ADVANCED CPU ARCHITECTURE AND HARDWARE ACCELERATORS LAB

FINAL PROJECT

MIPS BASED MCU ARCHITECTURE

361.1.4693

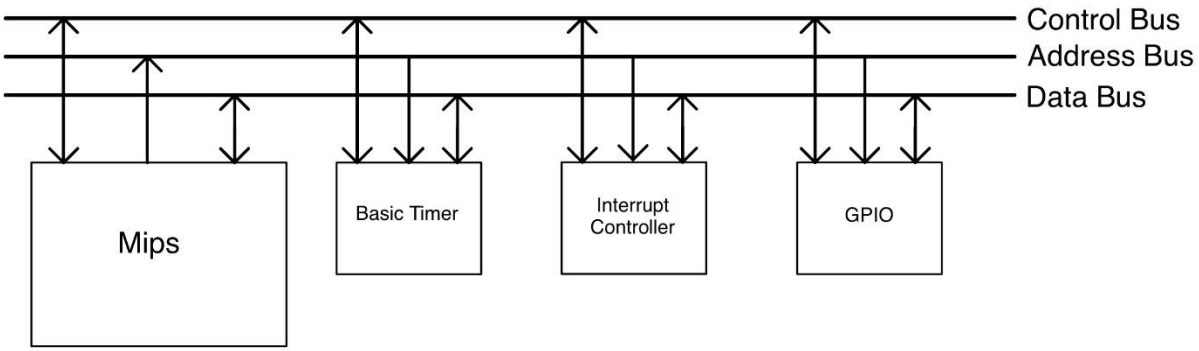
2024 SEM B

Nachman Mimoun 321730558

Danel Barsheshet 209471242

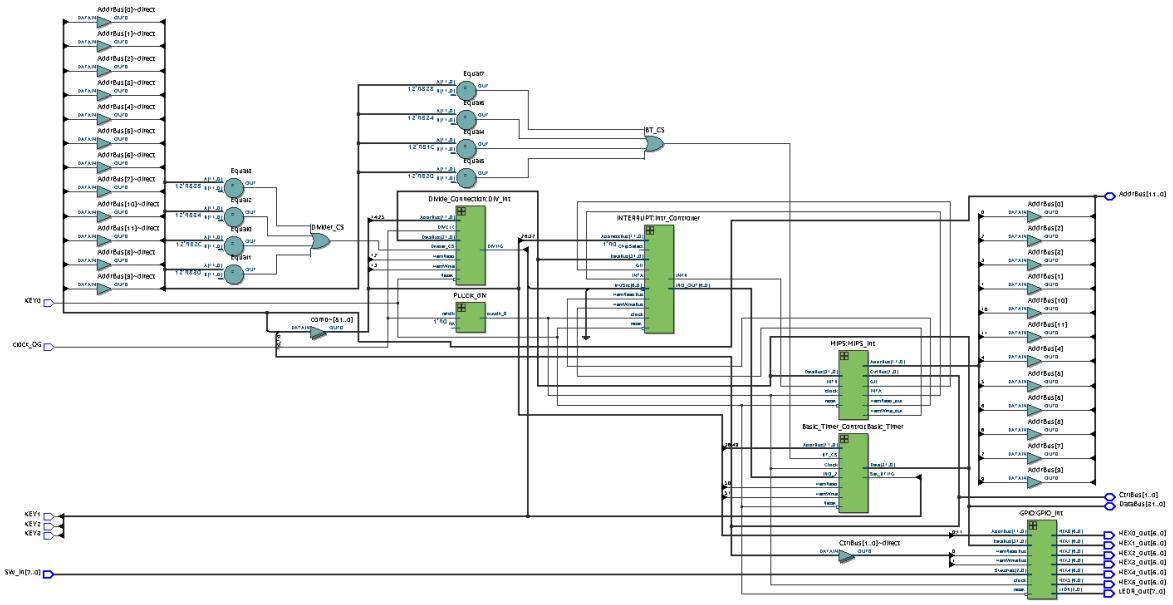
תוכן עניינים:

בפרויקט זה נרצה לבנות MCU המבוסס על מעבד MIPS בעל מחזור יחיד(Single Cycle).
המעבד יתמוך בסט ההוראות המלא של MIPS פשוט ויכלול יכולות טיפול בפסיקות. בנוסף למעבד, נוסיף מספר פריפריות חומרה כולל GPIO, טיימר בסיסי, מאיץ חלוקה בינארית, ובקר פסיקות.
המערכת תכלול זיכרון תוכנית (ITCM) וזיכרון נתונים (DTCM) בארכיטקטורת Harvard. התקשורת בין רכיבי החומרה השונים תהיה מבוססת על גישת Memory Mapped I/O, כאשר הפריפריות ממופות לכתובות זיכרון מעל לזיכרון הנתונים.
להלן שרטוט המערכת בדיאגרמת בלוקים–



איור 1 שרטוט מערכת MCU

להלן דיאגרמת ה RTL-הכללית של המערכת:



MIPS

רכיב זה הוא ה-CPU של ה-MCU והוא מבוסס על ארכיטקטורת MIPS בעלת מחזור יחיד (Single Cycle). בניגוד לארכיטקטורת פון נוימן, המעבד שלנו פועל לפי ארכיטקטורת Harvard, עם זיכרון נפרד להוראות (ITCM) ולנתונים (DTCM). זה מאפשר גישה מקבילה לזיכרון ההוראות והנתונים, מה שמשפר את הביצועים.
המעבד תומך בסט ההוראות המלא של MIPS פשוט, כפי שמוגדר במסמכי הפרויקט. הוא מסוגל לבצע את כל התוכניות שמשתמשות בהוראות אלו. בנוסף, המעבד כולל יכולת טיפול בפסיקות, מה שמאפשר אינטראקציה יעילה עם הפריפריות השונות במערכת ה-MCU.
אף על פי שזהו מעבד בעל מחזור יחיד, הוא עדיין מסוגל לבצע את כל ההוראות ביעילות. כל הוראה מתבצעת בתוך מחזור שעון אחד, כולל גישה לזיכרון וביצוע פעולות ה-ALU
להלן שרטוט המעבד–

Instruction Fetch

בשלב זה, המטרה היא לאחזר את ההוראה הנמצאת בכתובת המצוינת על ידי ה PC-מזיכרון ההוראות .(ITCM) במקביל, אנו מטפלים בעדכון ה PC-לקראת המחזור הבא. בארכיטקטורת Single Cycle, כל ההחלטות לגבי ה PC-מתקבלות ומבוצעות באותו מחזור שעון.
קיימות שלוש אפשרויות עיקריות לעדכון ה:PC-
PC ← PC + 4: זהו המקרה הסטנדרטי, כאשר אנו מתקדמים להוראה הבאה ברצף.
PC ← Jump Address: במקרה של פקודת jump, ה PC-מתעדכן לכתובת היעד המוגדרת בהוראה.
PC ← Branch Target: אם תנאי ה branch-מתקיים) מה שנקבע בשלב ה ,(Execute-ה PC-מתעדכן לכתובת היעד של ה.branch-
בנוסף, במקרה של פסיקה ,(interrupt) ה PC-יכול להתעדכן לכתובת של שגרת הטיפול בפסיקה (Interrupt Service Routine).
חשוב לציין כי בארכיטקטורת Single Cycle, כל ההחלטות הללו מתקבלות ומבוצעות באותו מחזור שעון, מה שמפשט את התכנון אך עלול להגביל את תדר השעון המקסימלי של המערכת.

Instruction Decode

בשלב זה אנו מפענחים את ההוראה שהתקבלה משלב ה Fetch-למרכיביה על פי פורמט ההוראות של MIPS הבא:
על פי פורמט זה, כתובות הרגיסטרים המתאימים מועברות ל Register File, מה שמאפשר גישה לערכי הרגיסטרים הרלוונטיים. כחלק משלב הפענוח, המעבד מחלץ את שדות ה opcode-וה funct-מתוך ההוראה

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)				

Table 1 : MIPS Instruction format

)כאשר בהוראות שאינן מסוג R-Type, שדה ה funct-אינו רלוונטי .(בהתאם להוראה המפוענחת, יחידת הבקרה מפעילה את אותות הבקרה המתאימים.
בנוסף, בשלב זה מתבצע חישוב כתובות היעד עבור הוראות branch ו jump, אם רלוונטי. חישוב זה כולל השוואה בין תוכן ההוראה לתנאי הקפיצה הספציפי, וחישוב הכתובת החדשה של ה PC-במידת הצורך.

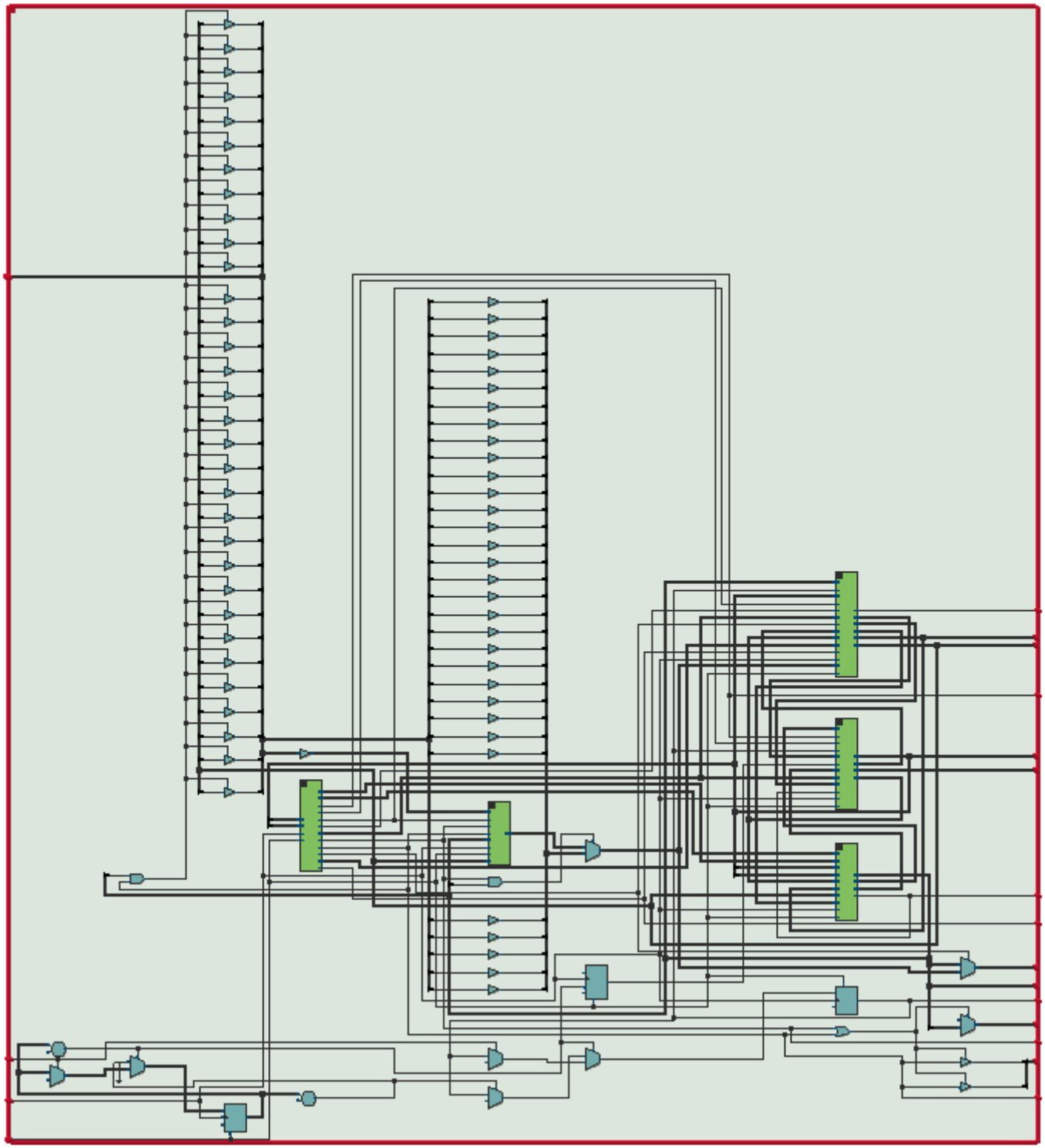
Execute:

בשלב הביצוע (Execute) מתבצעות פעולות כמו חישוב כתובות או ביצוע פעולות אריתמטיות וכו' ברכיב ה-ALU. במעבד ה single-cycle-שלנו, בשלב הביצוע מתבצע גם חישוב כתובת ה PC-החדשה ופעולות ה .branch-
התוצאה מועברת להמשך טיפול - כתיבה לזיכרון או עדכון רגיסטרים.

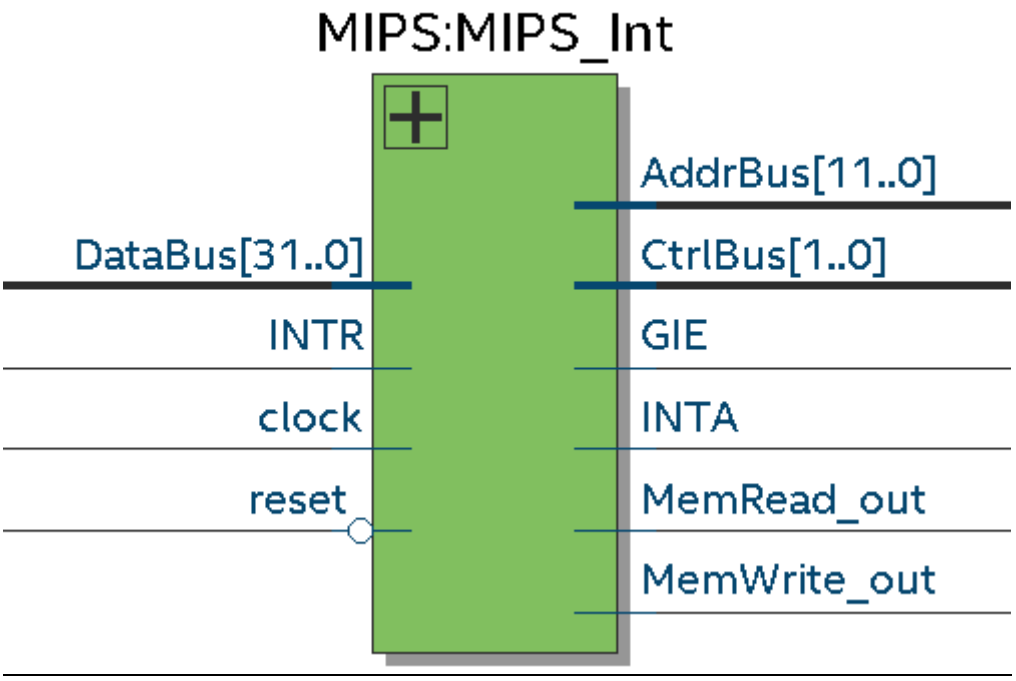
Data Memory: מודול זה אחראי על הכתיבה והקריאה מ .(DTCM) Data Memory-הוא מטפל בגישות לזיכרון הנדרשות על ידי הוראות load ו ,store-ומתקשר עם ה Memory Mapped I/O-לגישה להתקני קלט/פלט.

Write Back: מודול ה Write Back-אחראי על כתיבת התוצאות בחזרה לרגיסטרים או לזיכרון. הוא מחליט איזה מידע) מה ALU-או מה (Data Memory-צריך להיכתב בחזרה לרגיסטר היעד, בהתאם לסוג ההוראה שבוצעה.

RTL Diagram:

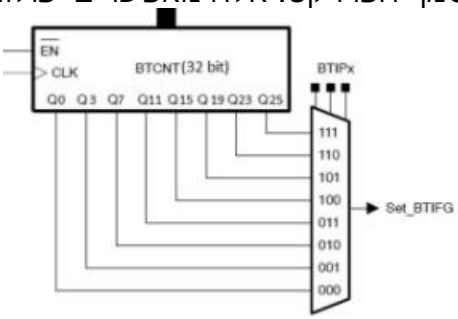


תיאור גרפי של המודל:

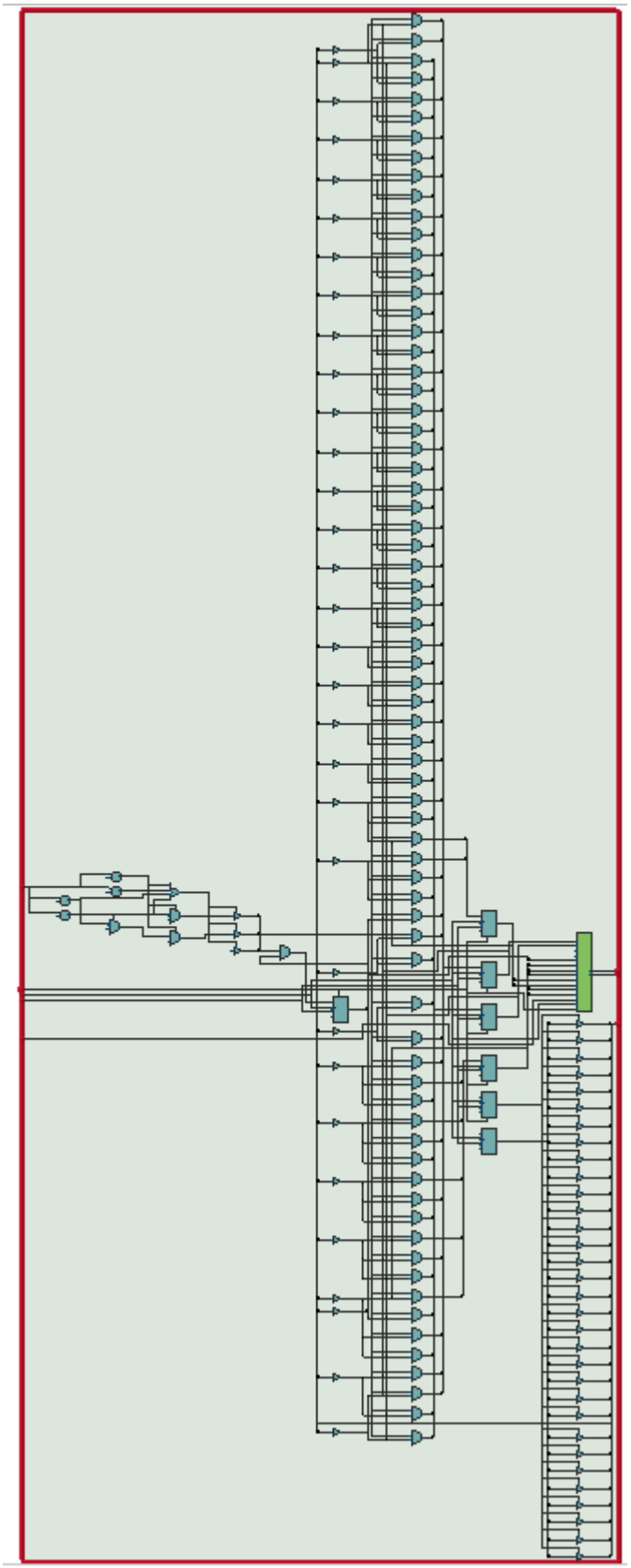


Basic Timer:

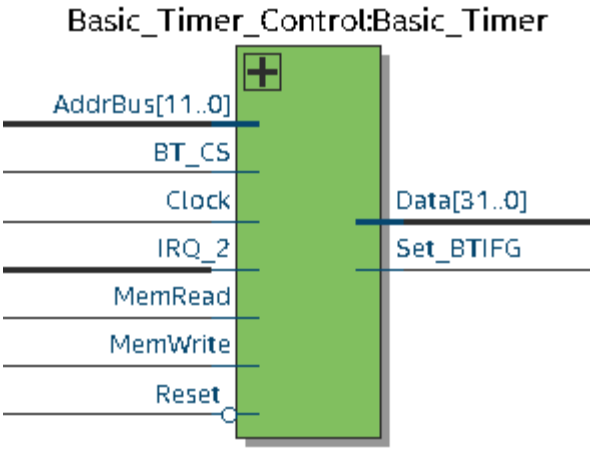
זהו רכיב פריפריאלי חיצוני למעבד, המשמש להרחבת יכולותיו. ה Basic Timer-ממלא שתי מטרות עיקריות:
מונה חומרה פשוט :
פועל על סמך עליות שעון.(clock edges)
כולל קווי בקרה שונים הקשורים להגדרות שצוינו במשימה.
מכיל רגיסטר BTCNT המייצג את ערך הספירה הנוכחי.
ניהול פסיקות :
כולל רגיסטר.(Basic Timer Interrupt Pending) BTIP
ערך ה BTIP-קובע אם תופעל פסיקה, בהתאם לערכי הביטים כמתואר בתרשים הבא:
]כאן תוכל להוסיף את התרשים או הטבלה המתארת את ערכי ה[BTIP-
בנוסף, ה Basic Timer-כולל רגיסטרי השוואה (BTCCRx) ורגיסטרי לכידה (BTCLx) כפי שמתואר בעמוד 7 של מסמך הפרויקט. אלה מאפשרים יכולות השוואה (output compare) מתקדמות.



פעולות נוספות של ה
הפעלת בקשות פסיקה :
הטיימר מסוגל להוציא בקשה לפסיקה ל.Interrupt Controller-
תכונה זו מאפשרת תזמון מדויק של פעולות במרווחי זמן ספציפיים.
יצירת אות :PWM
באמצעות הטיימר הבסיסי ורגיסטרי ההשוואה BTCCR0 ו.BTCCR1-
רגיסטרים אלה מאפשרים לקבוע את זמן המחזור (period) ואת ה duty cycle-של האות.PWM
יכולת זו מרחיבה את השימושיות של ה MCU-לשליטה במנועים, תאורה מתכוננת, וכדומה.
תכונות אלו מדגימות את הגמישות והיעילות של ה Basic Timer-כחלק אינטגרלי מה ,MCU-מאפשרות תזמון מדויק של אירועים ויצירת אותות בקרה מורכבים.
דיאגרמת:RTL



תיאור גרפי של המודל:


Basic_Timer_Control:Basic_Timer
AddrBus[11..0]
BT_CS
Clock
IRQ_2
MemRead
MemWrite
Reset
Data[31..0]
Set_BTIFG

Interrupt Controller:

בפרויקט זה, הוספנו תמיכה בפסיקות חיצוניות מהרכיבים הפריפריאליים. לכל פסיקה הוגדרה רמת דחיפות כפי שצוין במסמך המשימה (עמוד 11-10).

מנגנון הפסיקות:

הפעלת פסיקה: כאשר רכיב פריפריאלי מבקש לבצע פסיקה, סיגנל ה IRQ-של אותו רכיב מופעל.

טיפול בפסיקה: פסיקה תטופל רק אם אין פסיקה אחרת המתבצעת כרגע. אין תמיכה בפסיקות מקוננות.

תיעדוף: פסיקות ממתינות עד לסיום הטיפול בפסיקה הנוכחית.

פרוטוקול כניסה לפסיקה:

שמירת מצב: ערך ה PC-והרגיסטרים החשובים נשמרים בזיכרון) כנראה ב $k0-ו $k1-כמצוין בעמוד 13.(

קפיצה לשגרת הטיפול: המעבד קופץ לכתובת הטיפול בפסיקה המתאימה, כפי שהוגדרה מראש לכל רכיב פריפריאלי.

טיפול בפסיקה: המעבד מבצע את הפעולות הנדרשות לטיפול באירוע שגרם לפסיקה.

חזרה למצב המקורי: בסיום הטיפול, המעבד משחזר את המצב המקורי PC) ורגיסטרים (וממשיך את ביצוע התוכנית מהנקודה בה הופסקה.

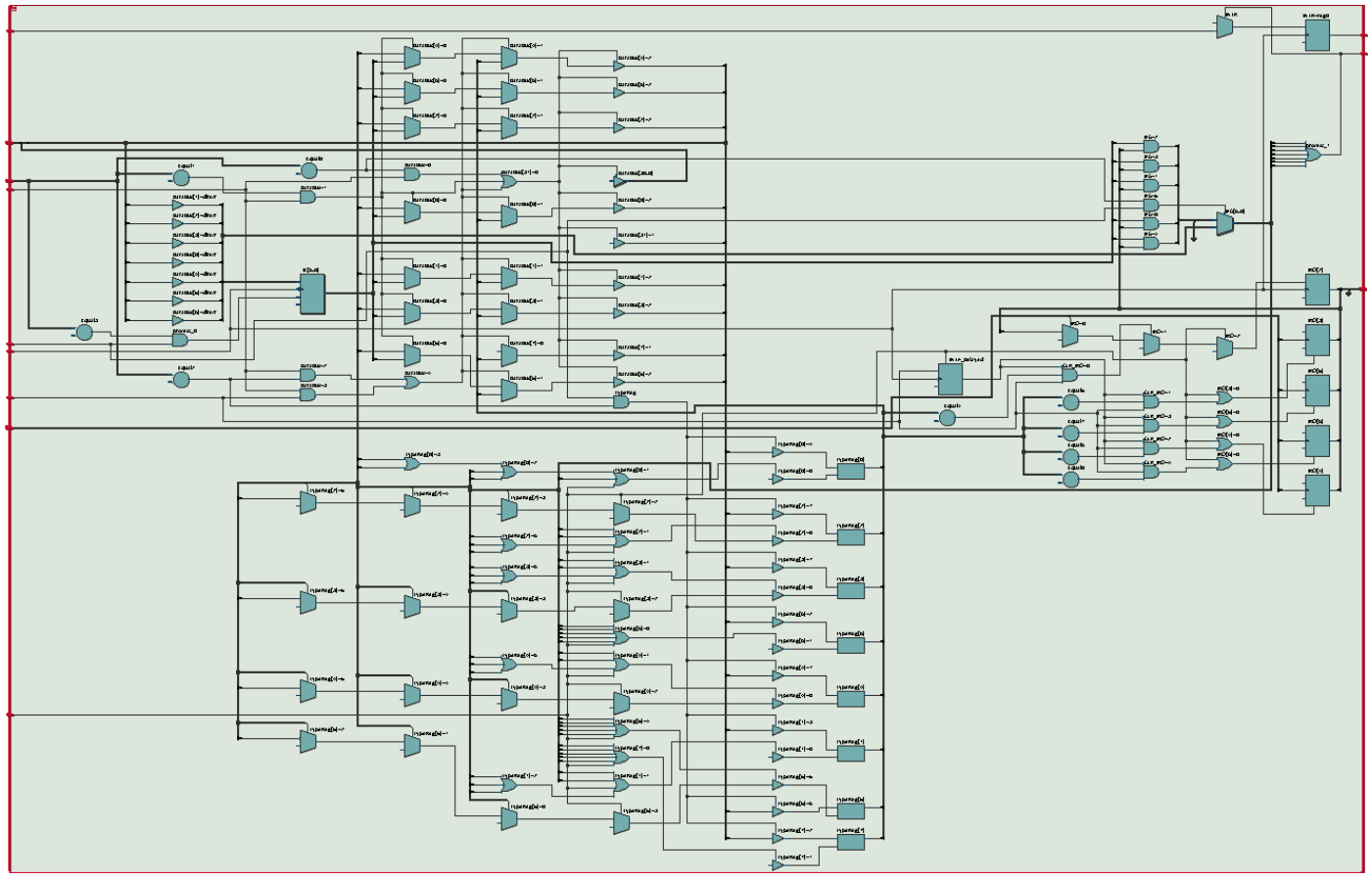
חשוב לציין:

GIE (Global Interrupt Enable)מנוקה בחומרה בעת כניסה לפסיקה, ומופעל בחזרה בעת היציאה ממנה (עמוד 11).

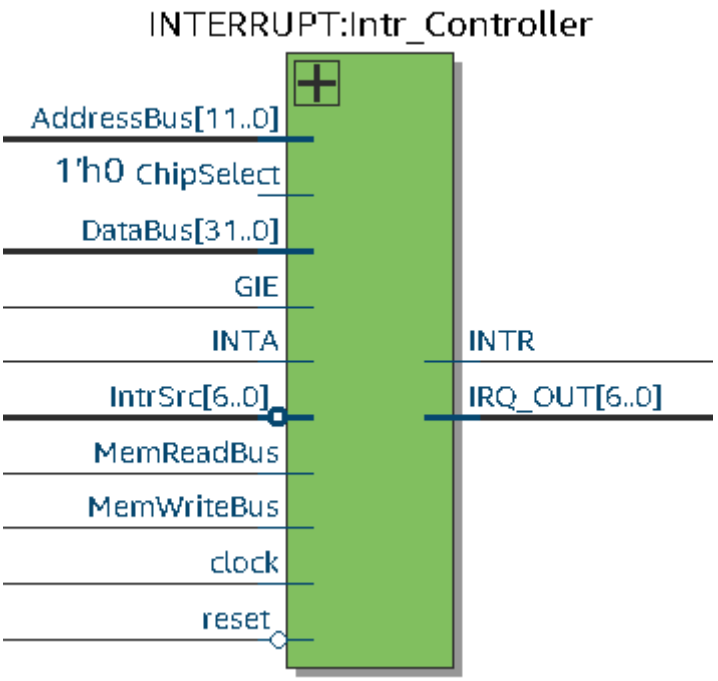
הדגלים BTIFG, DIVIFG, RXIFG, ו TXIFG-מאופסים אוטומטית כאשר הפסיקה מטופלת (עמוד 10).

דגל KEYiIFG מאופס ידנית בתוכנה.

דיאגרמת:RTL



תיאור גרפי של המודל:


INTERRUPT:Intr_Controller
AddressBus[11..0]
1'h0 ChipSelect
DataBus[31..0]
GIE
INTA
IntrSrc[6..0]
MemReadBus
MemWriteBus
clock
reset
INTR
IRQ_OUT[6..0]

GPIO (General Purpose Input/Output):

לצורך הוספת ממשק משתמש, שילבנו במערכת ה MCU-שלנו רכיבי GPIO הכוללים LEDs, תצוגות 7segment-
(הקסות), וכפתורים. מימוש זה תואם את דרישות הפרויקט כפי שמפורט בעמוד 4-5 של מסמך ההגדרות.

מאפייני ה:GPIO-

כניסות :

מערך של 10 מתגים(SW9-SW0)

4כפתורים עם(KEY3-KEY0) debounce

יציאות :

10נורות LED אדומות(LEDR9-LEDR0)

6תצוגות 7segment- (HEX5-HEX0)

מיפוי זיכרון :

הכתיבה והקריאה מרכיבי ה GPIO-מתבצעת באמצעות גישה לכתובות זיכרון ייעודיות.(Memory Mapped I/O)
כתובות אלו ממוקמות מעל אזור ה ,Data Memory-כפי שמוצג באיור 2 בעמוד 4 של מסמך הפרויקט.

תקשורת :

השימוש ברכיבי ה GPIO-נעשה דרך קווי ה BUS-המוגדרים במערכת.

פסיקות :

חלק מרכיבי ה ,GPIO-כמו הכפתורים, יכולים לבקש פסיקות.
לדוגמה, לחיצה על כפתור יכולה לגרום לפסיקה שתטופל על ידי ה.Interrupt Controller-

מימוש :

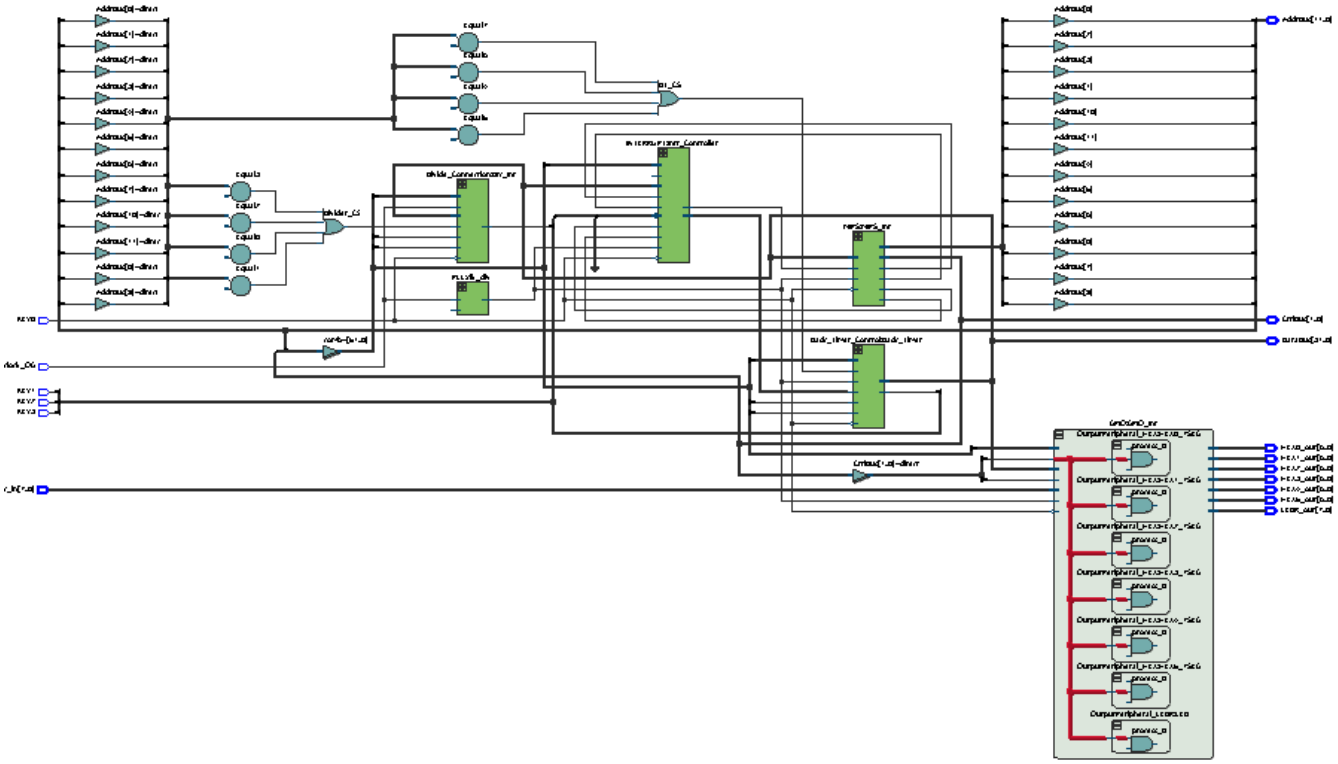
ה GPIO-מיושם כדיקודר פשוט עם רגיסטרי חוצץ, כפי שמוזכר בעמוד 4 של מסמך הפרויקט.

חשוב לציין:

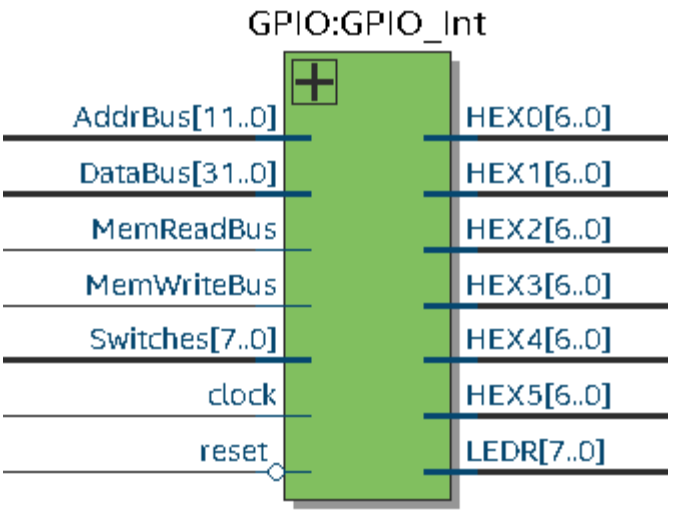
KEY0משמש כ ,System RESET-מחזיר את ה PC-להוראה הראשונה בתוכנית (עמוד 3).
הפסיקות מהכפתורים (KEY3-KEY1) מטופלות דרך דגל ,KEYiIFG שמאופס ידנית בתוכנה (עמוד 10).

דיאגרמת:RTL

דיאגרמת rtl:



תיאור גרפי של המודל:


GPIO:GPIO_Int
AddrBus[11..0]
DataBus[31..0]
MemReadBus
MemWriteBus
Switches[7..0]
clock
reset
HEX0[6..0]
HEX1[6..0]
HEX2[6..0]
HEX3[6..0]
HEX4[6..0]
HEX5[6..0]
LEDR[7..0]

DIVIDER(מאיץ חלוקה):(

הרכיב הפריפריאלי DIVIDER הוא מאיץ חומרה שתוכנן לביצוע פעולות חלוקה בין שני מספרים בינאריים באופן מהיר ויעיל. מימוש זה תואם את דרישות הפרויקט כפי שמפורט בעמוד 9-8 של מסמך ההגדרות.

מאפייני ה:DIVIDER-

פונקציונליות :

מבצע חלוקה בינארית ללא סימן.(Unsigned Binary Division)

פועל כמאיץ חומרה, מחליף חישוב תוכנתי ב.CPU-

ביצועים :

משפר את זמני החישוב של פעולות חלוקה.

חוסך זמן עיבוד יקר של ה.CPU-

משפר את הביצועים הכוללים של המערכת.

מבנה :

מכיל רגיסטרים לאחסון המחולק ,(Dividend) המחלק ,(Divisor) והתוצאה.

כולל מנגנון בקרה לניהול תהליך החלוקה.

זמן ביצוע :

התוצאות מוכנות לאחר N מחזורי ,DIVCLK כאשר N=32 במקרה שלנו.

פסיקות :

מסוגל לייצר פסיקה (DIVIFG) בסיום פעולת החלוקה.

הדגל DIVIFG מאופס אוטומטית כאשר הפסיקה מטופלת.

ממשק :

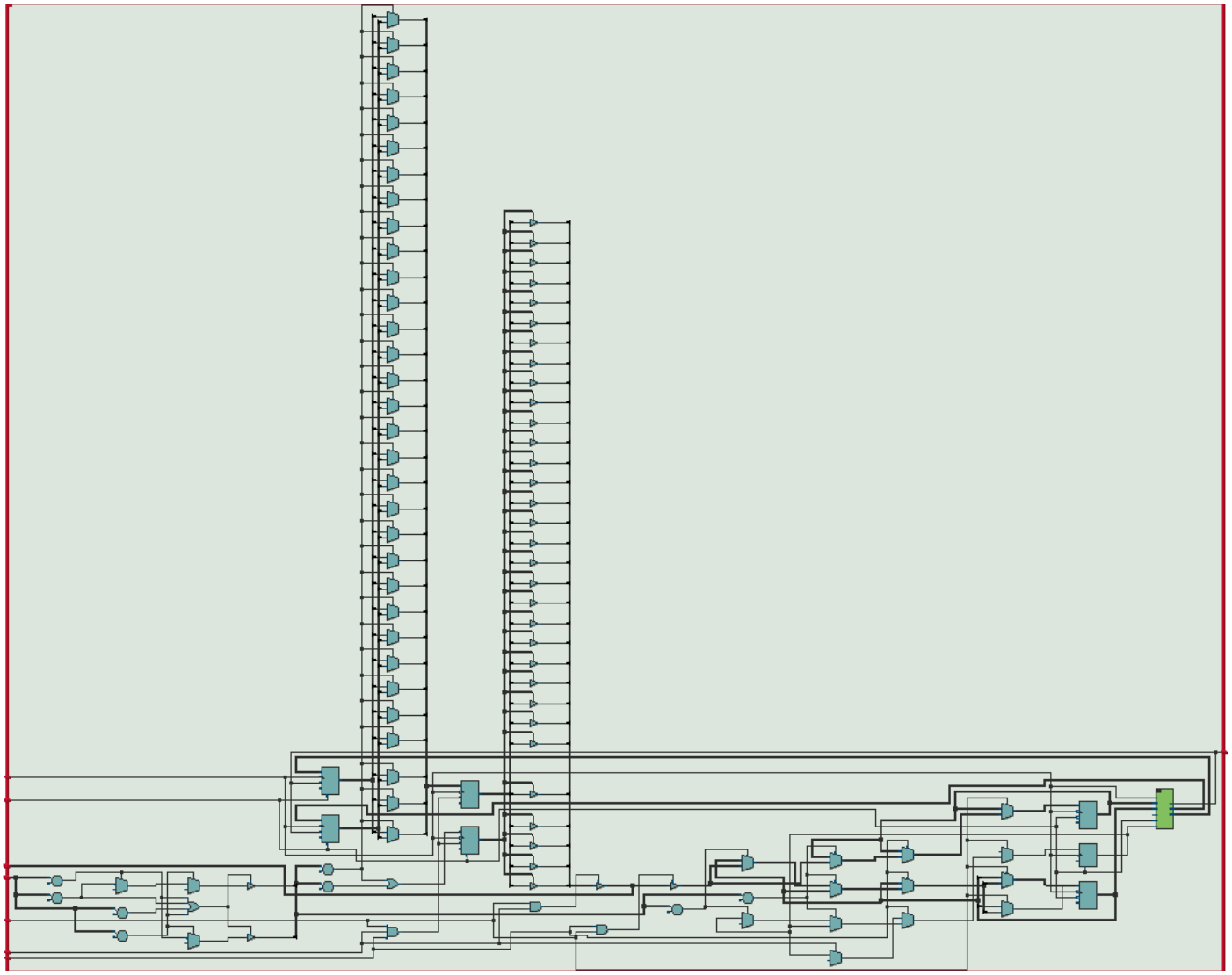
מתקשר עם ה CPU-דרך ממשק ייעודי, כנראה באמצעות.Memory Mapped I/O

חשוב לציין:

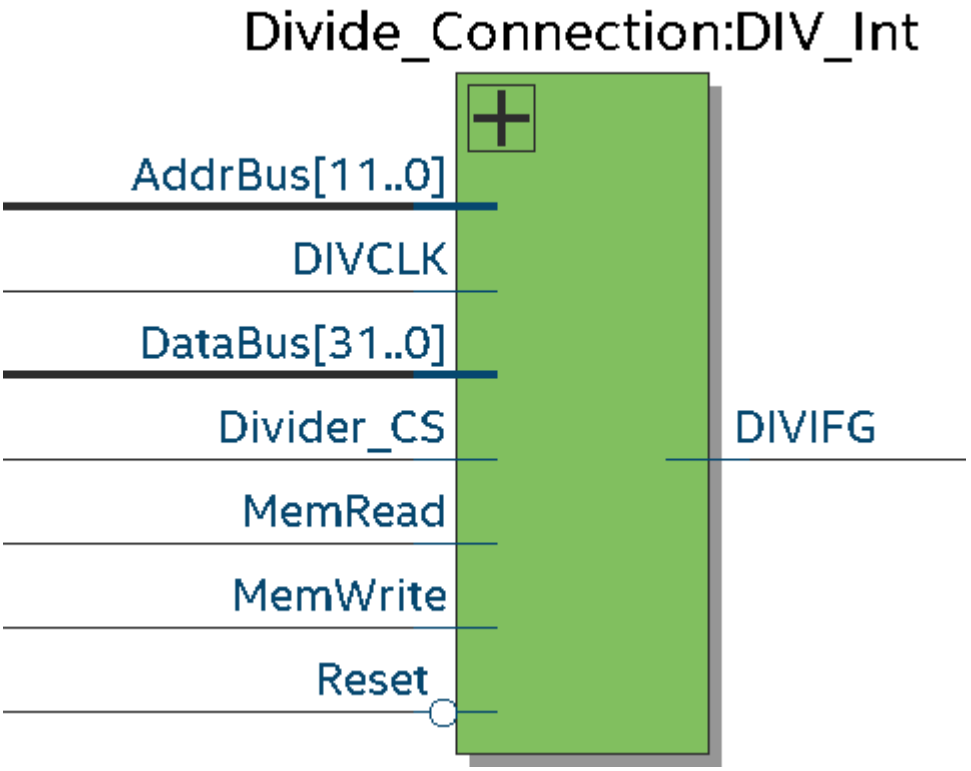
השימוש במאיץ חומרה כזה מדגים את היתרון של ארכיטקטורת ,MCU המשלבת רכיבים ייעודיים לשיפור ביצועים.

יש לוודא סנכרון נכון בין ה DIVIDER-וה CPU-כדי להבטיח קריאה נכונה של התוצאות.

דיאגרמת:RTL



תיאור גרפי של המודל:


Divide_Connection:DIV_Int
AddrBus[11..0]
DIVCLK
DataBus[31..0]
Divider_CS
MemRead
MemWrite
Reset
DIVIFG

נתיב קריטי של כל המערכת:

	Fmax	Restricted Fmax	Clock Name	Note
1	68.64 MHz	68.64 MHz	altera_reserved_tck	

SIM MODEL : ניתוח תוצאות ב