

LAB 5

Advanced CPU architecture and hardware
accelerators lab

Ido Ron: 322384330

Binyamin Oumanski: 322688946

הקדמה :

מטרה:

בארכיטקטורת mips לעבור מ single cycle ל pipeline ,
שהיתרון בו הוא היעילות כלומר Throughput גבוה יותר.
במעבדה שילבנו ויישמנו אלמנטים תיאורטיים שלמדנו
בהרצאות של הקורס ארכיטקטורת מחשבים.
ביצענו בעבודה isa בסיסית של פקודות אסמבלי , שאותן
סימלצנו, בדקנו והעברנו ורפיקציה במodelsim.
ולאחר מכן מימשנו אותן בחומרה בFPGA.

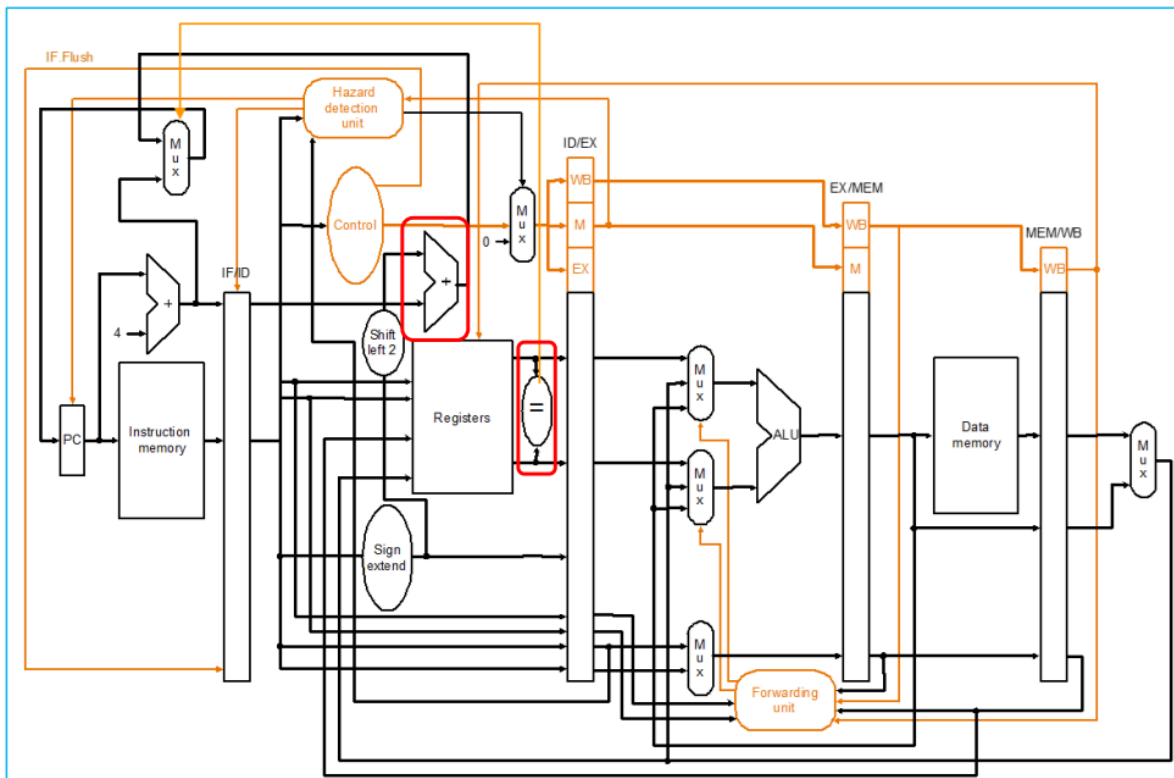


Figure 6: Five-stage pipelined MIPS architecture with forwarding and single delay slot support

מציאת תדר מקסימלי:

נדרשנו למצוא תדר מקסימלי של המערכת, הוא חושב בQUARTUS על ידי מציאת המסלול הרכיבי הארוך ביותר בין שני שעונים. התדר המקסימלי שיצא הוא 50.37 MHz.

מסלול קריטי:

	Fmax	Restricted Fmax	Clock Name	Note
1	42.5 MHz	42.5 MHz	altera_reserved_tck	
2	50.37 MHz	50.37 MHz	clk_i	

נתיב קריטי:

הנתיב הקריטי מגדיר את הזמן הארוך ביותר שדרוש לאות לעבור בין שני איזורים במחזור שעון אחד ולכן קובע את תדירות השעון המקסימלית. אם הנתיב הקריטי ארוך מדי, יייתכנו הפרות של תנאי setup/hold והתפקוד ישתבש. חשוב לשמור על נתיב קריטי קצר על מנת ליצור מערכת אמינה, יעילה ומהירה. אצלנו הנתיב הקריטי הוא בין ה-DECODE לבין ה-FETCH.



פירוט של המערכת :

נראה ונסביר על המערכת והמודולים שמרכיבים אותה.

המערכת:

למערכת הוספנו סיגנלים שסופרים :

CLKCNT- – כמות עליות השעון

FHCNT- – כמות הflushים

STCNT- – כמות הstallים

וסיגנל שמראה מתי עושים ברייק פוינט:

BPADD – כתובת BreakPointn

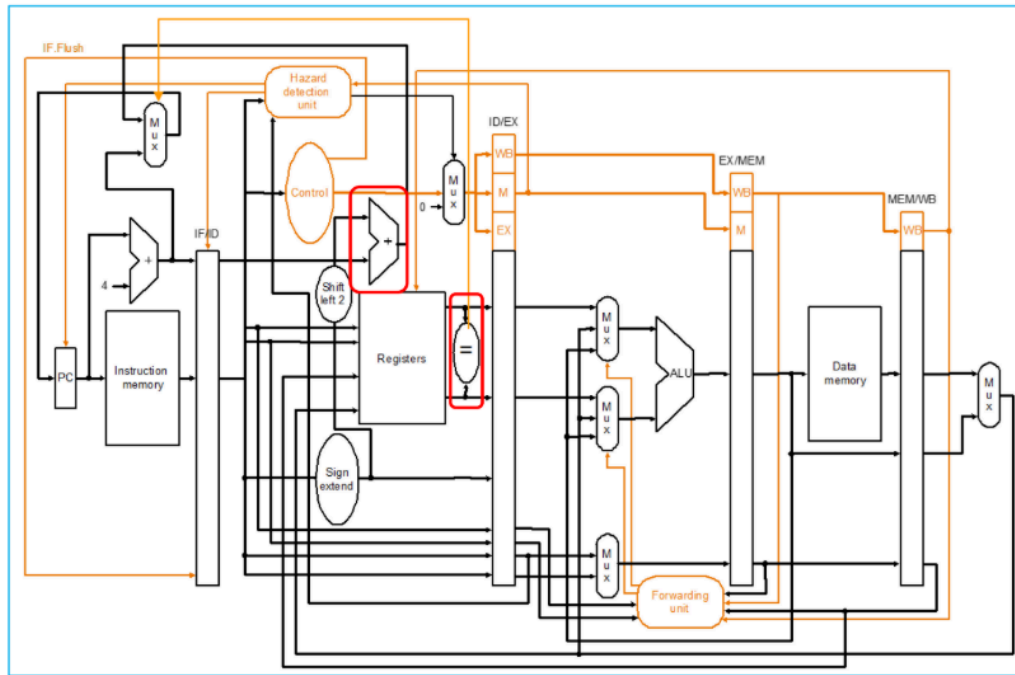
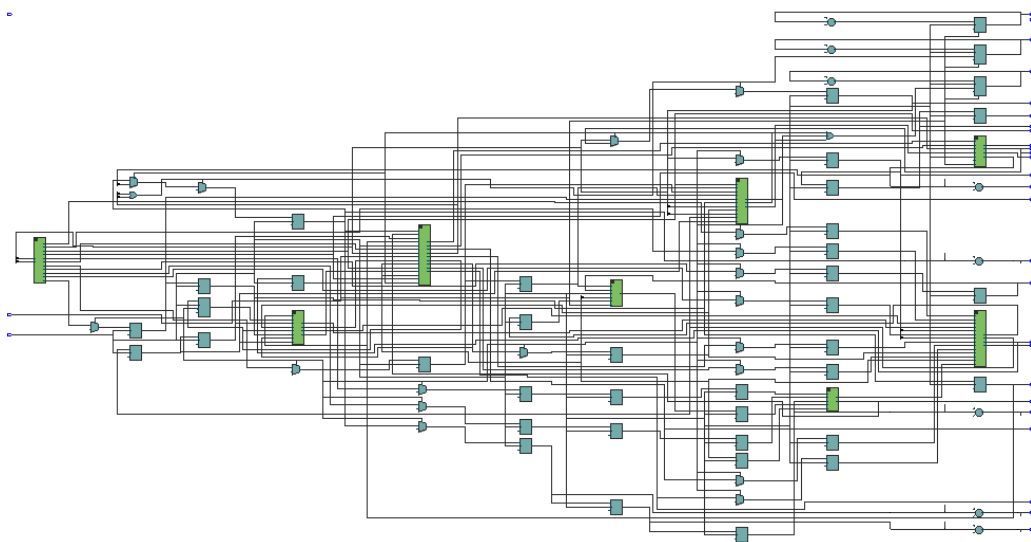


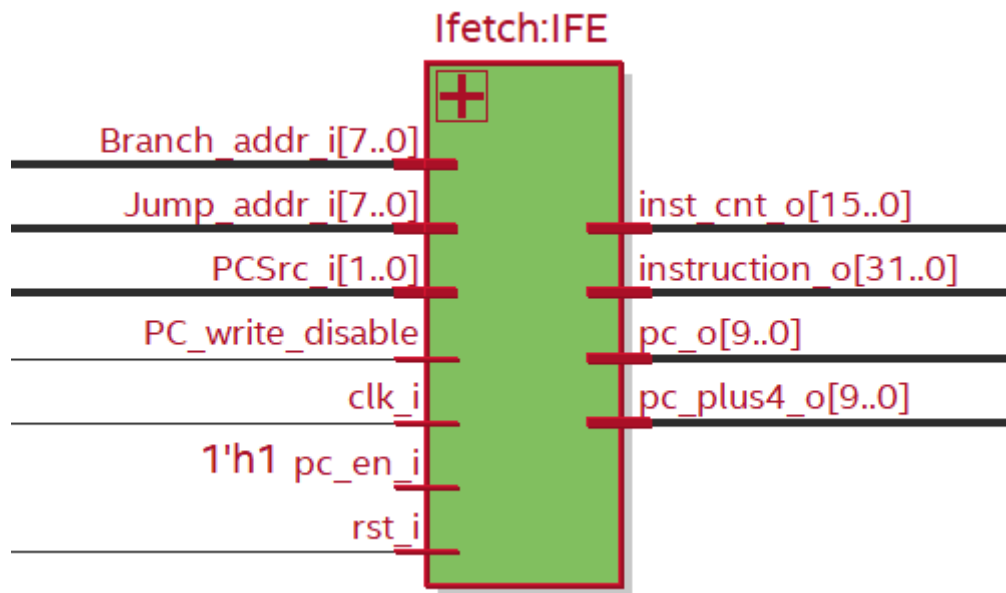
Figure 6: Five-stage pipelined MIPS architecture with forwarding and single delay slot support

תרשים RTL :

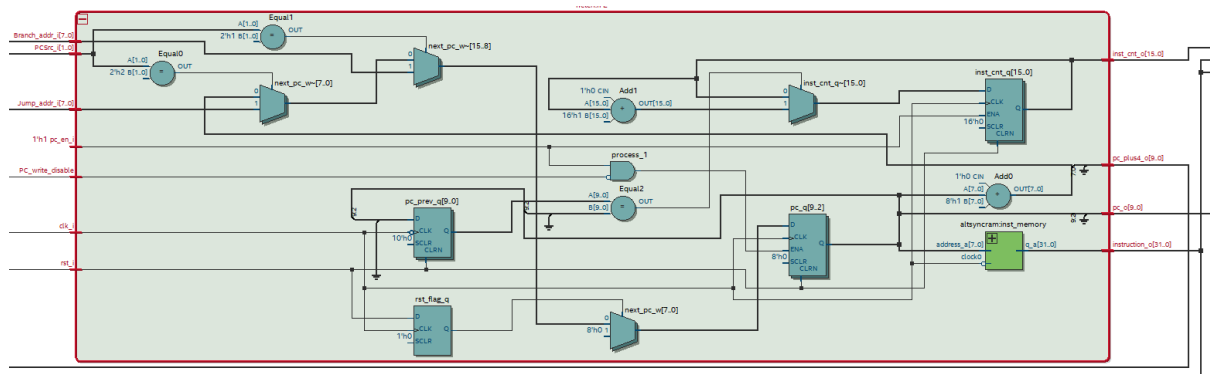


שלב ה-IF

סקירת פעולת המודול
במודול הזה מוציאים את הפקודה בכתובת ה-PC מה-
ITCM.
במימוש ה-Pipeline יישמנו בורר שיבחר את כתובת ה-PC
בהתאם לסוג הפקודה.
האטום בורר בין אופציה של פקודה רגילה, jump ו-branch.



RTL:



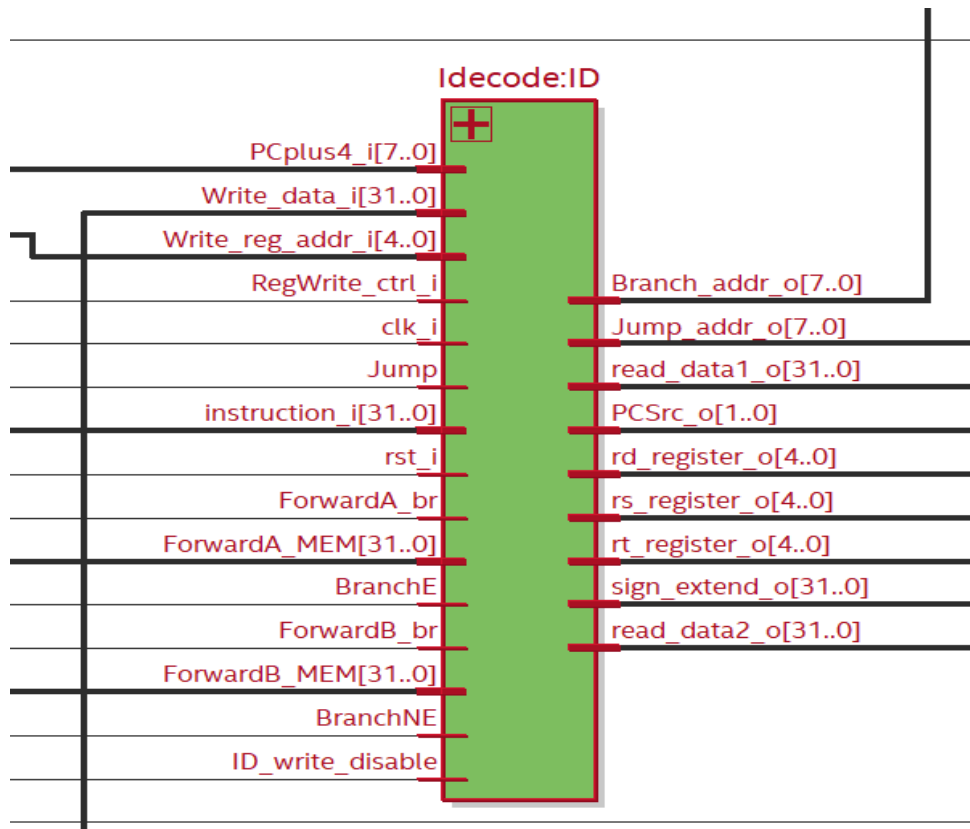
שלב ה-ID

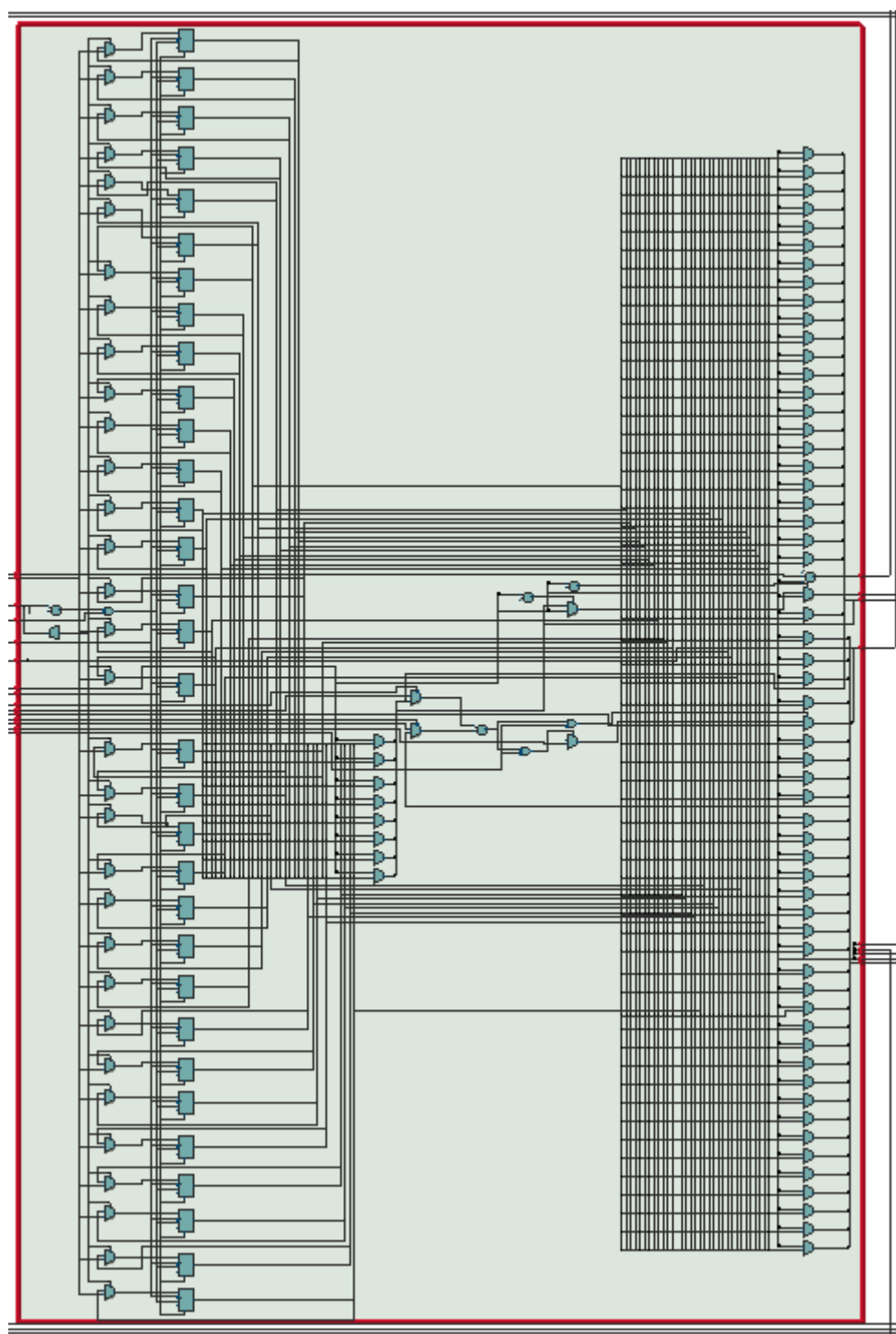
הפקודה עוברת מה IF ל-ID, שם המעבד מסווג את סוג הפעולה לפי Jump TYPE I, J, R, , סיווג זה מתבצע לפי מבנה הפקודה.

Type	-31-	format (bits)					-0-
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)	
I	opcode (6)	rs (5)	rt (5)	immediate (16)			
J	opcode (6)	address (26)					

שלב זה קורה במקביל ל- Control.

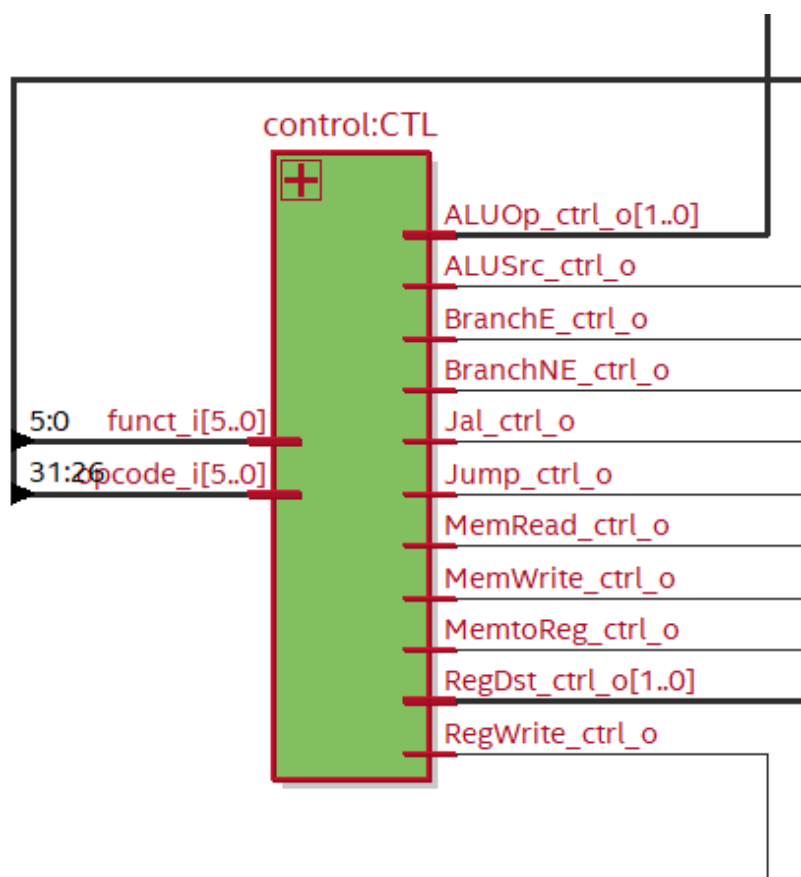
במוצא ה RF יש משווא שבעזרתו ניתן לדעת האם צריך לבצע brunch או jump בהתאם לפקודה של המשתמש. במודולים מערכת קווי בקרה שבהתאם לסוג הפעולה היא יודעת איזה קווי בקרה להפעיל ולהעביר הלאה לשאר המודולים ב Pipeline . לדוגמא כאשר יש פעולה של כתיבה לרגיסטר , המערכת מדליקה את קו הבקרה של regwrite . או לדוגמא כאשר יש פקודה ששיכת לITYPE , ידלק קו הבקרה של pcsrc שאחראי במודול הבא לברור את ה immediate.

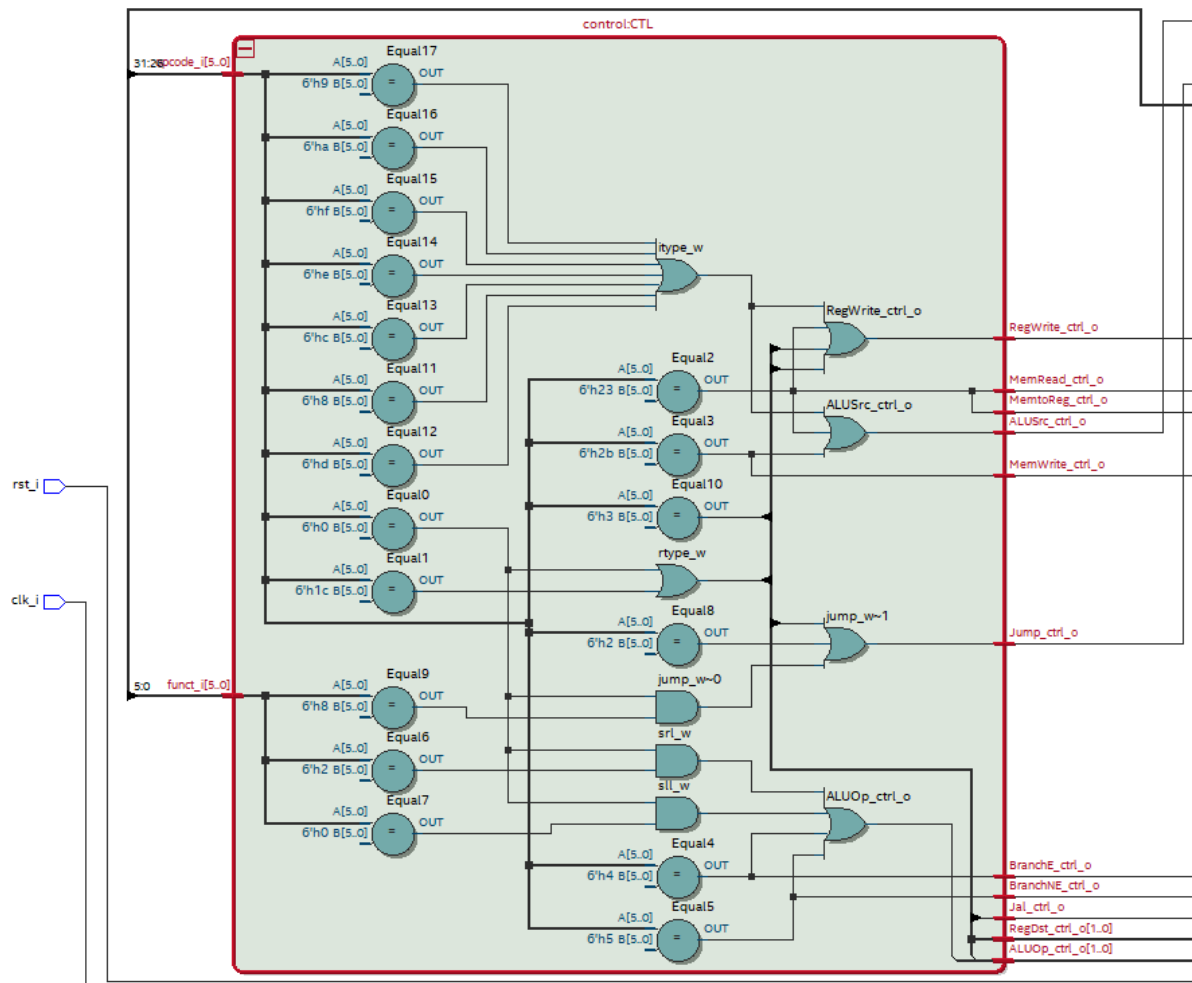




Control

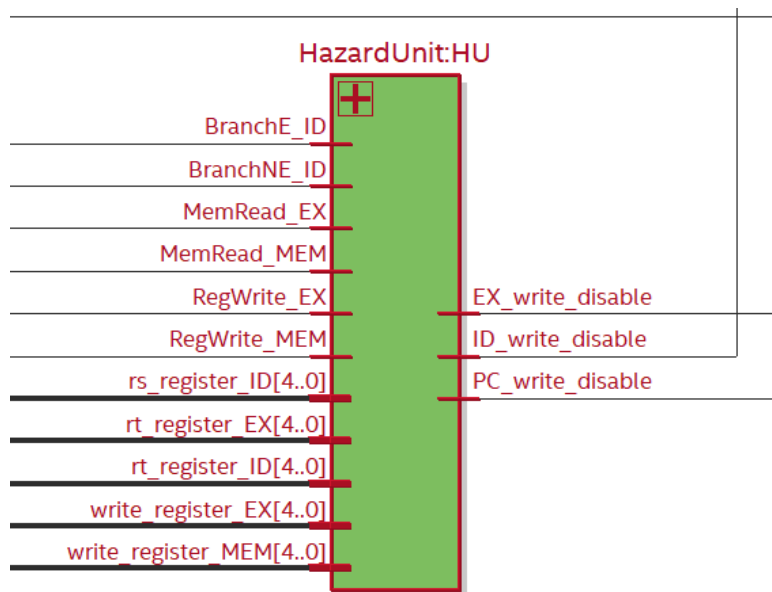
רכיב זה אחראי להוציא קווי בקרה שגם הם רצים ב-Pipeline. קווי הבקרה נדלקים כאשר פעולה מגיעה לשלב ה-ID בהתאם לפעולה. קווי בקרה לדוגמא: כתיבה ל-RF, סלקטורים של MUX, קריאה או כתיבה לזיכרון וכו'.

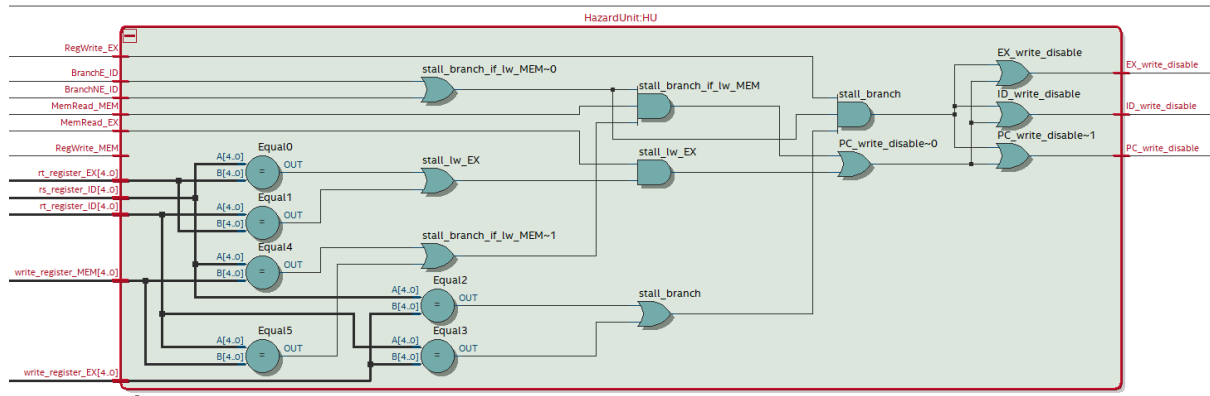




Hazard:

מטרת הרכיב הינה לזהות מצבים המידע דרוש אך לא מוכן וגם לא ניתן לפתרון ע"י Forwarding. במצבים כאלה הרכיב מוציא קוויב בקרה אשר תפקידם לבצע Stall או Flush. Stall גורם ל-PC לעצור, עוצר את רגיסטר ה-ID/IF מלהתעדכן מה שגורם לפעולה ב-ID להישאר, ושם פעולת NOP בשלב ה-EXE. זה בעצם נותן עוד מחזור לפעולה מסויימת להתקדם הלאה ב-Pipeline בכדי שהמידע יהיה מוכן. תפקיד ה-Flush הוא "לנקות" את שלב ה-ID במצבים של Branch taken או Jump.

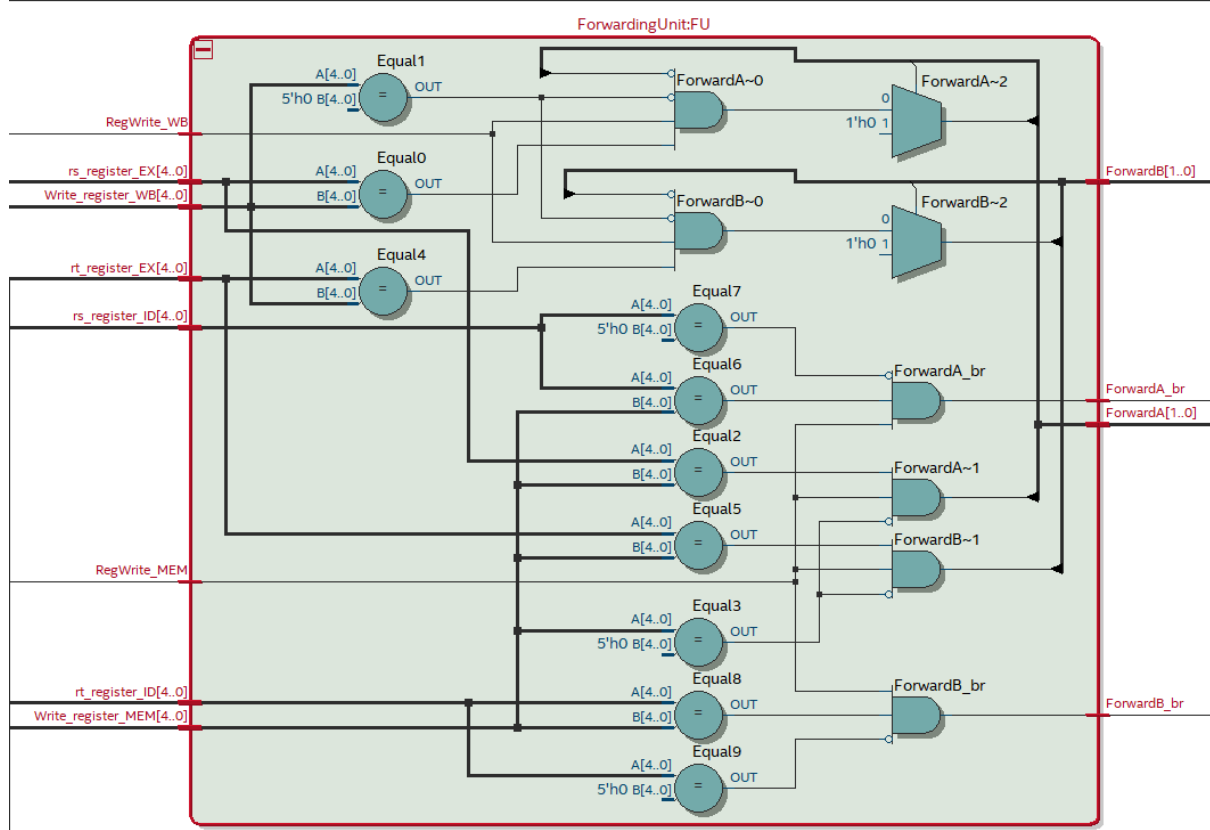
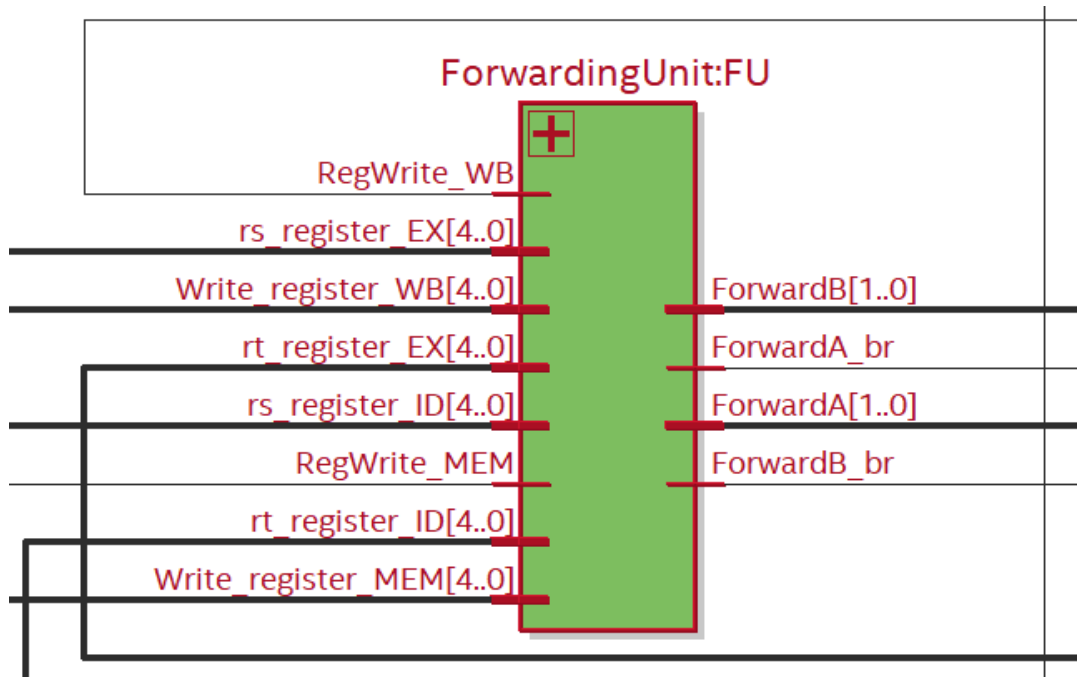




Forwarding:

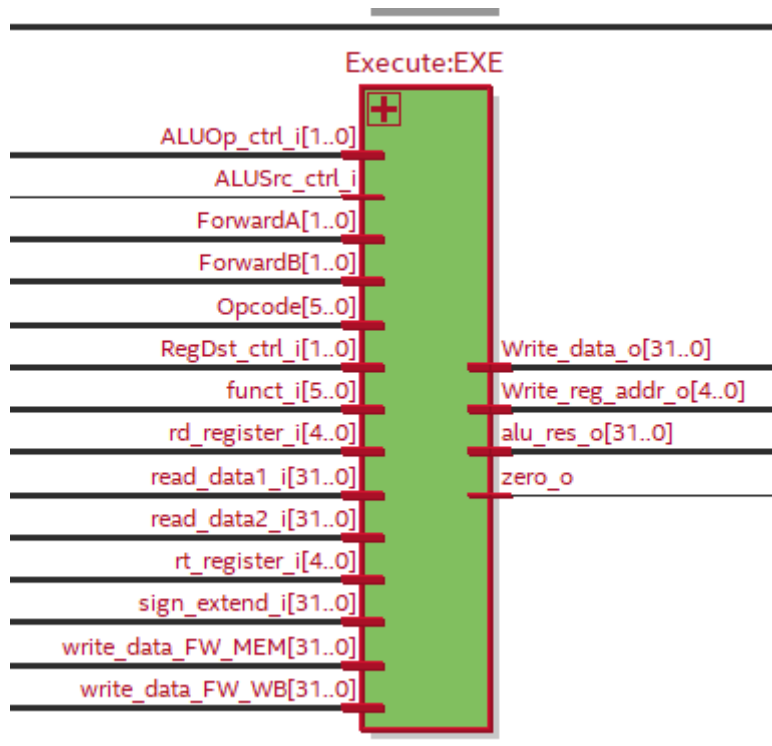
מטרת הרכיב הינו להביא מידע מוכן משלב ה-MEM ושלב ה-WB אל ה-EXE וה-ID.

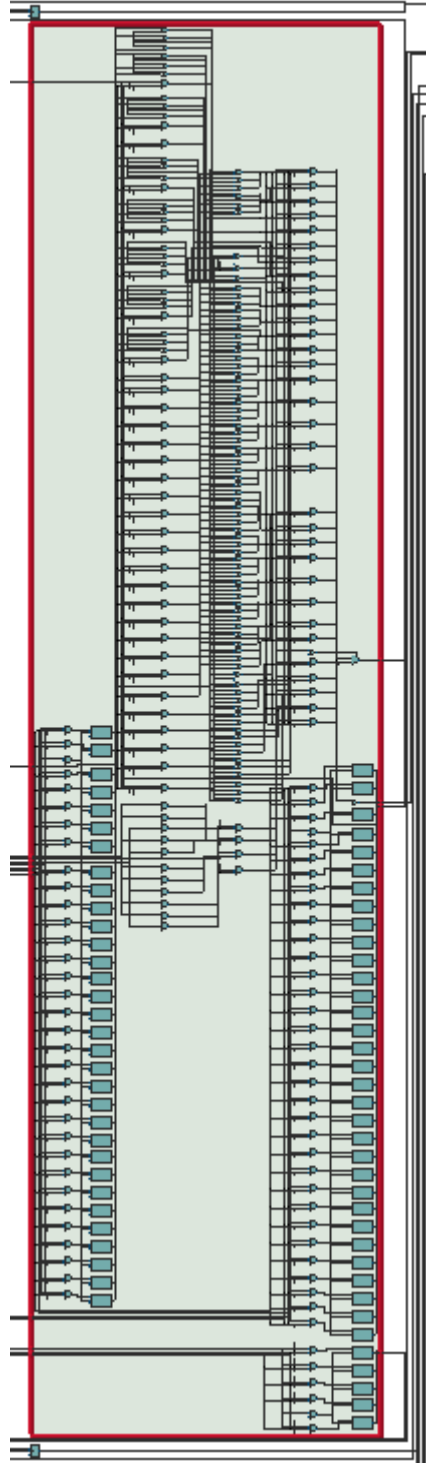
במקרה של פעולה ב-EXE דורשת אופרנדים שעוד לא הגיעו לRF אבל כבר כן חושבו ונמצאים ב-Pipeline אז נחברם לשני הכניסות של ה-ALU בעזרת MUX. בנוסף במקרה של Branch equal/not equal אז הרכיב עושה Forwarding אל ה-ID וזאת במטרה להקטין את כמות השלבים שדורשות ההסתעפויות וכך בעצם גם להקטין את כמות ה-Flush אם אכן ההסתעפויות כן נלקחות. ההחלטה אם אכן לעשות Forward תלויה באם אחד מהרגיסטרים (שהם האופרנדים) של פעולה שבדיוק הגיעה לשלב ה-ID הוא רגיסטר המטרה של פעולות שנמצאות בשלבי ה-EXE וה-MEM.



Execute:

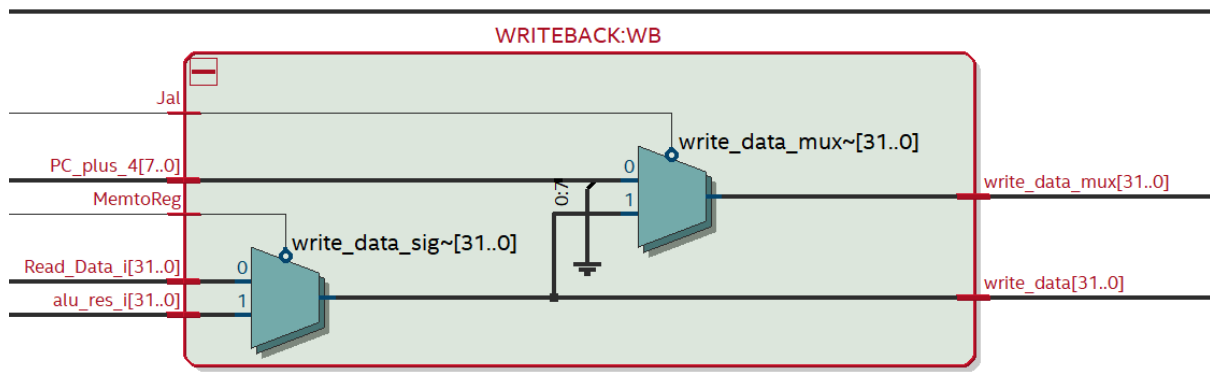
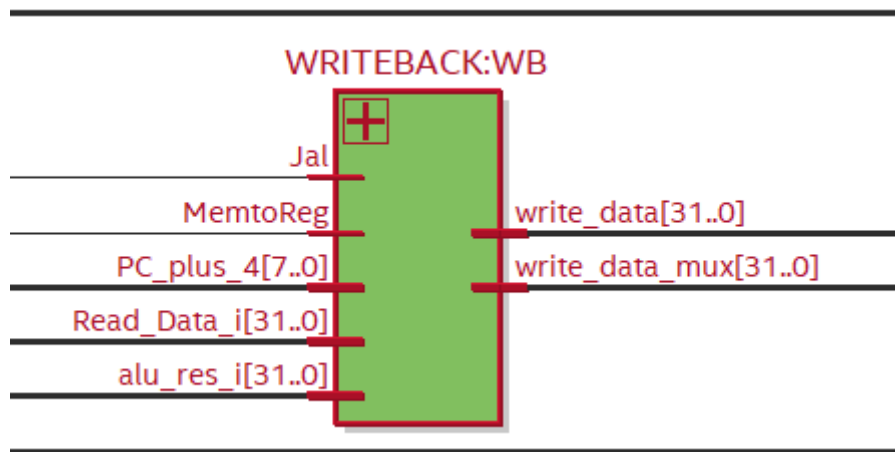
במודול זה קורות הפעולות הארתמטיות במעבד ברכיב הALU, כדוגמא איזה ערך להכניס לרגיסטר או מה לכתוב לזיכרון.
בנוסף בשלב זה קיים forwarding , עליו מפורט בהמשך.





Writeback:

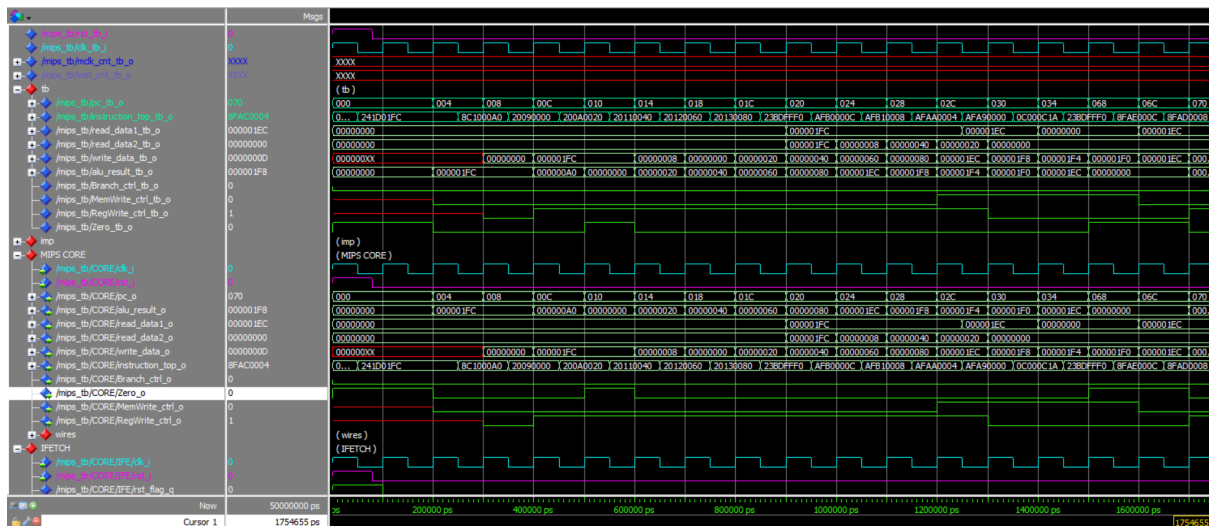
השלב האחרון ב-Pipeline, אחראי להעביר מידע ל-RF ובורר איזה מידע להעביר (תוצאה של ה-ALU, מוצא מהזיכרון או $PC + 4$ במקרה של JAL) ובורר ע"י קווי בקרה שיוצאים מרכיב ה-Control.



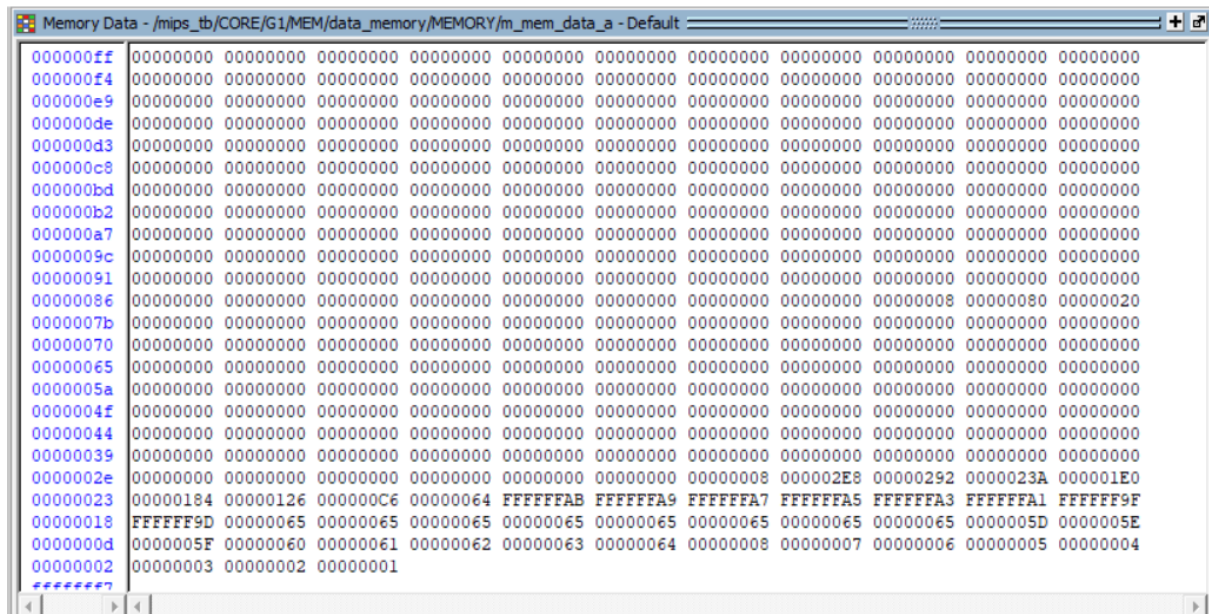
:Model sim and Quartos simulation

ביצענו כמה סימולציות בכדי לבדוק את תקינות המערכת.
להלן test 3 שהועלה לנו כדוגמא:

חלק מה-Wave



תוצאת ה-DTCM:



בנוסף ביצענו טסטים ב-Quartus שרצות על ה-FPGA.
להלן התוצאות ב-DTCM:

test3:

Instance 1: DTCM												
000000	00 00 00 01	00 00 00 02	00 00 00 03	00 00 00 04	00 00 00 05	00 00 00 06	00 00 00 07	00 00 00 08	00 00 00 09	00 00 00 0A	00 00 00 0B	00 00 00 0C
000008	00 00 00 64	00 00 00 63	00 00 00 62	00 00 00 61	00 00 00 60	00 00 00 5F	00 00 00 5E	00 00 00 5D	00 00 00 5C	00 00 00 5B	00 00 00 5A	00 00 00 59
000010	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65	00 00 00 65
000018	FF FF FF 9D	FF FF FF 9F	FF FF FF A1	FF FF FF A3	FF FF FF A5	FF FF FF A7	FF FF FF A9	FF FF FF AB	FF FF FF AD	FF FF FF AF	FF FF FF B1	FF FF FF B3
000020	00 00 00 64	00 00 00 C6	00 00 01 26	00 00 01 84	00 00 01 E0	00 00 02 3A	00 00 02 92	00 00 02 E8	00 00 03 40	00 00 03 98	00 00 03 F0	00 00 04 38
000028	00 00 00 08	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000038	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000040	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000048	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000050	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000058	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00
000060	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

SIGNAL TAP :

log: Trig @ 2025/07/09 15:57:17 (00)		click to insert time bar																											
Type	Alias	Name	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308										
		INSTCNT_o[15..0]	0124h	0125h	0126h	0127h	0128h	0129h	012Ah	012Bh	012Ch	012Dh	012Eh	012Fh	0130h	0131h	0132h	0133h											
		CLKCNT_o[15..0]	0124h	0125h	0126h	0127h	0128h	0129h	012Ah	012Bh	012Ch	012Dh	012Eh	012Fh	0130h	0131h	0132h	0133h											
		STCNT_o[15..0]				0017h																							
		FHCNT_o[15..0]				0017h																							
		BPADD[7..0]													0018h														
		STRIGGER_o																											
		IFpc_o[9..0]	107h	108h	10Fh	113h	117h	11Bh	11Fh	0E7h	0E8h	0EFh	103h	107h	10Bh	10Fh	113h												
		IFInstruction_o[31..0]	ADA80000b/216B0004b/218C0004b/21AD0004b/21CEFFFEh/15COEFF6h	23B00010h	BD6F0000b/BD980000b/BD990000b/71F84002b/ADA80000b/216B0004b/218C0004b/21AD0004b																								
		IDpc_o[9..0]	104h	108h	10Ch	110h	114h	118h	11Ch	3ECh	0E8h	0FCh	100h	104h	108h	10Ch	110h												
		IDInstruction_o[31..0]	71F84002b/ADA80000b/216B0004b/218C0004b/21AD0004b/21CEFFFEh	15COEFF6h	00000000b/BD6F0000b/BD980000b/BD990000b/71F84002b/ADA80000b/216B0004b/218C0004b																								
		EXpc_o[9..0]	100h	104h	108h	10Ch	110h	114h	118h	3ECh	11Ch	3ECh	0E8h	0FCh	100h	104h	108h	10Ch											
		EXInstruction_o[31..0]	BD990000b/71F84002b/ADA80000b/216B0004b/218C0004b/21AD0004b/21CEFFFEh/00000000b/15COEFF6h/00000000b/BD6F0000b/BD980000b/BD990000b/71F84002b/ADA80000b/216B0004b																										
		MEMpc_o[9..0]	0FCh	100h	104h	108h	10Ch	110h	114h	118h	3ECh	11Ch	3ECh	0E8h	0FCh	100h	104h	108h											
		...Minstruction_o[31..0]	BD980000b/BD990000b/71F84002b/ADA80000b/216B0004b/218C0004b/21AD0004b/21CEFFFEh/00000000b/15COEFF6h/00000000b/BD6F0000b/BD980000b/BD990000b/71F84002b/ADA80000b																										
		WBpc_o[9..0]	0FCh	0FCh	100h	104h	108h	10Ch	110h	114h	118h	3ECh	11Ch	3ECh	0E8h	0FCh	100h	104h											
		WBInstruction_o[31..0]																											

מעבר מקובץ C לקובץ אסמבלי והטסט שלו:

קיבלנו קובץ C שמחבר ערכים של 2 מטריצות למטריצה שלישית , תרגמנו אותו לקובץ אסמבלי וביצענו עליו טסט ב Quartus .

קובץ C :

```
#define M 4

void addMats(int Mat1[M][M], int Mat2[M][M], int resMat[M][M]){
    define it yourself ...
}

void main(){ //int=32bit
    int Mat1[M][M]={1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16};
    int Mat2[M][M]={13,14,15,16},{9,10,11,12},{5,6,7,8},{1,2,3,4};
    int resMat[M][M];

    addMats(Mat1,Mat2,resMat); // resMat = Mat1 + Mat2
}
```

קובץ אסמבלי :

```
.data
arr1: .word 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
arr2: .word 13,14,15,16,9,10,11,12,5,6,7,8,1,2,3,4,
res1: .space 64 # SIZE*4=32[Byte] - ADD result array
SIZE: .word 16

.text
.globl main

main:
    li $sp,0x01FC # stack initial address is 200
    lw $s0,SIZE($0) # s0 = SIZE
    la $t1,arr1 # t1 points to arr1
    la $t2,arr2 # t2 points to arr2
    la $s1,res1 # s1 points to res
loop:
    addi $sp,$sp,-16
    sw $s0,12($sp) # push SIZE
    sw $s1,8($sp) # push res1 pointer
    sw $t2,4($sp) # push arr2 pointer
    sw $t1,0($sp) # push arr1 pointer
    jal mat_add

finish: beq $zero,$zero,finish
```

פלט :

Instance 1: DTCM																								
000000	00	00	00	01	00	00	00	02	00	00	00	03	00	00	00	04	00	00	00	05	00	00	00	06
000006	00	00	00	07	00	00	00	08	00	00	00	09	00	00	00	0A	00	00	00	0B	00	00	00	0C
00000c	00	00	00	0D	00	00	00	0E	00	00	00	0F	00	00	00	10	00	00	00	0D	00	00	00	0E
000012	00	00	00	0F	00	00	00	10	00	00	00	09	00	00	00	0A	00	00	00	0B	00	00	00	0C
000018	00	00	00	05	00	00	00	06	00	00	00	07	00	00	00	08	00	00	00	01	00	00	00	02
00001e	00	00	00	03	00	00	00	04	00	00	00	0E	00	00	00	10	00	00	00	12	00	00	00	14
000024	00	00	00	0E	00	00	00	10	00	00	00	12	00	00	00	14	00	00	00	0E	00	00	00	10
00002a	00	00	00	12	00	00	00	14	00	00	00	0E	00	00	00	10	00	00	00	12	00	00	00	14

