``

המערכת

בפרויקט הזה נבנה MCU המכיל MIPS ורכיבים פריפריאליים הפועלים בתיאום כמערכת. המעבד שאותו נעשה הינו מעבד SINGLE CYCLE אותו בנינו במעבדה 5 לצורך הוספת הפקודות התומכות ב ISAשל MIPS לפני מעבר ל PIPELINE.

הרכיבים הפריפריאליים אותם הוספנו הינם BASIC TIMER,GPIO ו INTERRUPT CONTROLLER.
המערכת מתקשרת על ידי 3 BUS שונים : DATABUS,ADDRESSBUS ו CONROLBUS.

את המערכת נצרוב על בקר אינטל FPGA DE10 STANDART בו נביא לידי ביטוי את הרצת התוכנית על ידי שימוש בלחצנים, מתגים, נורות, לדים וגל PWM.

המתואר על ידי האיור הבא :

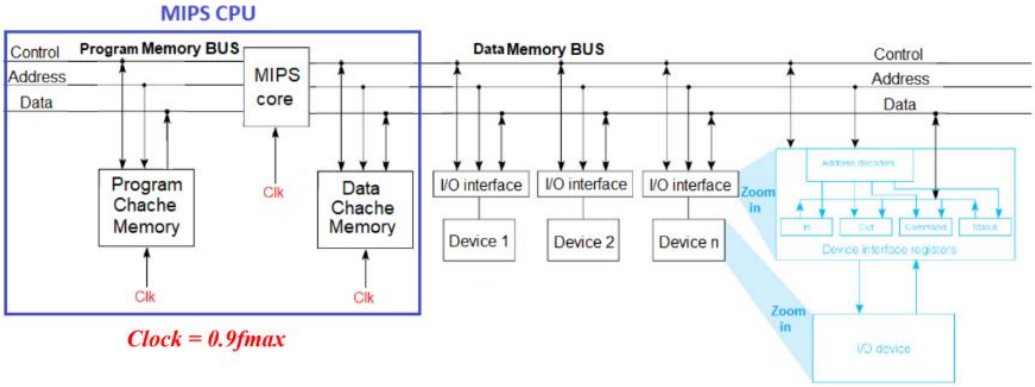
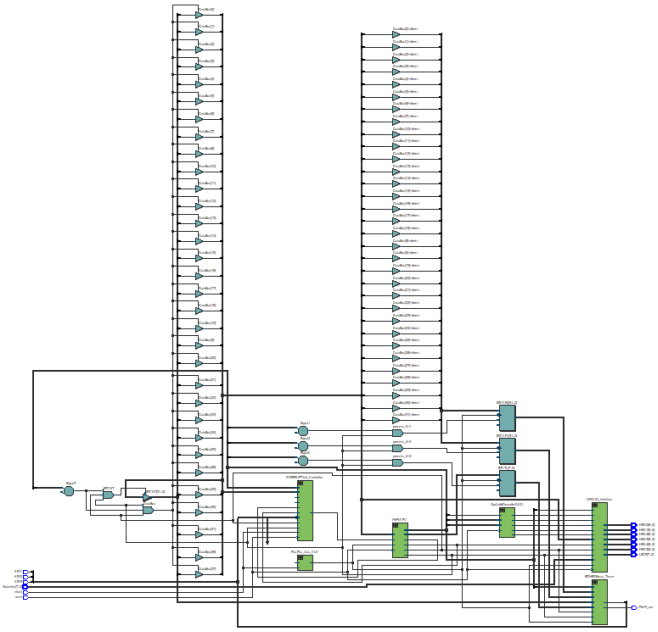


Figure 1 : MCU System architecture

שרטוט ה RTL של המערכת הינו :



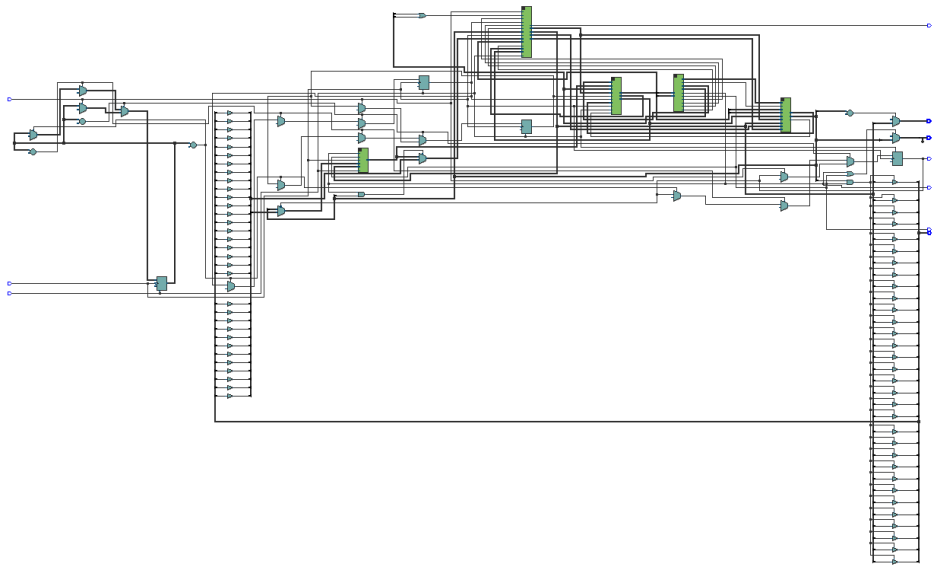
``

החלקים השונים של המערכת:

MIPS

רכיב ה CPU של ה MCU, הינו מעבד SINGLE CYCLE, המבצע את כל שלבי הפעולה תחת אותו מחזור.
כלומר מחזור השעון של המעבד נקבע על ידי הפעולה הארוכה ביותר שהיא פעולת ה LW.
המעבד יודע לתמוך בסט הפקודות אותן ביקשנו להוציא לפועל.

דיאגרמת RTL של הMIPS:



הלוגיקה שביצענו לעיבוד פקודות במעבד מחולקת לפי שלבי עיבוד הפעולה שהן: FETCH, DECODE,EXECUTE, DEMEMORY, WRITE BACK.

``

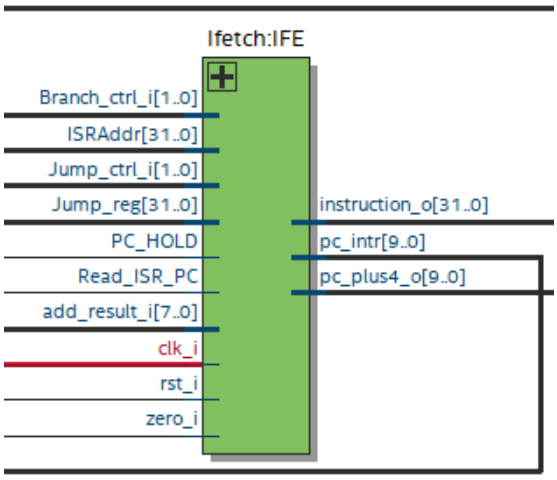
<u>נפרט פה על שלבי הפעולה:</u>

<u>FETCH:</u>

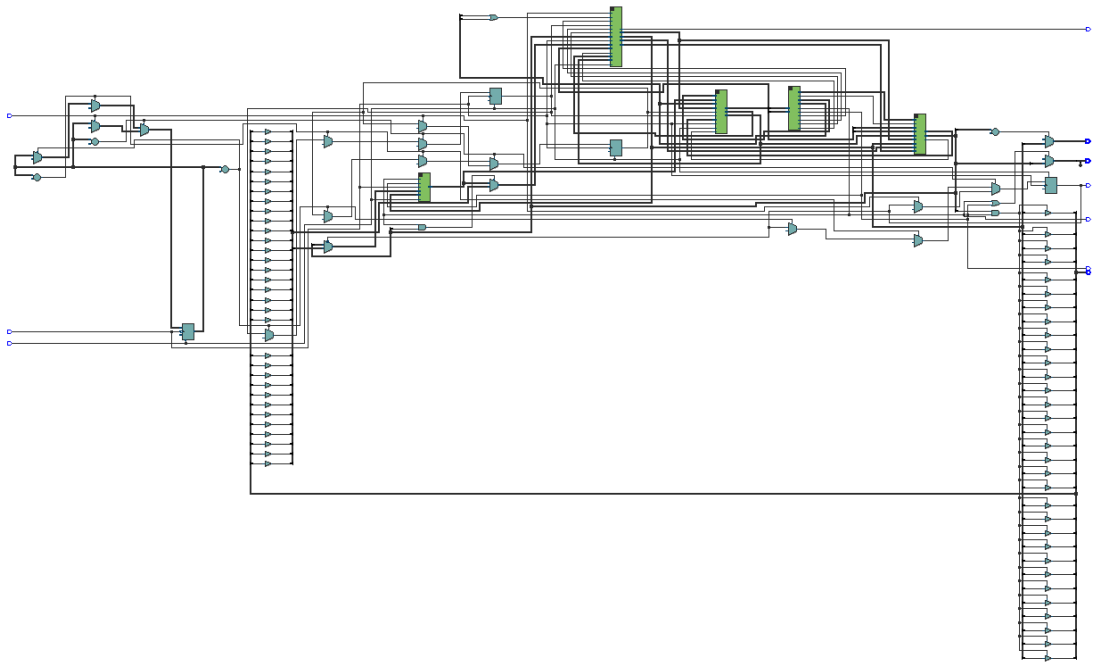
בשלב זה נביא את הפקודה המתאימה מתוך ה־ITCM לפי ערך ה־PC הנוכחי. במקביל, נחשב גם את ערך ה־PC למחזור הבא בעזרת שני MUX:

1. MUX ראשון – בוחר את כתובת ה־PC הבאה מבין:
   - PC + 4 המשך רצף רגיל
   - כתובת מטרה של JUMP
   - כתובת מטרה של BRANCH
2. MUX שני – מקבל כקלט את הפלט של ה־MUX הראשון, ובוחר בין:
   - ה־PC שנבחר ב־MUX הראשון
   - כתובת היעד של פסיקה (Interrupt)

RTL של שלב ה FETCH:



``



DECODE:

בשלב זה נפרק את הפקודה שקיבלנו מה PC לפי סוג הפקודה:

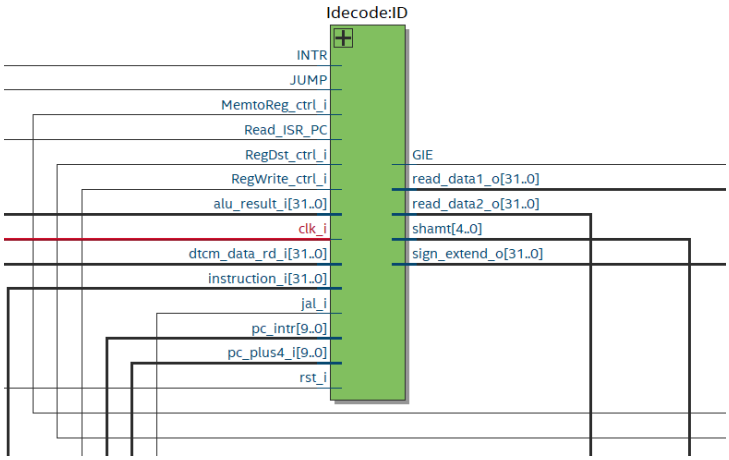
טבלת חלוקת רגיסטרים לפקודות:

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)				

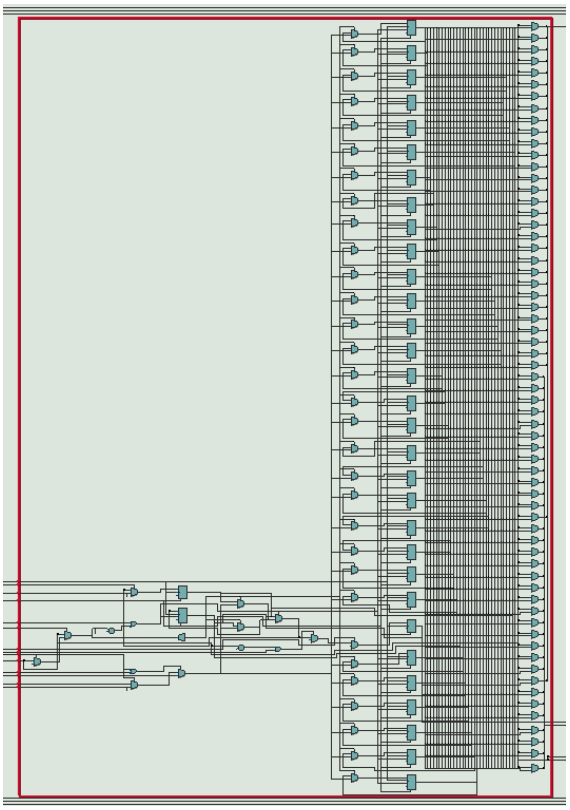
את הרגיסטרים והקבועים שנוציא מהפקודה שקיבלנו נעביר לשלב ה EXECUTE.

בשלב זה גם נקבע את ערך ה GIE.

RTL של שלב ה DECUDE:



``



EXECUTE:

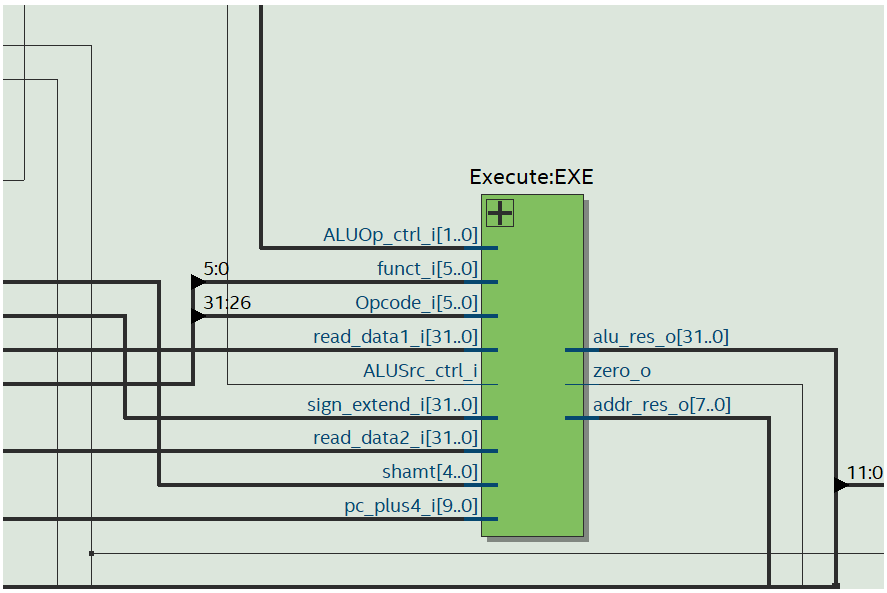
בשלב זה המעבד מחשב את החישוב הנדרש מהפקודה, פעולות אריתמטיות, לוגיות והזזה.
במעבד SINGLE CYCLE כל ההוראה מתבצעת במחזור אחד, ולכן שלב ה־EXאחראי לחשב את התוצאה לפני המעבר לשלב ה־MEMאו ל־WB.

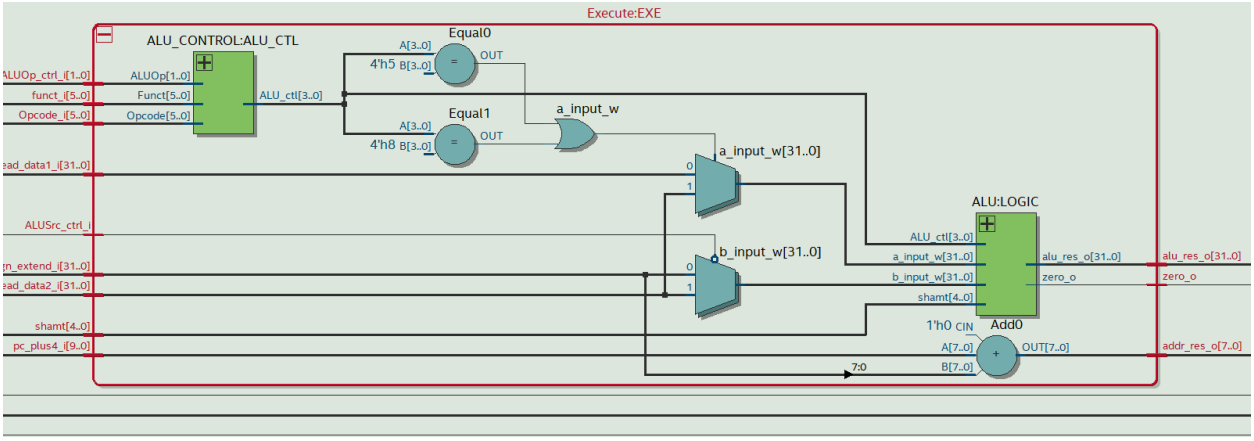
הפעולות שמתבצעות בשלבEX

1. חישוב כתובת לזיכרון (Load/Store)
2. ביצוע חישוב אריתמטי/לוגי (R-type / I-type)

RTL שלב ה EX:

``





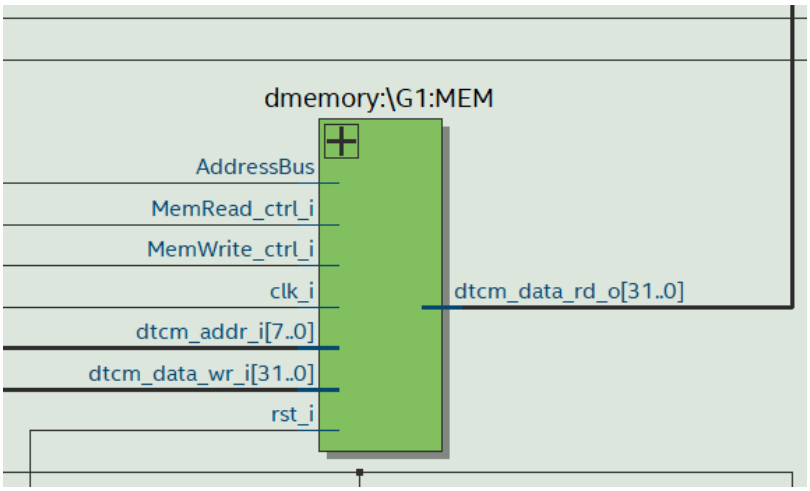
שלב הMEMORY:

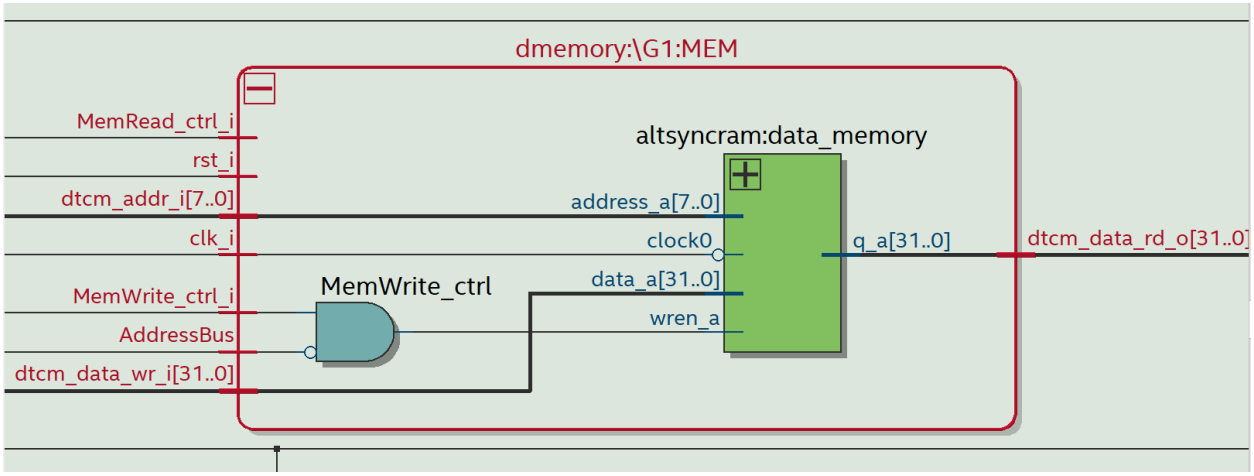
בשלב זה המעבד ניגש לזיכרון לביצוע פעולות על ידי כתובת מתאימה לערך אותו נרצה להביא מהזיכרון ולהעביר באופן מתאים לביצוע הפעולה המתבקשת.

את הגישה ל DTCM נבצע על ידי שימוש ב altsyncram.

RTL של שלב ה MEM:

``





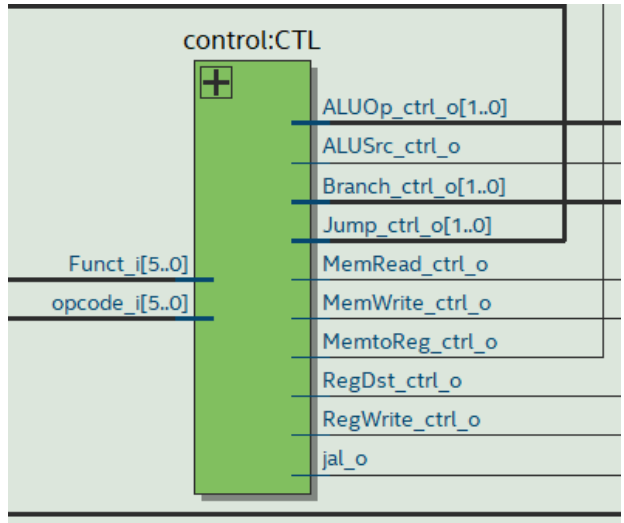
WB:

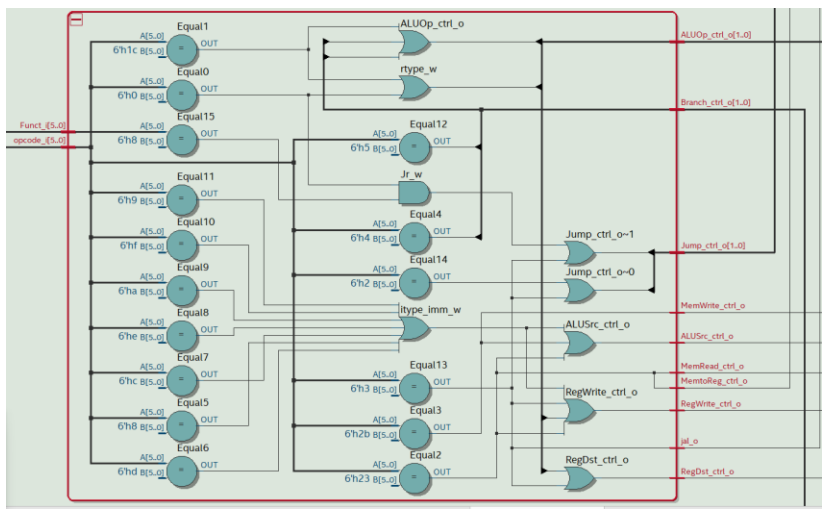
שלב כתיבת הערך לזיכרון או לרגיסטר המתאים.

CONTROL:

יחידה המעלה את כל קווי הבקרה הנדרשים לפעילות התקינה של הפקודות במעבד לפי סוג הפקודה שצריכה להתבצע.

``

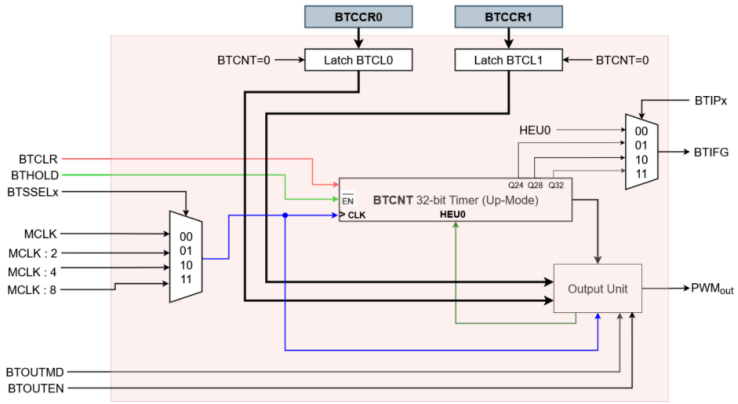

control:CTL
ALUOp_ctrl_o[1..0]
ALUSrc_ctrl_o
Branch_ctrl_o[1..0]
Jump_ctrl_o[1..0]
Funct_i[5..0]
opcode_i[5..0]
MemRead_ctrl_o
MemWrite_ctrl_o
MemtoReg_ctrl_o
RegDst_ctrl_o
RegWrite_ctrl_o
jal_o

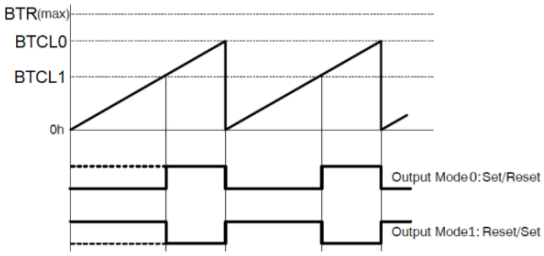

Funct_i[5..0]
opcode_i[5..0]
Equal1
Equal0
Equal15
Equal11
Equal10
Equal9
Equal8
Equal7
Equal5
Equal6
Equal12
Equal4
Equal14
Equal13
Equal3
Equal2
ALUOp_ctrl_o
rtype_w
Jr_w
itype_imm_w
Jump_ctrl_o~1
Jump_ctrl_o~0
ALUSrc_ctrl_o
RegWrite_ctrl_o
RegDst_ctrl_o
ALUOp_ctrl_o[1..0]
Branch_ctrl_o[1..0]
Jump_ctrl_o[1..0]
MemWrite_ctrl_o
ALUSrc_ctrl_o
MemRead_ctrl_o
MemtoReg_ctrl_o
RegWrite_ctrl_o
jal_o
RegDst_ctrl_o

``

רכיבים פריפריאליים:

BASIC TIMER:





רכיב זה סופר על סמך עליות שעון של CLK השונה מ MCLK בעקבות חלוקה בערך קבוע DIV היוצר שעון איטי יותר בפקטור של 2, 4,8.

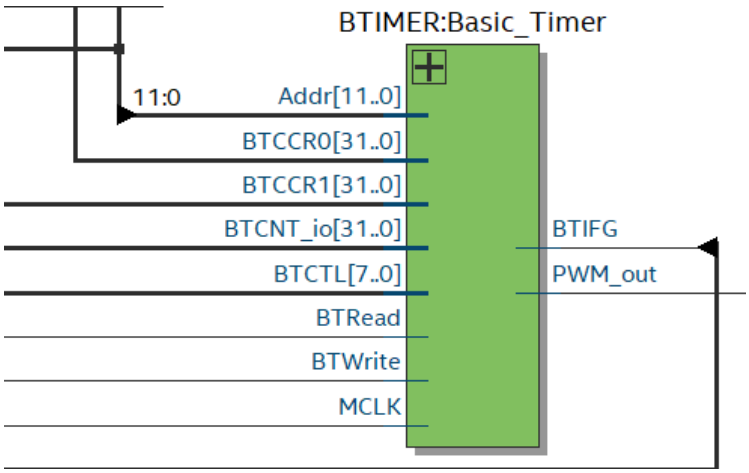
רכיב זה מחזיק רגיסטר בקרה בו שמורים כל אותות הבקרה הנדרשים לרכיב זה כמו CLR,HOLD וכו.. בכתובת ידוע מראש המוקצת לרגיסטר זה.

הערכים הנכנסים לרכיב זה הינם גם שמורים בכתובת ידועה בזיכרון והם יהוו לנו המונים של הרכיב שמייצרים את הגל הריבועי.

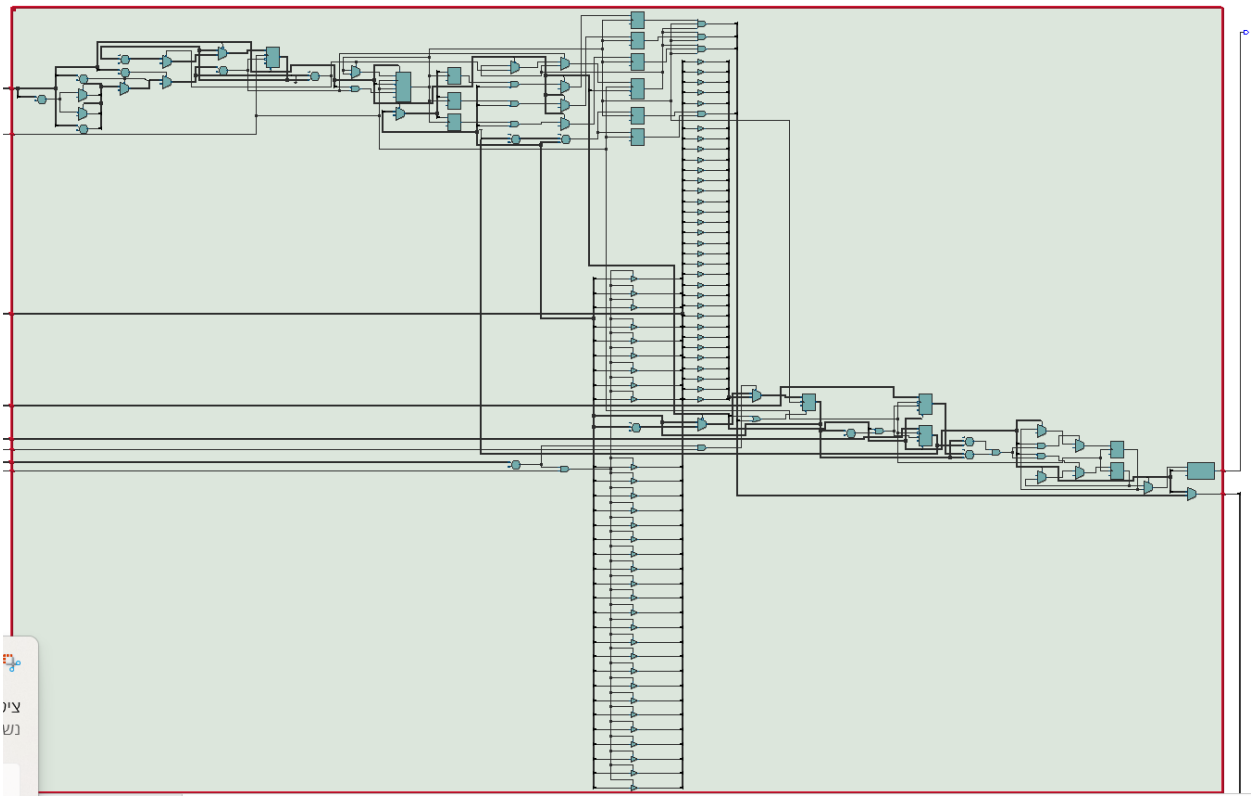
ברכיב זה יש מונה BTCNT המונה עד CCR0 ובהגעה ל CCR1 או CCR0 משתנה מוצא ה PWM כל בעצם ניצור גל ריבועי.

בנוסף ברכיב זה יש אות המודיע על INTERRUPT על פי MUX המוגדר במטלה.

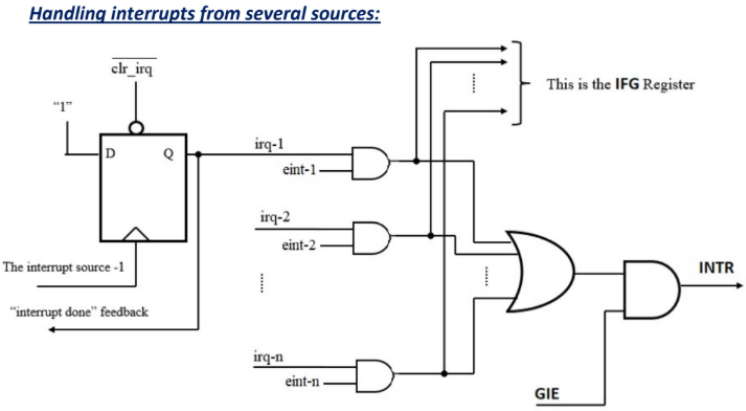
דיאגרמת RTL של רכיב זה:



``



:INTERRUPT CONTROLLER



רכיב זה הינו רכיב בקרה האחראי על הפסיקות במעבד. רכיב זה תומך בפסיקות חיצוניות שבאות מרכיבים פריפריאלים ומתועדפות לפני סדר חשיבות שהוגדר מראש. פסיקה מתבצעת כאשר יש בקשה לפסיקה וגם GIE=1 כלומר מאופשרת קבלת פסיקה במערכת.

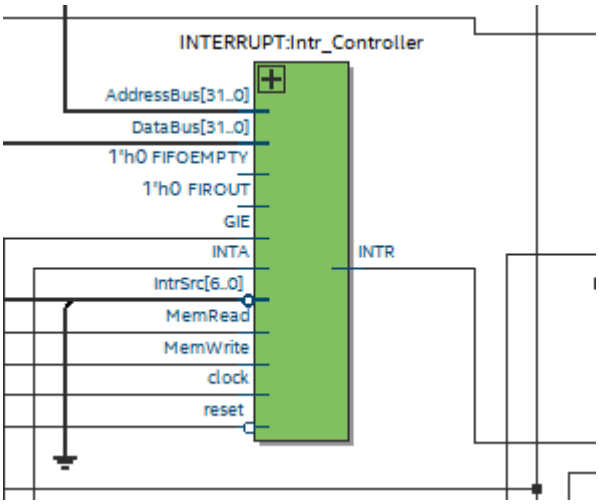
TYPE Contents	Interrupt Source	Interrupt Flag	Interrupt Priority	
00h	RESET	NMI	Highest	(Non)-Maskable Interrupt
04h	UART status error	RXIFG		Maskable Interrupt
08h	UART RX			
0Ch	UART TX	TXIFG		
10h	Basic Timer	BTIFG		
14h	KEY1	KEY1IFG		
18h	KEY2	KEY2IFG		
1Ch	KEY3	KEY3IFG		
20h	FIFOEMPTY	FIRIFG	Lowest	
24h	FIROUT			

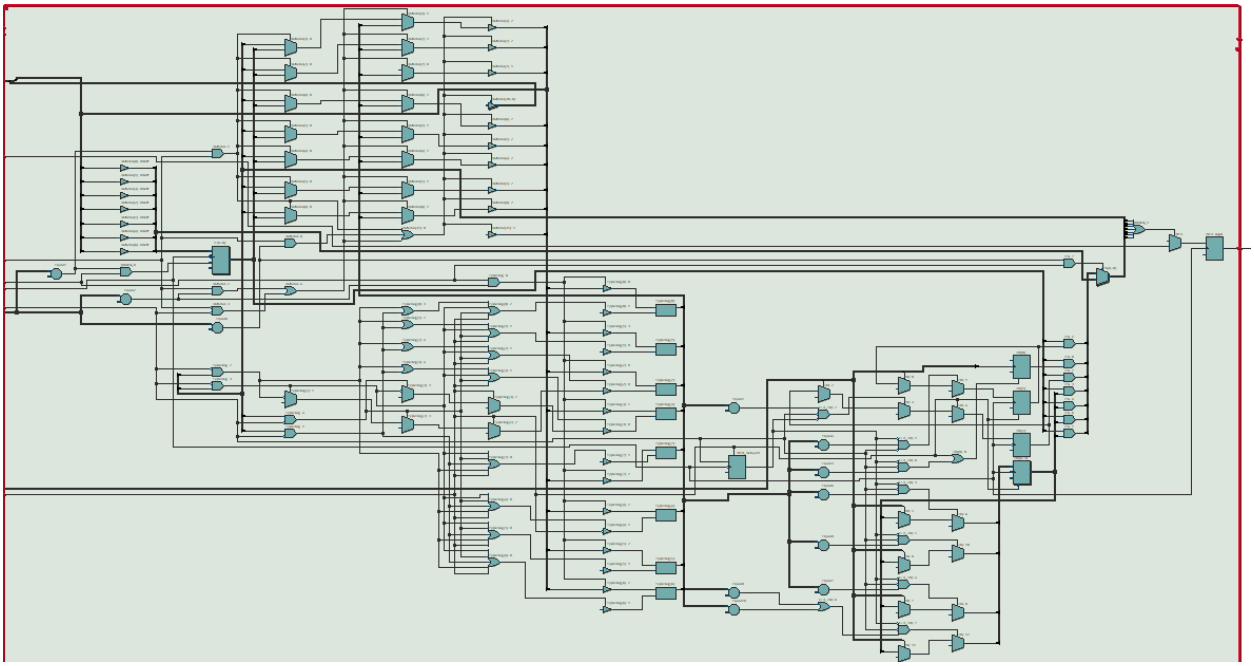
הסיגנל שמודיע על פסיקה הינו סיגנל IRQ, ואת הפסיקה שתתבצע נראה ב IFG שזה לאחר בדיקת גם GIE.

``

בקבלת פסיקה הMIPS מקבל בקשה לפסיקה ומחזיר ACK. לאחת קבלת ACK הPC יקפוץ לכתובת המתאימה ל INTERRUPT לצורך ביצוע סט הפקודות הנדרשות.

RTL של רכיב זה:

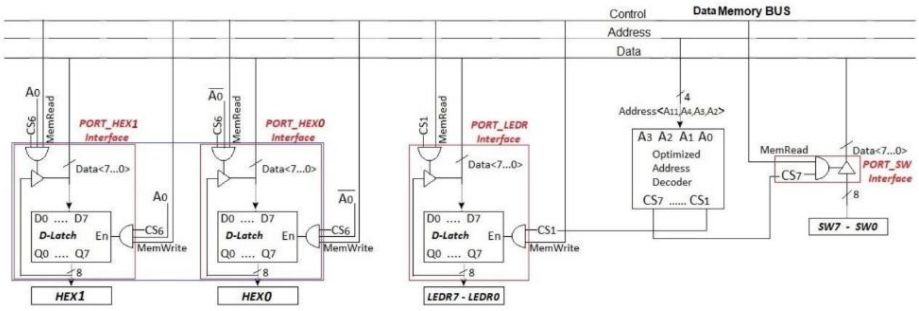




``

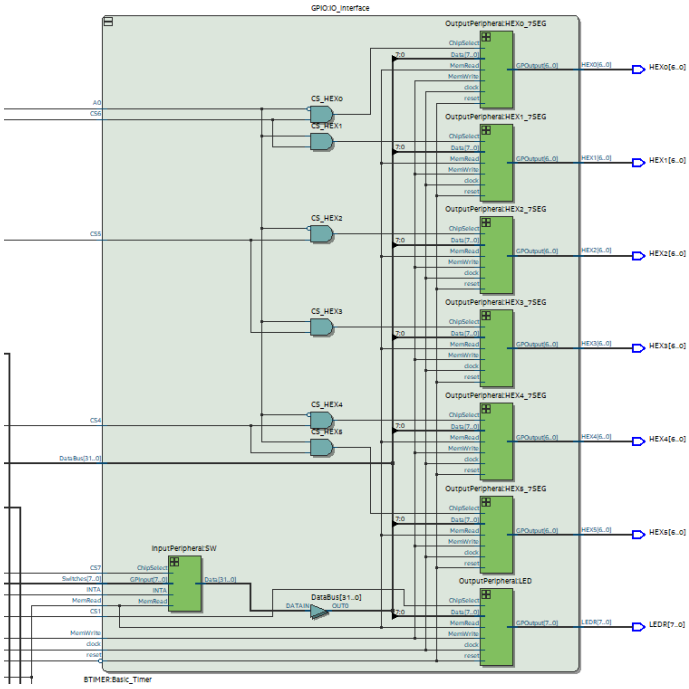
GPIO:

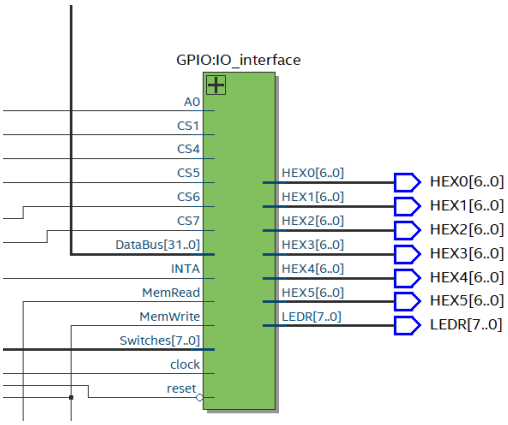
הגדרת הפעולה של הרכיב:



כדי להוסיף ממשק למשתמש, הוספנו למערכת שימוש בכפתורים, מתגים, לדים ונורות. כך ש KEY0 מהווה ה RESET של המערכת ושאר הכפתורים משתמשים ליצירת INTERRUPT. המתגים מהווים INPUT למערכת ועל הלדים נוכל להציג את תוצאות הפקודות המתבצעות במעבד.

דיאגרמת ה RTL של רכיב זה:

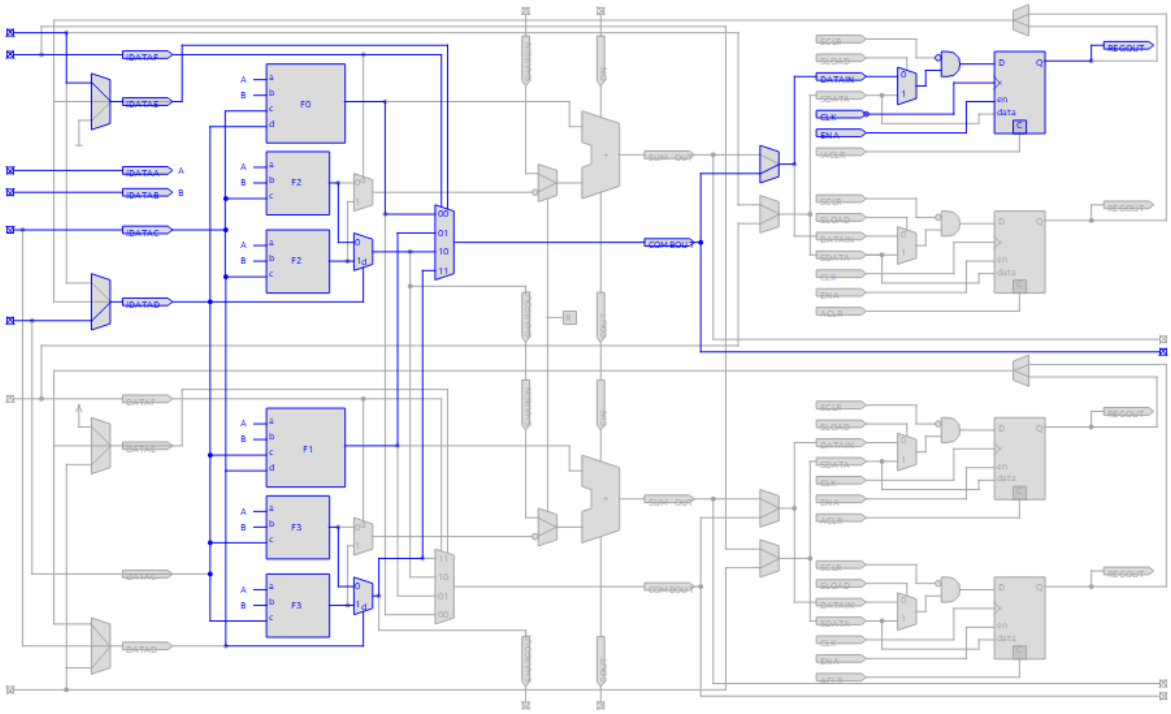


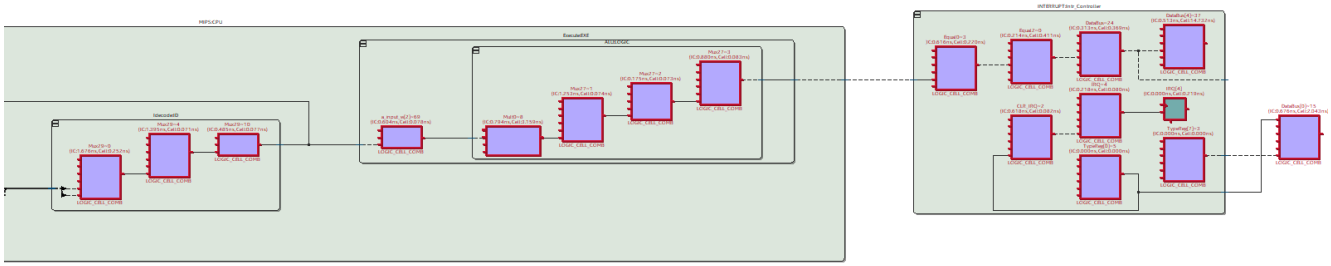


``

נתיב קריטי:

הנתיב הקריטי של המערכת הוא נתיב ההתפשטות הארוך ביותר בתוך המעבד.





הנתיב הקריטי מחבר בין יחידות שלב ה־ Execute/Decode ועד למודול ה־Interrupt Controller.

זה אומר שהשילוב של לוגיקת ALU/Decode + לוגיקת Interrupt לדוגמה זיהוי IRQ, שמירת PC, כתיבה לרגיסטרים יצר עיכוב גדול במיוחד.

במעבדים במיוחד עם Interrupt Controller החיבורים האלה לפעמים נהיים "עמוסים" בלוגיקה:

- הרבהMUX ים.
- בדיקות תנאי לדוגמה IRQ, CLR_IRQ, ENABLE .

``

- כתיבה/קריאה לרגיסטרים פנימיים.

כל זה נערם לנתיב לוגי ארוך שזהו הנתיב הקריטי.

היינו משפרים את הנתיב הקריטי על ידי מעבר לPIPELINE : חציצת המערכת על ידי רגיסטרים יגרום למערכת להתנהל לפי מחזור שעון מקסימלי בין רגיסטרים. כלומר, חציצה שכזאת תגרום למחזור שעון להיות קצר יותר מאשר של מעבד ה SINGLE CYCLE. כך, שלמרות שלכל פעולה יהיו יותר מחזורי שעון זמן התוכנית הכולל יהיה קצר יותר מכיוון שהפעולות יעבדו במקביל.

בנוסף, היינו מנסים לייעל את הנתיב הקריטי של המערכת על מנת לצמצם MUXים לדוגמה.

התדר המקסימלי של המערכת הינו:

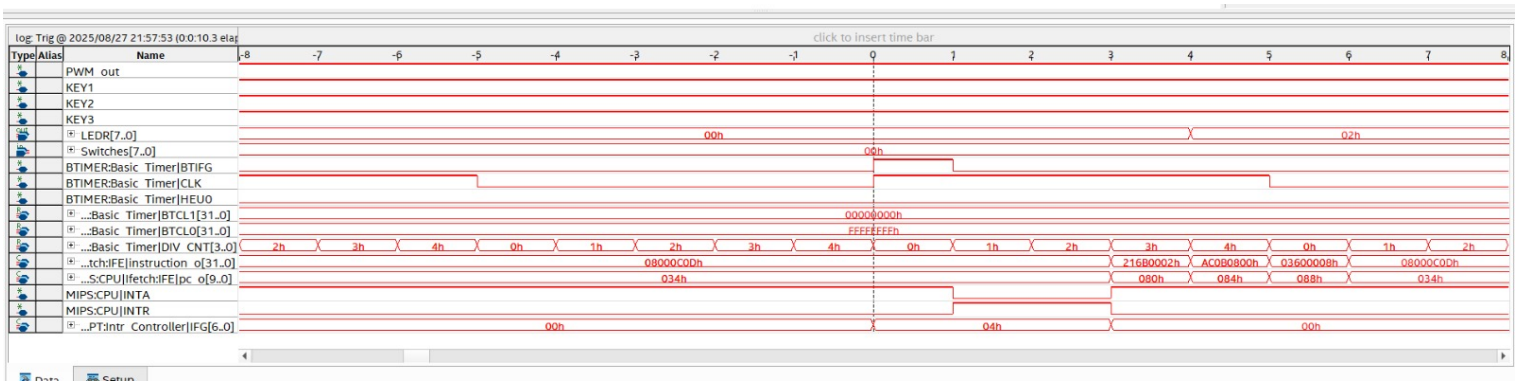
Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	16.2 MHz	16.2 MHz	PLL_ALL_C...ER|divclk	
2	32.8 MHz	32.8 MHz	clock	
3	90.18 MHz	90.18 MHz	altera_reserved_tck	

ניתוח תוצאות:

לאחר כתיבת קבצי VHDL העלנו את הקבצים לתוכנת ה QUARTUS שם הרצנו טסטים בעזרת SIGNAL TAP להלן הניתוח ביצועים של הטסטים שהרצנו.



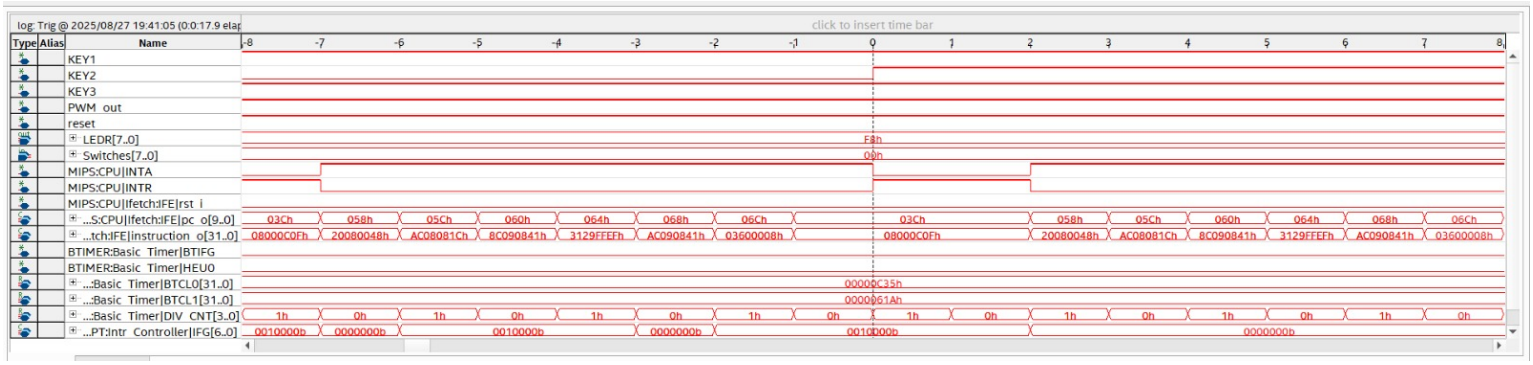
טסט 3 –

כפי שניתן לראות עבור טסט זה הגדרנו את הטריגר כעליית השעון של האינטרפט המגיע מהטיימר, בנקודה זו בה הביט ה-24 מתהפך עולה האינטרפט ובעקבותיו ניתן לראות כי ערך הלדים גדל ב-2 בעקבות שגרת האינטרפט.

ניתן לראות כי הפעולה המבוצעת על פי שורת ה-instruction היא לולאה אינסופית עד אשר מגיע האינטרפט

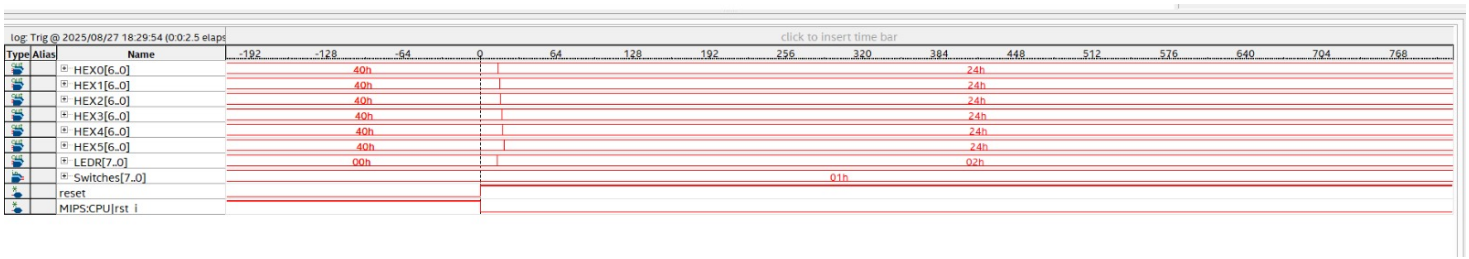
``

ומוציא לשגרה ובנוסף ניתן לראות את בקשת האינטרפט והמענה אליו בסיגנלים : INTR,INTA מגיבים בהתאם ואת וקטור ה-IFG משתנה כך שנדלק הביט השלישי שמציין את הטיפול באינטרפט מהטיימר.



טסט 2 –

בטסט זה הוצאנו גל מחזורי מהבקר על פי תדר שנקבע לפי האינטרפט, ניתן לראות כי מפעולת הלולאה האינסופית ישנה קפיצה לשגרת האינטרפט ותדר השעון משתנה לחצי מהשעון הרגיל (שלפני כן היה שמינית).
בנוסף ניתן לראות את הסיגנלים הרלוונטים לטיפול באינטרפט כמו IFG,INTA,INRT נדלקים ונכבים בהתאם.



טסט 2 (GPIO)–

ניתן לראות איך על פי טסט זה בכל איטרציה לאחר דילאי קופץ ערך הלדים והhex ב-2 לפי ה-switch, ניתן לראות בתמונה זו את האיטרציה הראשונה ובגלל הדילאי ההגדול לא מופיעה איטרציה נוספת במסך.
הטריגר נתפס בעזרת ה-reset.

` `

<u>טסט 2 (GPIO)–</u>

ניתן לראות את הראשון בו מבצעת ספירה מחזורית על גבי הלדים וה-HEX , בנוסף בטסט זה הורדנו את ערך הדילאי על מנת שנוכל לראות כמה שיותר שינויים על גבי הרכיבים הפריפריאלים.