דו''ח מסכם לפרויקט גמר

קורס "מבנה מחשבים ספרתיים"

361-1-4191

Barcode39 Reader

מגישים: בר פורטנוי 308244185, ולדיסלב קוז'וחוב 332305747

תאריך הגשה: 12/09/2020

**הגדרת ומטרת הפרויקט**

מטרת הפרויקט היא תכנון כרטיס קורא ברקוד המבוסס על מעבד 8086.

הפרויקט כולל פיתוח ותכנון של מודול חומרה המאפשר קריאה של ברקוד 39 בטכניקה המבוססת על DMA. במסגרת הפרויקט בנוסף לפיתוח ותכנון מפורט של מודול החומרה מבוצעת גם סימולציה מלאה של החומרה בסביבת ModelSim באמצעות שפת VHDL.

**תיאור הפרויקט**

המערכת היא כרטיס המחובר למעבד 8086 וכוללת שני מודולים של Timer8254, מודול DMA8237, מודול Interrupt Controller 8259 ורכיב זיכרון.

ברקוד 39 הוא שיטה לייצוג 43 תווים שונים (מספרים, אותיות גדולות וסימנים) בעזרת רצף פסים שחורים ולבנים עבים ודקים. הברקוד מתקבל על ידי הקורא בצורה טורית כאשר '1' מייצג קו לבן ו-'0' מייצג קו שחור. משך פס דק הוא 200ns ומשך פס עבה הוא 400ns. במצב idle מוצא הרכיב מייצג רקע לבן, כלומר '1' קבוע. ייצוג הקלט בסביבת ModelSim מתואר על ידי קובץ טקסט בשם Scanner.txt המכיל תווים '1'/'0' בלבד כאשר ציר הזמן בקובץ הוא 200ns. המערכת מוזנת על ידי שעון בעל תדר של 20MHz, ובעלת חיבור Reset.

העברת המידע בין ההתקן הסורק לזיכרון מתבצעת בעזרת מודול ה-DMA ללא התערבות ה-CPU בתהליך. המערכת בעלת Bus מרובב בגודל 20bit כאשר 16 הביטים התחתונים משמשים כ-Data, וכל 20 הביטים משמשים כ-Address, בפועל מרחב הזיכרון הוא 16Kbit ולכן נעשה שימוש רק ב-13 ביט התחתונים.

הזיכרון הפנימי של הרכיב הוא RAM בגודל 16Kbit בתצורת Single Port ובעל חיבור Interleaving.

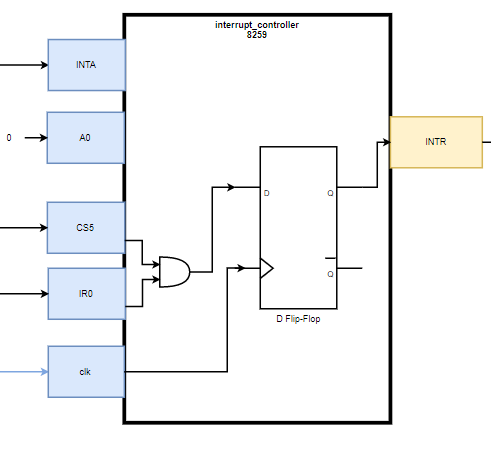
הטיימרים מבצעים שמירה של ערך המונה הפנימי בעלייה ובירידה של אות הכניסה ושומרים אותם ברגיסטרים פנימיים, תוצאת חיסור רגיסטרים אלו מייצגת את רוחב הפולס הנכנס. לאחר החישוב מתבצעת בקשת העברת מידע מ-IO (טיימר) ל-MEM בעזרת ה-DMA על ידי בקשת DREQ. ה-DMA מחזיר DACK לרכיב המבקש, שולח לזיכרון את הכתובת לכתיבה ומאפשר לטיימר לכתוב ל-Bus ולזיכרון לקרוא מה-Bus. בסיום העברת כל התווים ה-DMA שולח בקשת פסיקה לבקר הפסיקות וממנו נשלחת פסיקה ל-CPU המסמנת שנגמר תהליך הסריקה וכל המידע זמין לקריאה בזיכרון.

**שרטוט רכיבים**

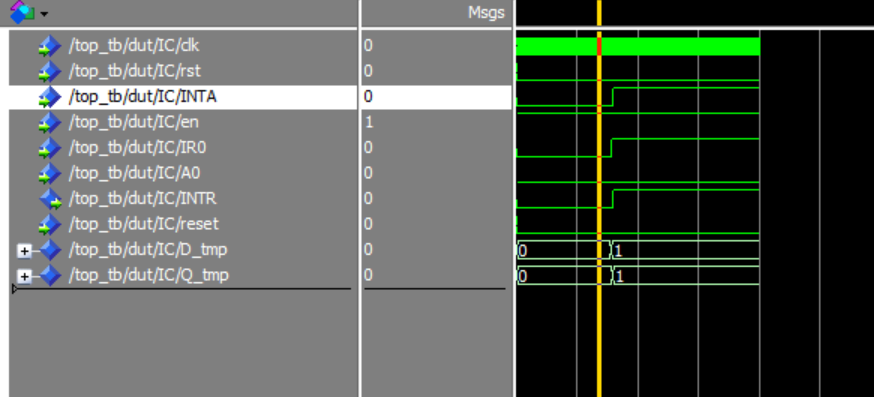
Interrupt\_controller.vhd

כניסות – clk, rst, INTA, reset,en, IR0

יציאות – INTR



בנוסף נציג את הWAVE של המודול:

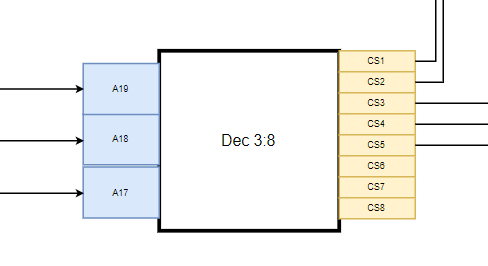


מטרת המודול בתכנון שלנו זה להודיע ל-CPU שה-DMA סיים את העברת הנתונים ומוכן להחזיר את הבעלות על ה-BUS ל-CPU. הוא מודיע בעזרת INTR ומקבל INTA מהCPU שהוא קיבל את בקשת הפסיקה שלו. מתוך ה-WAVE ניתן לראות שכאשר IR0 (פסיקה מה-DMA שהוא סיים) הגיע IC שולח בקשת INTR ל-CPU וה-CPU מחזיר INTA.

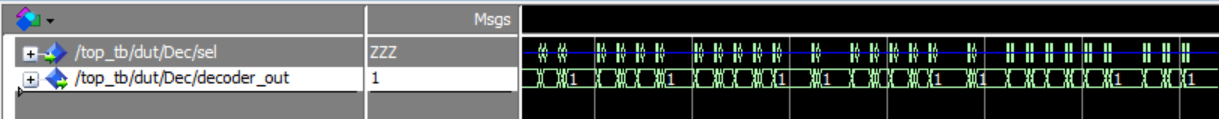
decoder.vhd

כניסות – sel

יציאות – decoder\_out



נציג WAVE:

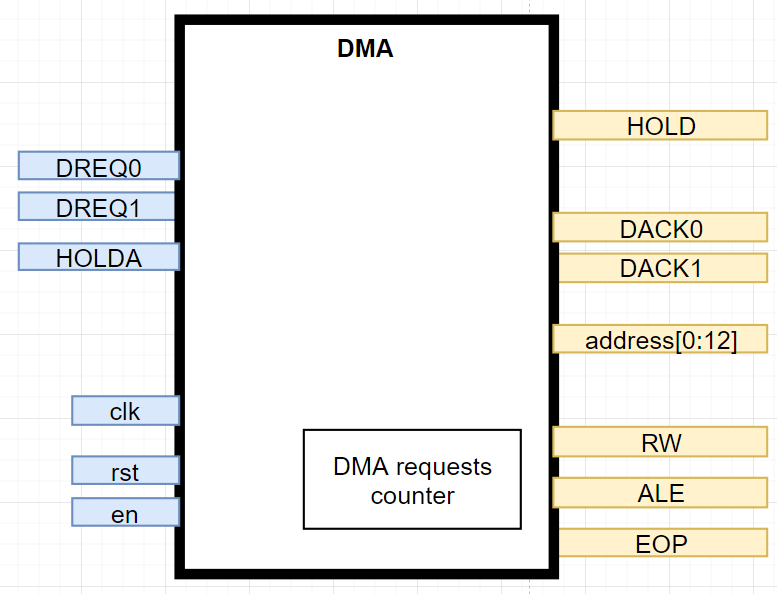


מטרת המודול זה לפנעח 3 ביט העליון של הBUS ולהדליק את הCS המתאים. לפי הWAVE ניתן לראות שמודול אכן עובד, מוציא את ערך של הCS שנדלק במצב שהוא לא אמור להדליק CS הוא מוציא Z.המימוש שלו בחומרה זה DECODER. בתכנון שלנו קבענו שכתובות של הרכיבים יהיו ב-3 ביטים העליונים בשביל לבצע מימוש יותר פשוט ויעיל ולא להשתמש ב BUS המשותף של כתובות ודאטא.

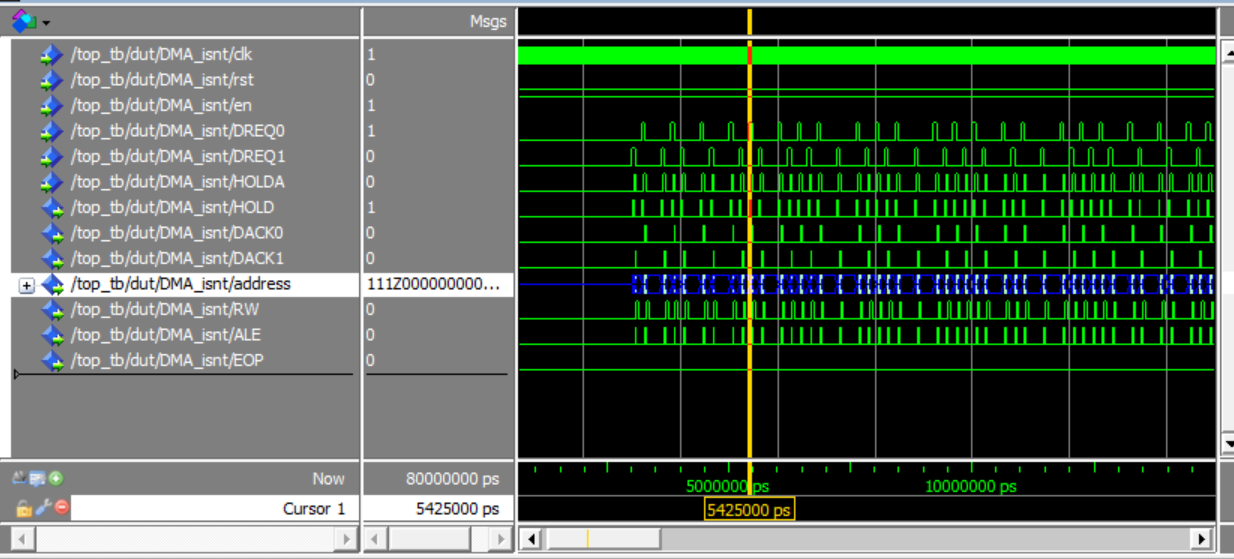
DMA.vhd

כניסות – clk, rst, en, DREQ0, DREQ1, HOLDA

יציאות – DACK0, DACK1, address, HOLD, RW, ALE, EOP



נציג WAVE של המודול:

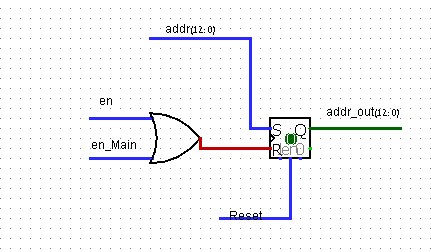


מודול זה מממש את יכולות ה-DMA כפי שנלמד בכיתה אך מותאם למטרה שלנו. המודול מקבל בקשה מאחד ה-TIMERS להעברת המידע ממנו אל ה-MEM. בנוסף DMA מקבל בעלות על הBUS בדיוק כפי שנלמד בהרצאות ולאחר מכן מצבע העברת מידע במחזור אחד. בחצי המחזור הראשון הוא שם כתובת של הזכרון (לאן לכתוב) ומדליק את ה-CS של ה-TIMER המתאים ובחצי המחזור השני ה-TIMER שם דאטא על ה-BUS וה-DMA מדליק ביט WRITE\_EN על מנת לסמן CS עבור ה-MEM. בעצם מימשנו את על התהליך של העברת הנתונים ודאגנו שהוא יקח בדיוק אותה כמות המחזורים כפי שנלמד בכיתה. בסיום התהליך ה-DMA שולח פסיקה של EOP ל-IC וה-CPU לוקח בעלות על ה-BUS. מתוך ה-WAVE ניתן לראות שה-DMA עובד בצורה נכונה וכל הסיגנלים מסונכרנים ביניהם, בנוסף כאשר DMA לא משתמש בBUS הוא מוציא Z.

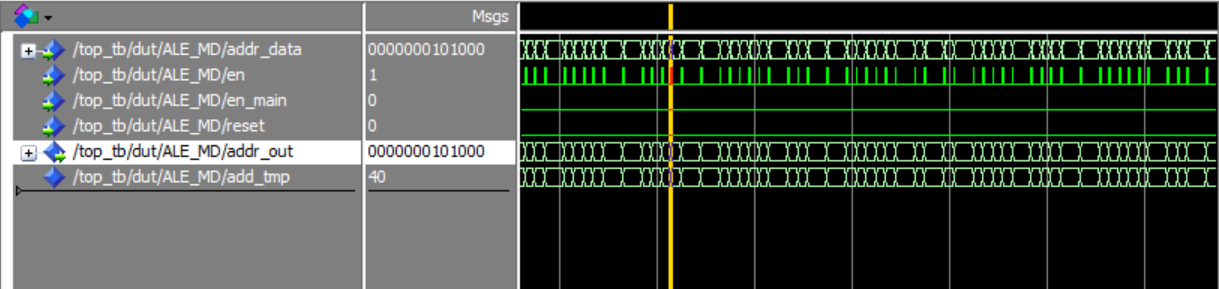
Ale.vhd

כניסות – addr\_data, en, en\_main, reset

יציאות – addr\_out



נציג בנוסף את WAVE:

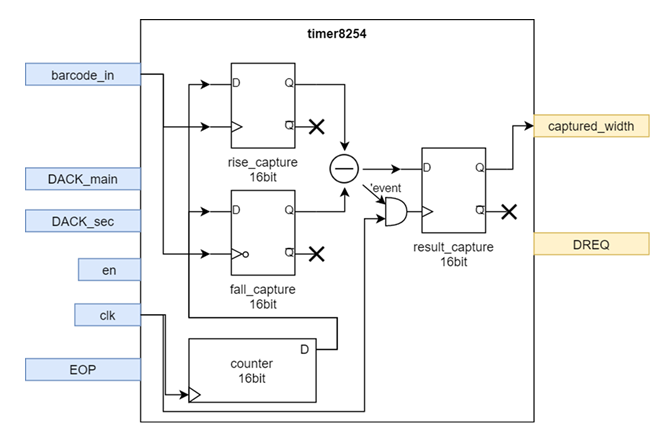


המודול נועל את ערך הכניסה בעת ירידת enable ל-'0' ושמירת הערך. על מנת להבדיל האם עכשיו נמצאת כתובת או דאטא על הBUS.לפי ה-WAVE ניתן לראות שמודול עובד כמו שציפינו והוא נועל את הערך הרצוי .מימוש בחומרה כ-LATCH.

Timer8254.vhd

כניסות – barcode\_in, clk, rst, en, EOP, DACK\_main, DACK\_sec.

יציאות - captured\_width, DREQ.



Counter – מונה את עליות ה-clk.

DFF rise\_capture – שומר את ערך ה-counter בעת עליית האות barcode\_in.

DFF fall\_capture – שומר את ערך ה-counter בעת ירידת האות barcode\_in.

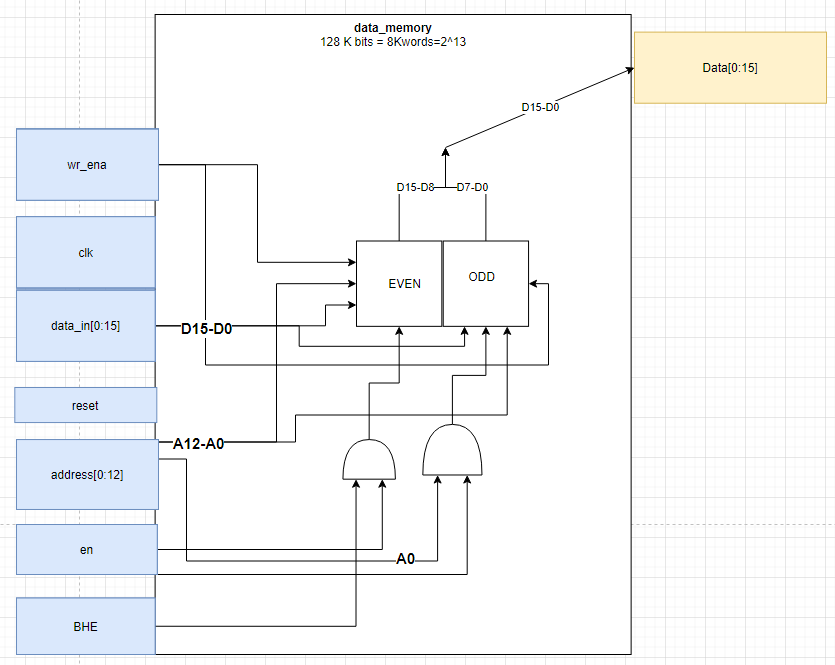
DFF result\_capture – שומר את ערך החיסור בין ה-FF.

כאשר ערך החיסור מוכן להעברה לזיכרון נשלח אות DREQ לבקשת העברת מידע בין הטיימר לזיכרון. כאשר מתקבל DACK\_main ו-en מאופשר התוצאה השמורה ב-result\_capture יוצאת אל ה-BUS.

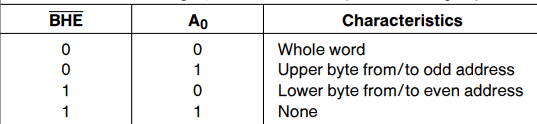
Memory\_interleaving.vhd

כניסות – clk, en, reset, en, wr\_ena, BHE, address, data\_in.

יציאות – data\_out.



זהו זיכרון RAM מסוג Single-Port בגודל 16Kbit עם גודל מילה 16bit בעל חיבור interleaving. לכתיבה יש להדליק את האות wr\_ena, ולקריאה יש לאפשר en. הקריאה והכתיבה מתבצעות על פי הטבלה הבאה:



readFile.vhd

יציאות – input\_top.

משמש לקריאת קובץ טקסט והכנסת כל תו לסימולציה בכל זמן קבוע (200ns).

writeFile.vhd

כניסות – input\_top, clk, en.

משמש לכתיבת ערך אות מסוים המחובר אליו לקובץ טקסט בכל זמן קבוע המוגדר על ידי ה-clk המחובר.

**קבצי הפרוייקט**

|  |  |
| --- | --- |
| **שם קובץ** | **תיאור מימוש** |
| Ale.vhd  **Latch** | כניסות – addr\_data, en, en\_main, reset  יציאות – addr\_out  נועל את ערך הכניסה בעת ירידת enable ל-'0' ושמירת הערך. |
| counter.vhd  **מונה** | כניסות – clk, rst, en  יציאות – count\_out  מונה את עליות השעון בכניסה ושומר אותן ברגיסטר פנימי. |
| decoder.vhd  **מפענח** | כניסות – sel  יציאות – decoder\_out  מפענח מספר בינארי ליציאה המתאימה ליצירת CS מתאים. |
| DFF.vhd  **DFF** | כניסות – D, clk, rst  יציאות – Q  רכיב זיכרון, הערך ב-D נשמר ברכיב בעליית השעון, הערך השמור יוצא מ-Q. |
| DMA.vhd  **DMA** | כניסות – clk, rst, en, DREQ0, DREQ1, HOLD, HOLDA  יציאות – DACK0, DACK1, address, RW, ALE, EOP  אחראי על קבלת בעלות על ה-BUS מה-CPU, שליטה באותות הבקרה על מנת לבצע העברות IO-MEM ושולח פסיקת EOP בסיום השליחות. |
| Interrupt\_controller.vhd  בקר פסיקות | כניסות – clk, rst, INTA, CS5, IR0  יציאות – INTR  כאשר מתקבלת פסיקה ב-IR0 מרכיב IO נשלח INTR ל-CPU ומקבל חזרה INTA. |
| Memory\_interleaving.vhd  זיכרון | כניסות – clk, en, reset, en, wr\_ena, BHE, address, data\_in  יציאות – data\_out  זיכרון RAM מסוג Single-Port בגודל 16Kbit עם גודל מילה 16bit. |
| readFile.vhd  **Read from file** | יציאות – input\_top  כתיבת ערכים מקובץ למודול חיצוני בקצב קבוע של 200ns. |
| Timer8254.vhd  **Timer8254** | כניסות – barcode\_in, clk, rst, en, EOP, DACK\_main, DACK\_sec  יציאות - captured\_width, DREQ  הטיימר שומר את ערך ה-counter הפנימי בעלייה ובירידת אות הכניסה barcode\_in, בעזרת תמיכה בחצץ כפול מתאפשר חיסור בין שני הרגיסטרים לקבלת מרווח בין שני event סמוכים ביחידות של 50ns. |
| TOP.vhd  **TOP** | כניסות - STRSCAN, clk, reset, barcode\_in, HOLDA, INT, INTA, ALE, BHE, BUS\_AD  יציאות - BUS\_AD, barcode\_out, HOLD, EOP  מודול זה מכיל את כל שאר המודולים ואת החיבורים ביניהם. |
| writeFile.vhd  **Write to file** | כניסות – input\_top, clk, en  מודל המקבל ערך וכותב אותו לקובץ חיצוני בעליית שעון. |

**הבדלים בין הסכמה למימוש בפועל**

* לטובת פעולה תקינה של תזמון המודולים השונים הוספנו כניסות נוספות כמו EOP ו-DACK\_sec בטיימרים והכנסנו אותם לרשימת הרגישות במודולים שהשתנו.
* הרחבת ה-BUS ל-20bit כמו ב-8086 ושימוש ב-3 ביט העליונים כקוד ל-CS עבור הרכיבים השונים שפוענח על ידי ה-Decoder.

**מסקנות והצעות לשיפור**

* הרכיב עובד כמצופה, קורא בצורה מדויקת את הקלט, שומר את הפלט בזיכרון הפנימי ומוציא אותו לקובץ טקסט כנדרש.
* תכנון ראשוני יסודי מקל ומייעל את המימוש.