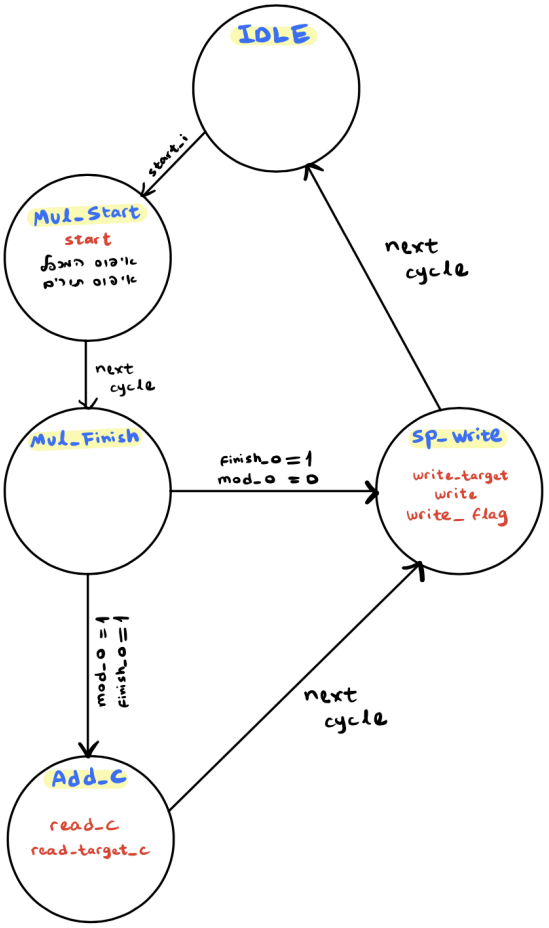


Figure 2: view of the FSM of the design

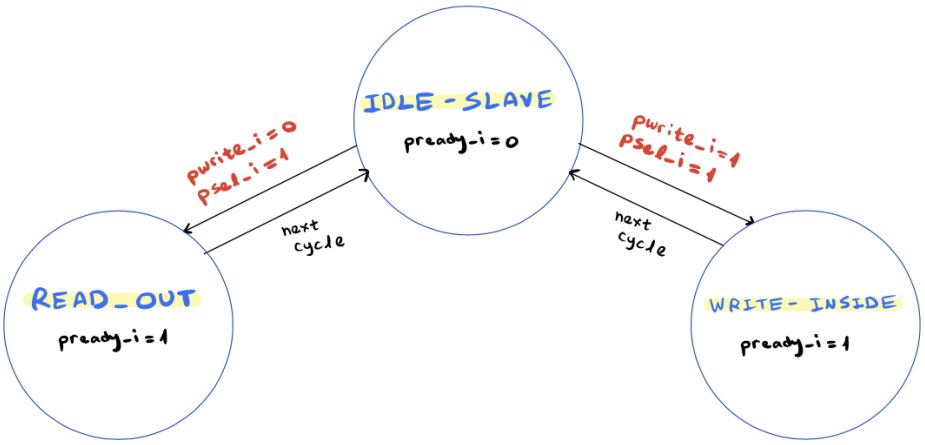


Figure 3: view of the FSM of the Apb_Slave

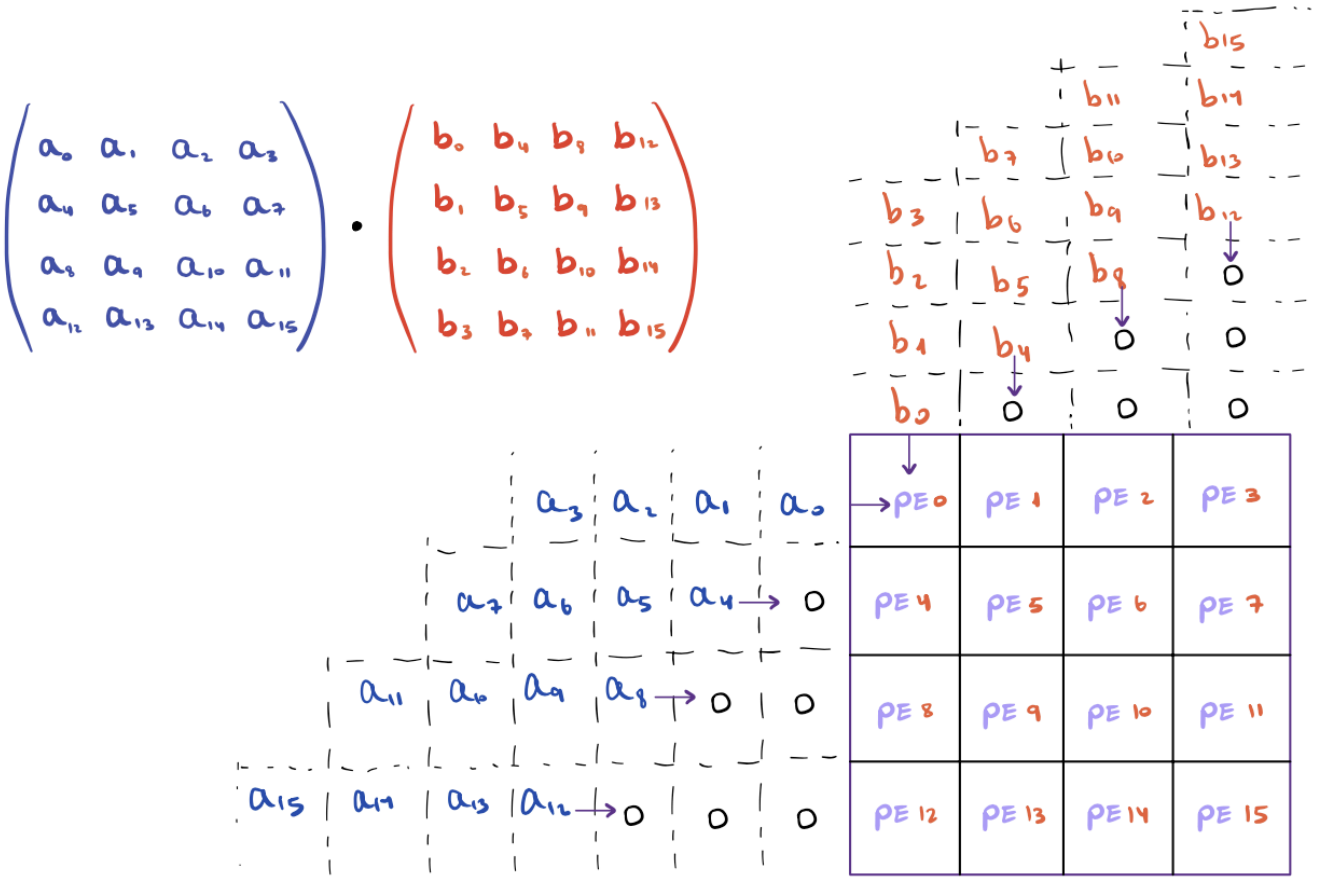


Figure 4: view of the Buffers/ Systolic Array

2. LIST OF TABLES

Table 1: Apb_Slave interface. 8
Table 2: Memory interface. 9
Table 3: Control interface. 9
Table 4: Buffers interface. 9
Table 5: Systolic_Array interface. 9
Table 6: Adder interface. 9
Table 7: ScratchPad interface. 9

Table 1: Apb_Slave interface.

```
input                          clk_i; // Clock signal for the design
input                          rst_ni; // Reset signal, active low
input                          psel_i; // APB select
input             penable_i; // APB enable
input                          pwrite_i; // APB write enable
input      [MAX_DIM-1:0]  pstrb_i; // APB write strobe ('byte' select)
input        [BUS_WIDTH-1:0] pwdata_i; // APB write data
input        [ADDR_WIDTH-1:0] paddr_i; // APB address
input        [BUS_WIDTH-1:0] rdata_i;
output                    pready_o; // APB slave ready
output          pslverr_o; // APB slave error
output   [BUS_WIDTH-1:0] prdata_o; // APB read data
output          busy_o; // Busy signal, indicating the design cannot be written to
output   [ADDR_WIDTH-1:0] addr_o; // the address to read from memory
output   [BUS_WIDTH-1:0] wdata_o; // transfers the data to the design
output                   write_o; // gives the permission to write to the design
output                    read_o; // read bit from the design
```

Table 2: Memory interface.

```
input clk_i; // clock
input rst_ni; // reset

input [ADDR_WIDTH-1:0] addr_i; //
input [BUS_WIDTH-1:0] wdata_i;
input write_i; // write signal from the Apb module
input read_i; // read signal from the Apb module

input [BUS_WIDTH-1:0] sp_mat_i; // matrix element that arrives from ScratchPad module
input [MAX_DIM**2-1:0] flags_adder_i; // vector of carries that arrives from the Adder module
input [MAX_DIM**2-1:0] flags_systolic_i; // vector of carries that arrives from the systolic array module
input write_flag_i; // signal from the control that tells to write the carries to the flags register

output [BUS_WIDTH-1:0] rdata_o; // transfer element from this module to the APB module
output mod_o; // tells the Adder module which operation to execute

output [BUS_WIDTH*MAX_DIM-1:0] operand_A_o; // the first operand to multiply - assigned to the buffers module
output [BUS_WIDTH*MAX_DIM-1:0] operand_B_o; // the second operand to multiply - assigned to the buffers module

output [SP_NTARGETS/4:0] sp_read_target_o; // the target to read from ScratchPad
output sp_read_o; // read from the ScratchPad
output [MAX_DIM-1:0] sp_mat_index_o;

output start_FMEM_o; // tells the control to start the multiply operation
output [1:0] write_target_o; // tells the control what is the target to write in ScratchPad
output [1:0] read_target_c_o; // tells the control what is the target to read from the ScratchPad as Matrix C
```

Table 3: Control interface.

```
input clk_i; // clock
input rst_ni; // reset

input mod_i; // mod bit - define if to add matrix C
input start_FMEM_i; // bit start from the memory
input finish_i; // finish bit from the systolic array (end of multiply)

input [1:0] write_target_i; // the target to write in the scratchpad
input [1:0] read_target_c_i; // the read target for operand C

output  start_o; // start bit - reset for the Buffers and Systolic_Array
output read_c_o; // read bit for operand C from the ScratchPad
output write_o;

output [SP_NTARGETS/4:0] read_target_c_o; // the read target for operand C
output [SP_NTARGETS/4:0] write_target_o; // the target to write in the scratchpad

output write_flag_o; // write bit for the vector of carries
```

Table 4: Buffers interface.

```
input clk_i; // clock
input rst_ni; // reset
input start_i; // start bit
input [BUS_WIDTH*MAX_DIM-1:0] operand_A_i; // operand A
input [BUS_WIDTH*MAX_DIM-1:0] operand_B_i; // operand B


output [MAX_DIM*DATA_WIDTH-1:0] left_o; // left input to the systolic array (input 0)

output [MAX_DIM*DATA_WIDTH-1:0] up_o;   // up input to the systolic array (input 0)
```

Table 5: Systolic_Array interface.

```
input clk_i; // clock
input rst_ni; // reset
input start_i; // start bit

input [MAX_DIM*DATA_WIDTH-1:0] left_i; // input element of matrix A all rows

input [MAX_DIM*DATA_WIDTH-1:0] up_i; // input element of matrix B (column 0)

output [(BUS_WIDTH*(MAX_DIM**2))-1:0] mul_o; // concatenation of all the arithmetic blocks
results

output [MAX_DIM**2-1:0] flags_systolic_o; // // concatenation of all the arithmetic blocks
carries

output finish_o; // finish bit - connected to the control
```

Table 6: Adder interface.

```
input clk_i; //clock
input rst_ni; //reset

input [(BUS_WIDTH*(MAX_DIM**2))-1:0] mul_i; // result of the systolic array

input mod_i; // mod bit - determine if the output of this module includes the add of matrix C

input [(BUS_WIDTH*(MAX_DIM**2))-1:0] operand_c_i; //matrix C that comes from ScratchPad

output [(BUS_WIDTH*(MAX_DIM**2))-1:0] res_o; // the result of the adder module

output [(MAX_DIM**2)-1:0] flags_adder_o; // output vector of carries
```

Table 7: ScratchPad interface.

```
input clk_i; // clock
input rst_ni; // reset
input write_sp_i; // signal from the control that indicates to write to the ScratchPad

input [(BUS_WIDTH*(MAX_DIM**2))-1:0] res_i; // result from the Adder module that need to be stored in the ScratchPad

input [SP_NTARGETS/4:0] read_target_c_sp_i; //matrix read target from ScratchPad to matrix C
input [SP_NTARGETS/4:0] write_target_sp_i; // matrix write target to ScratchPad from the Adder module
input read_c_i; // signal from the control module that indicates to read matrix C from the ScratchPad

input [SP_NTARGETS/4:0] sp_read_target_i; // matrix read target from ScratchPad to memory_decoder module
input sp_read_i; // signal from the memory_decoder module that indicates to read matrix from the ScratchPad to the memory_decoder
input [MAX_DIM-1:0] sp_mat_index_i;

output [(BUS_WIDTH*(MAX_DIM**2))-1:0] operand_c_o; // matrix c - suppose to get added with the systolic array output when mod bit is set

output [BUS_WIDTH-1:0] sp_mat_o; // element of chosen matrix that goes to the memory_decoder module
```

3. BLOCKS FUNCTIONAL DESCRIPTIONS

3.1 << Apb_Slave functionality>>

מודול זה אחראי על תקשורת עם ה "עולם החיצון".

דבר זה מתבטא בפעולות קריאה וכתיבה מהבאס החיצוני אל תוך הדיזיין שלנו ובעצם מהווה כמתווך.

המודול בנוי כמכונת מצבים בה קיימים שלושה מצבים:

1. IDLE_SLAVE
2. WRITE_INSIDE
3. READ_OUT

במצב reset כל הסיגנלים שמחברים את המודול אל העולם החיצון מאופסים ומכונת המצבים נמצאת במצב IDLE_SLAVE.

עבור ביט חיצוני psel_i שנדלק , מקבל ה - דיזיין בחירה כרכיב פריפרייאלי ומוכן לבצע אופרציה מתאימה.

עבור סיגנל pwrite_i דלוק נעבור למצב WRITE_INSIDE כלומר מצב כתיבה ועבור סיגנל זה כבוי נעבור למצב READ_OUT כלומר למצב קריאה מהדיזיין החוצה.

נשים לב שפעולות אלו מתבצעות במחזור שעון אחד וחוזרות חזרה למצב IDLE_SLAVE (סיגנל pready נדלק).

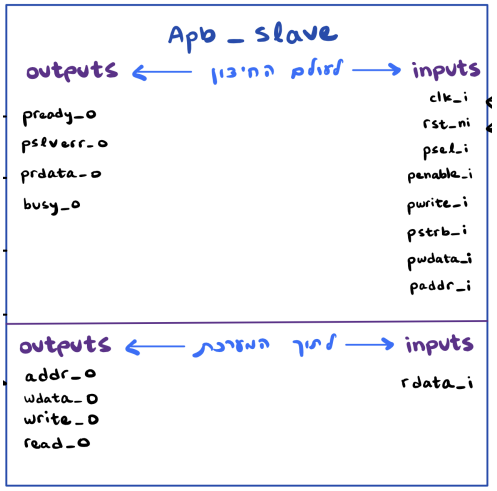
כמו כן, פעולות כתיבה וקריאה תבצעות באמצעות כתובת יעודית לקריאה או כתיבה.

כתובות לקריאה אפשריות הן האופרנדים A,B שקיימים ב - Memory, רגיסטר ה - control ורגיסטר ה - flags.

כמו כן ניתן לקרוא גם מה - ScratchPad מכתובת מתאימה.

במימוש הקוד התייחסנו לדרישה שניתן לקרוא מהדיזיין נתונים ללא קשר אם המערכת נמצאת בפעולה מסוימת (state) כלשהו.

לכן עיצבנו ומימשנו שני תרשימי FSM בלתי תלויים - אחד עבור ה Apb_Slave ואחד עבור הדיזיין עצמו.



3.2 <<Memory functionality>>

מודול זה אחראי על אחסון האופרנדים A,B , וקטור ה - flags (carries) ורגיסטור הבקרה (control).

מטריצות A,B נשמרות כמטריצות בגודל 1*1 , 2*2 , 4*4 כאשר הפרמטר MAXDIM קובע את המימד.

בנוסף לאחסון מתבטא תפקידו בתקשורת עם כל שאר המודולים:

בתקשורת עם המודול Apb_Slave תפקידו להעביר אליו ולקבל ממנו מידע ברוחב BUS_WIDTH ע"י סיגנלי בקרה שמסמנים את סוג האופרציה המתבצעת (קריאה / כתיבה).

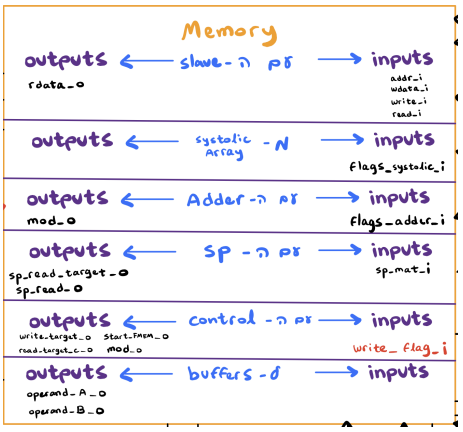
בתקשורת עם מודול ה - Control תפקיד ה - Memory הוא לסמן התחלת פעולת הכפלה במכפל (Systolic_Array).
כמו כן ה - Memory נותן ל - Control את הכתובת לכתיבה במודול ה - ScratchPad כאשר יש מטריצה מוכנה לכתוב לו וכתובת לקריאה כאשר רוצים למשוך מה - ScratchPad מטריצה בתור אופרנד C (אופרנד חיבור).
בנוסף לאלו נותן ה - Memory ל - Control את ה - mod כלומר את סוג הפעולה המתבצעת בדיזיין - (הכפלה/חיבור).

בתקשורת עם מודול ה - Buffers תפקיד ה - Memory הוא להעביר לו את האופרנדים A,B מוזנים נכון לבאפרים במצב reset כאשר מרפדים באפסים במקומות הנכונים בהתאם ל - MAXDIM.

בתקשורת עם מודול ה - Adder תפקיד ה - Memory לסמן את ה - mod כלומר האם ה - Adder נכנס לפעולת חיבור על תוצאת המכפל או שהאחסון ב - ScratchPad עוקף את פעולת החיבור ונשמרת פעולת כפל ותו לא.

בתקשורת עם ה - ScratchPad תפקיד ה - Memory לאפשר קריאה ממנו (ביט קריאה וכתובת לקריאה) ולקבל ממנו אלמנט אלמנט (sub addressing) מהמטריצה היעודית לקריאה.

עבור שני הוקטורים שבגודל MAX_DIM ** 2 המתקבלים במודול ה - Memory ע"י המודולים Adder ו Systolic_Array מתבצעת פעולת OR ביט ביט והתוצאה המתקבלת היא וקטור ה - flags הסופי עבור אופרציה מלאה.



3.3 << Control functionality>>

מודול זה אחראי על מכונת המצבים של הדיזיין כלומר לשלוח אותות בקרה למודולים השונים כדי ליצור תזמון נכון של פעולות הכפל והחיבור ואחסון מטריצת התוצאה במודול ה- ScratchPad .

בתקשורת עם ה - Buffers תפקיד ה - Control לשלוח לו סיגנל start שמורה על הזנת האופרנדים A,B לבאפרים ואיפוס התאים הנותרים.

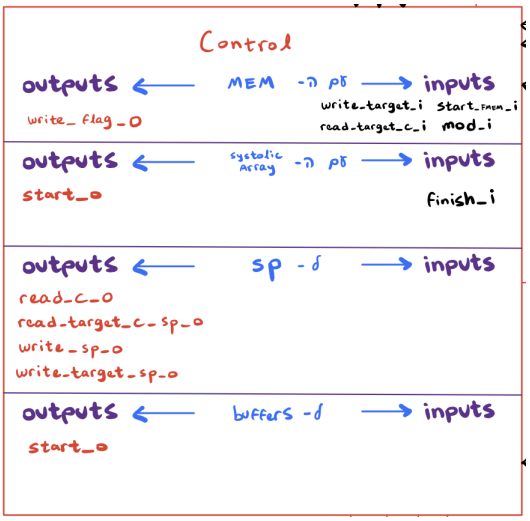
כמו כן סיגנל ה - start מתקבל גם ע"י מודול ה - Systolic_Array.

סיגנל זה מאפס עבור ה - Systolic_Array את תאי החישוב שלו (process elements).

בתקשורת עם רכיב ה - ScratchPad מורה ה - Control את כתובת הכתיבה וביט שמאשר כתיבה עבור תוצאת חישוב סופית שצריכה להיכתב.

כמו כן מורה ה - Control את כתובת הקריאה וביט שמאשר קריאה עבור אופרנד C שצריך להיקרא לטובת חיבור עם מודול ה - Adder.

בתקשורת עם ה - Memory נותן ה - Control אישור לכתיבה של וקטור ה - flags (carries) בזמן הנכון בו התוצאה כבר מוכנה אל רגיסטר יעודי ששומר את ה - flags.

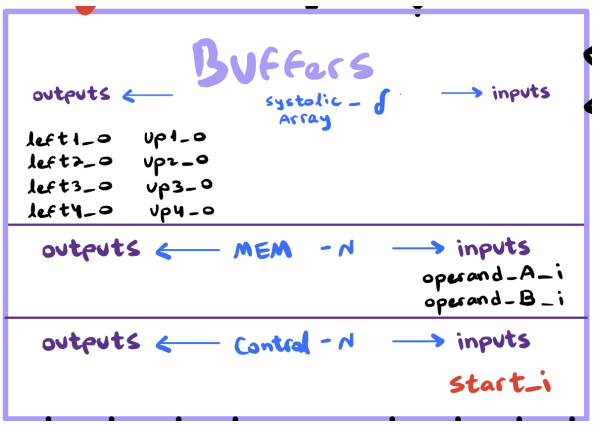


3.4 << Buffers functionality>>

מודול זה אחראי על מעבר האופרנדים A,B אל תוך יחידותיו השונות של ה - Systolic_Array בצורה נכונה

מצפון וממערב על מנת שהנתונים יזרמו בתוכו בשרשור ויתנו תוצאת כפל בזמן יעיל.

מבנה הבאפרים הוא במבנה פרמטרי ומותאם ל - MAXDIM בגדלים 1,2,4 כלומר הרגיסטרים בגדלים משתנים בהתאם לדרישה.



3.5 << Systolic_Array functionality>>

מודול זה אחראי על שרשור יחידות חישוב (aritmetic blocks) וליצור כמעין רשת בגודל של MAX_DIM ** 2

של aritmetic blocks, כאשר כל יחידה כזאת מקבלת מצפון וממערב קלט ברוחב DATA_BUS

ותפקידה לסכום את פעולת הכפל של שני הקלטים ולהעביר בהתאמה לדרום ולמזרח את שני הקלטים אל יחידת החישוב הבאה כדי ליצור רשת זרימת נתונים.

היתרון של רשת חישוב זו הוא החישוב המקבילי המתבצע ביחידות השונות (aritmetic blocks).

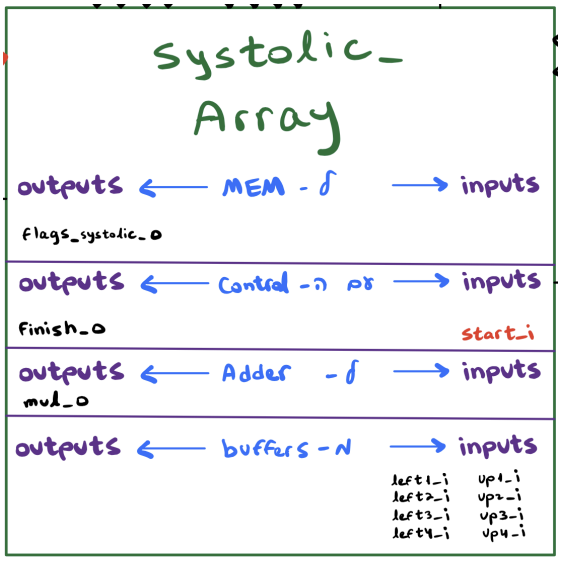
עבור כל יחידת חישוב שמרנו ביט אקסטרה כלומר הרחבנו את הרגיסטרים כדי לשמור carry עבור כל סוכם.

את כל ה - carries אנו משרשרים לוקטור ברוחב MAX_DIM ** 2 ושולחים אל ה - Memory.

בתקשורת עם מודול ה - Adder שולח ה - Systolic_Array אליו את תוצאת הכפל (mul).

בתקשורת עם מודול ה - Control שולח ה -Systolic_Array ביט finish שמורה ל - Control על סיום חישוב כפל המטריצות.

ביט זה נדלק כאשר עבר מספר מתאים של cycles שכולל בתוכו את החישוב הנדרש.



3.6 <<Adder functionality>>

מודול זה אחראי על פעולת החיבור של אופרנד C לתוצאת הכפל A*B בהתאם ל - mod שמתקבל ע"י ה - Memory .

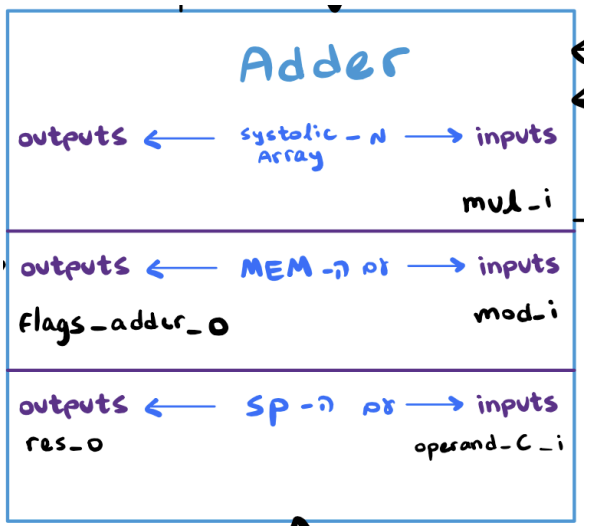
אם ה - mod הוא 1 לוגי אזי מתבקשת פעולת חיבור וה - Adder יכתוב תוצאה (res) למודול ה - ScratchPad.

כאשר ה - mod הוא 0 לוגי תיכתב ל - ScratchPad תוצאת הכפל שנכנסה למודול ה - Adder.

עבור כל אלמנט חישובי בתוך ה - Adder הוספנו ביט אקסטרה עבור ה - carry שתפקידו לקלוט overflow עבור החיבור פר אלמנט במטריצה.

נשים לב שכמות האלמנטים במודול ה - Adder הוא כגודל הפרמטרי MAX_DIM ** 2

את כל ה - carries אנו משרשרים לוקטור ברוחב MAX_DIM ** 2 ושולחים אל ה - Memory.



3.7 << ScratchPad functionality>>

מודול זה אחראי על אחסון מטריצות התוצאה המתקבלות ממודול ה - Adder ושמירתן ברגיסטרים שכמותם הוא על פי הדרישה בגודל פרמטרי 1,2,4.

התקשורת של מודול זה עם ה - Memory מתבטאת בקבלת כתובת לקריאה בגדול משתנה (2 ביט עבור 4 כתובות וביט עבור 1 או 2 כתובות) וביט קריאה שמורה על אופרציית קריאה.

נשים ב שפעולת הקריאה מה - ScratchPad יכולה להתבצע במקביל לפעולת הדיזיין (כפל/ חיבור) שכן מדובר בשתי מכונות מצבים שונות ובלתי תלויות שמאפשרות את הדרישה לקריאה בכל עת.

מחשבה זו נבעה מכך שכאשר אנו נמצאים באחד המצבים של פעולת הכפל לא נצליח לחתוך את המכונה בכל עת ולעבור לתקשורת מול מודול ה - Apb_Slave ולכן קיימנו את הפעולות במקביל.

כמו כן יודע ה - ScratchPad להוציא מטריצת תוצאה אל ה - Memory ומשם אל ה - Apb_Slave.

בתקשורת עם ה - Adder מעביר ה - ScratchPad את מטריצה C כתגובה לסיגנל קריאה וכתובת לקריאה בגודל משתנה

ממודול ה - Control.

את מטריצת התוצאה של רכיב ה - Adder שומר ה - ScratchPad באחד מתאיו על פי הצורך.