Ultra 37K Evaluation Board

חוברת ניסויים

Designed and Written by: Dr. Eli Flaxer ♣

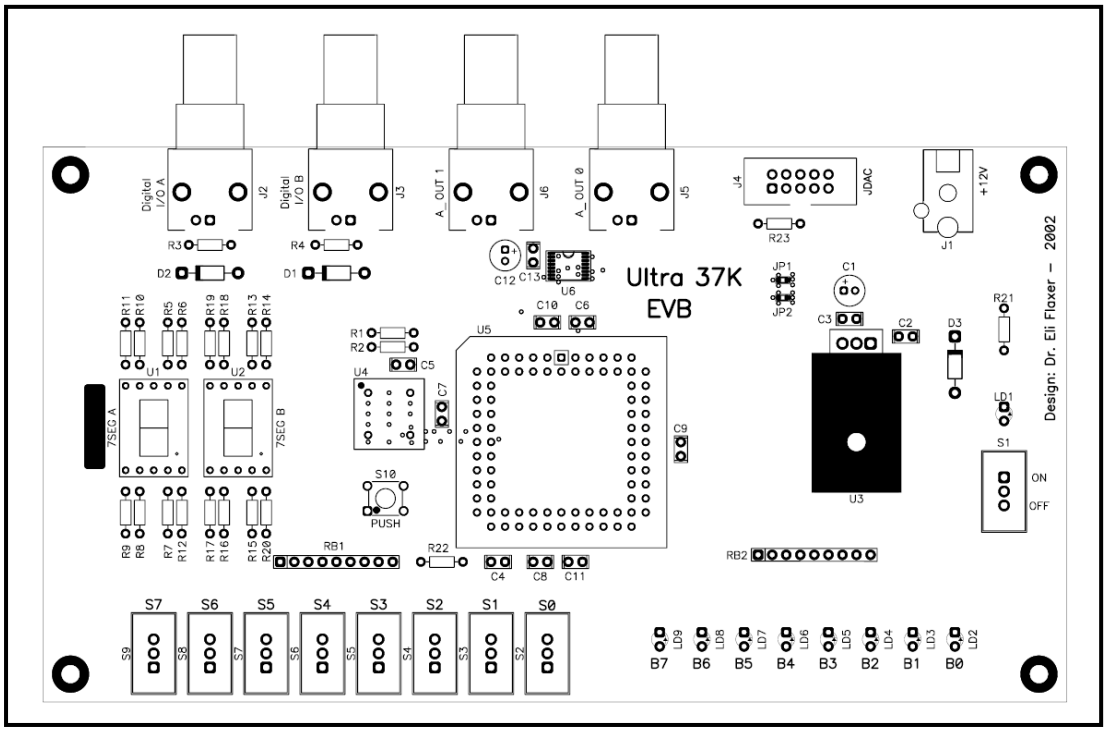
מהדורה ראשונה

2003

הקדמה

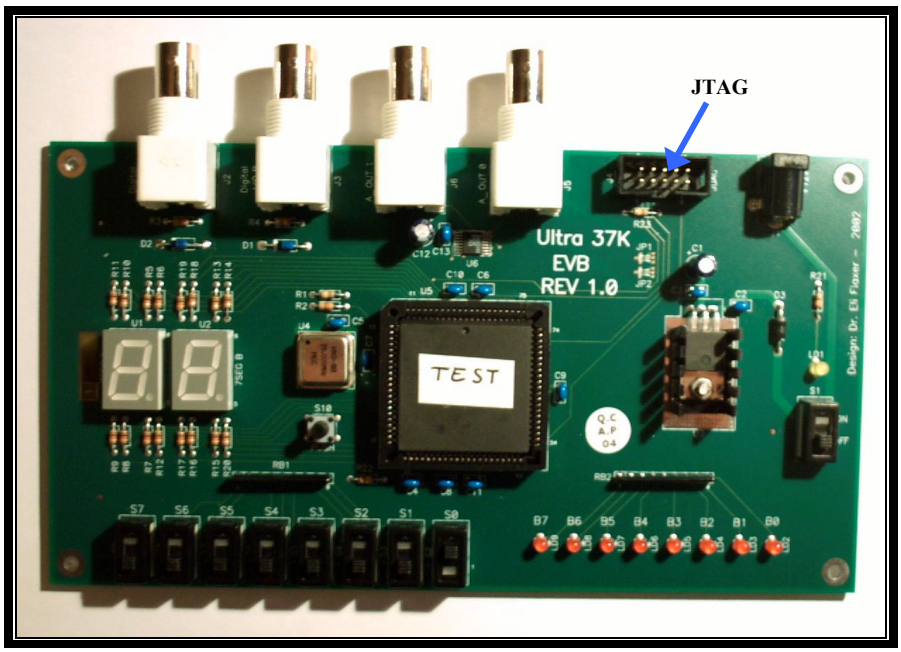
ערכת Ultra37K EVB משמשת לתרגול ניסיונות בזמן אמת בקורס שפות חומרה. הרכיב המרכזי בערכה - CY37128P84, הוא CPLD בעל 128 macro cells מסדרת ULTRA של חברת CYPRESS. בכדי לאפשר תרגול רחב ויעיל הוספו התקני סביבה ורכיבים תואמים, בהם (ראה תרשים למטה):

- שמונה דיודות פולטות אור (LED) B_0 - B_7.
- שמונה מפסקי הזזה (SWITCH) S_0 - S_7.
- שני תצוגות מקטעים (7SEG) $7SEG_A$ - $7SEG_B$.
- לחצן (PUSH) S_{10}.
- שתי כניסות/יציאות דיגיטליות (Bidirectional I/O) I/O_A - I/O_B.
- שתי יציאות אנלוגיות (מבוקרות ע"י ממיר DAC מסוג MAX5102) $AOUT_0$ - $AOUT_1$.
- שעון (10 MHz Clock) U_4.
- מחבר לתכנות (JTAG) J_4.
- מפסק הפעלה S_1.
- שקע לכניסת מתח (9V - 12V) J_1.



איור 1: תאור המעגל המודפס של ערכת Ultra37K EVB

בתמונה הבאה ניתן לראות צילום של הערכה על כל מרכיביה כפי שתוארו למעלה.

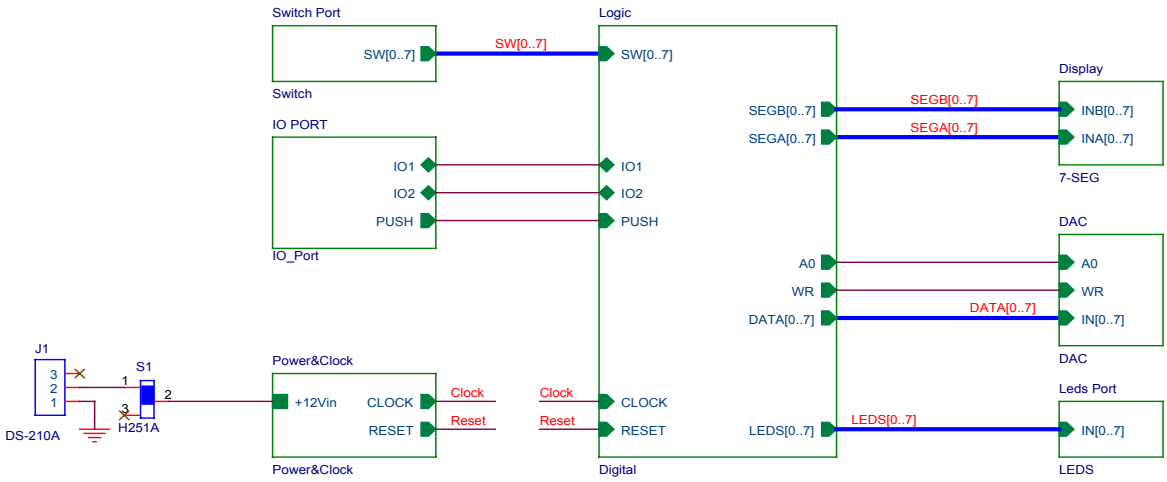


את הרכיב אנו צורבים על הלוח בטכנולוגיית In System Reprogramming (ISR). לשם כך קיים על הלוח מחבר JTAG (J_4) המתאים לכבל צריבה מיוחד (C3ISR Programming Cable) מתוצרת CYPRESS. הכבל, המופיע בתמונה למטה, מתחבר בין היציאה המקבילית של מחשב PC לבין מחבר ה JTAG שעל הכרטיס. תוכנה מתאימה, שבהמשך נסביר את פעולתה, צורבת את הרכיב.

ארכיטקטורת הכרטיס

בכרטיס הפיתוח פיני רכיב ה CPLD מחווטים בצורה קשיחה להתקני הסביבה. ארכיטקטורה זו מחייבת אותנו לאילוץ הפינים בתהליך הסינתזה. **אילוץ זה חשוב ביותר ואי התחשבות בו עלול לגרום נזק בלתי הפיך לערכת הפיתוח**. מדוע ?. בסינתזה ללא אילוץ פינים הסינתיסייזר בוחר את אילוץ הפינים בצורה עצמאית לפי שיקולים של יעילות פנימית. אילוץ כזה יכול (למעשה כמעט וודאי) לעמוד בסתירה לחיווט בפועל על הכרטיס ובכך לחבר למשל יציאה ליציאה דבר שיגרום לשריפת הרכיבים שיציאתם חוברה יחדיו.

הארכיטקטורה הפנימית של ערכת הפיתוח, המתוארת בדיאגרמה למטה באיור 1, מציינת את חיבורי ה- CPLD להתקני הסביבה שלו.



איור 2 : ארכיטקטורת כרטיס הפיתוח

בתרשים אנו רואים את שמות הסיגנלים המחברים בין הבלוק LOGIC, המכיל את הלוגיקה המתכנתת, לבין התקני הסביבה שצויינו בפסקה הראשונה. יתר על כן, בתרשים רואים גם את תכונת הסיגנלים (רוחב הבס, וכיוון הזרימה). אנו חייבים להתאים כל תכנון לוגי בשפת חומרה לארכיטקטורה זו. כלומר, תכנון בשפת VHDL מחייב כתיבת ENTITY המתאים בדיוק לארכיטקטורה המצויינת למעלה. כדי למנוע בעיות אפשריות, רצוי לכתוב ENTITY המכיל את כל הסיגנלים המופיעים בתרשים, גם אם לא נעשה שימוש בהם. בעמוד הבא, איור 3 מתאר את ה ENYITY המתאים כפי שנכתב בשפת VHDL. רצוי להשתמש בדיוק בדוגמא זו בהמשך התרגול על ערכת הפיתוח.

```
entity Main is
  port(
        IO1      : inout std_logic;
        IO2      : inout std_logic;
        A0       : out   std_logic;
        WR       : out   std_logic;
        PUSH     : in    std_logic;
        LEDS     : out   std_logic_vector(7 downto 0);
        SEGA     : out   std_logic_vector(7 downto 0);
        SEGB     : out   std_logic_vector(7 downto 0);
        DATA     : out   std_logic_vector(7 downto 0);
        SW       : in    std_logic_vector(7 downto 0);
        RESET    : in    std_logic;
        CLK      : in    std_logic
    );
```

איור 3: ENTITY של הכרטיס בשפת VHDL

להשלמת תהליך העבודה אנו צריכים לאלץ את מספרי הפינים על הרכיב לסיגנלים המופיעים ב ENTITY. שפת VHDL תומכת באילוץ פינים בצורה שונה בין סביבות הפיתוח. בסיבת WARP ניתן להוסיף קובץ בעל שם זהה לשם הפרוייקט ובסיומת ctl (לדוגמא MyProj.ctl) לספרייה בה אנו עובדים. קובץ זה צריך להכיל את ההוראות לאילוץ הפינים בהתאם לפורמט הבא:

```
Attribute PIN_NUMBERS of SignalName is "PinNum";
```

כאשר SignalName מייצג שם של סיגנל כל שהוא ב ENTITY ו PinName את מספר הפין של הרכיב המתאים לו. בצורה זו צריך להגדיר מספרי פינים לכל הסיגנלים שב ENTITY. באיור 4 מוצג הקובץ השלם של אילוץ הפינים, אותו ניתן להוריד מאתר הקורס (חשוב לציין שבאתר הקורס שם הקובץ הוא **Ultra37K.ctl** והמשתמש צריך להמירו בשם הפרוייקט שלו, לדוגמא: **MyProj.ctl**). יש לשים לב שסיגנלים שהם וקטורים אינם מוגדרים כיחידה אחת, אלא, כל ביט בוקטור מוגדר בפני עצמו ומשוייך לפין המתאים לו.

הערה חשובה למתכנן:

בפיתוח מעגל חדש, שלב המעגל המודפס נעשה לאחר סיום התכנון הלוגי. במצב כזה מאפשרים לסינטיסייזר לבחור את הקצאת הפינים בעצמו בצורה היעילה ביותר. באילוץ מראש של הפינים אנו עלולים להגביל מאד את הרכיב בניצול המשאבים הפנימיים שלו.

```
Attribute PIN_NUMBERS of SEGA(0) is "3" ;
Attribute PIN_NUMBERS of SEGA(1) is "4" ;
Attribute PIN_NUMBERS of SEGA(2) is "5" ;
Attribute PIN_NUMBERS of SEGA(3) is "6" ;
Attribute PIN_NUMBERS of SEGA(4) is "7" ;
Attribute PIN_NUMBERS of SEGA(5) is "8" ;
Attribute PIN_NUMBERS of SEGA(6) is "9" ;
Attribute PIN_NUMBERS of SEGA(7) is "10" ;

Attribute PIN_NUMBERS of SEGB(0) is "82" ;
Attribute PIN_NUMBERS of SEGB(1) is "81" ;
Attribute PIN_NUMBERS of SEGB(2) is "80" ;
Attribute PIN_NUMBERS of SEGB(3) is "79" ;
Attribute PIN_NUMBERS of SEGB(4) is "78" ;
Attribute PIN_NUMBERS of SEGB(5) is "77" ;
Attribute PIN_NUMBERS of SEGB(6) is "76" ;
Attribute PIN_NUMBERS of SEGB(7) is "75" ;

Attribute PIN_NUMBERS of SW(0) is "31" ;
Attribute PIN_NUMBERS of SW(1) is "30" ;
Attribute PIN_NUMBERS of SW(2) is "29" ;
Attribute PIN_NUMBERS of SW(3) is "28" ;
Attribute PIN_NUMBERS of SW(4) is "27" ;
Attribute PIN_NUMBERS of SW(5) is "26" ;
Attribute PIN_NUMBERS of SW(6) is "25" ;
Attribute PIN_NUMBERS of SW(7) is "24" ;

Attribute PIN_NUMBERS of LEDS(0) is "61" ;
Attribute PIN_NUMBERS of LEDS(1) is "60" ;
Attribute PIN_NUMBERS of LEDS(2) is "59" ;
Attribute PIN_NUMBERS of LEDS(3) is "58" ;
Attribute PIN_NUMBERS of LEDS(4) is "57" ;
Attribute PIN_NUMBERS of LEDS(5) is "56" ;
Attribute PIN_NUMBERS of LEDS(6) is "55" ;
Attribute PIN_NUMBERS of LEDS(7) is "54" ;

Attribute PIN_NUMBERS of DATA(0) is "50" ;
Attribute PIN_NUMBERS of DATA(1) is "49" ;
Attribute PIN_NUMBERS of DATA(2) is "48" ;
Attribute PIN_NUMBERS of DATA(3) is "47" ;
Attribute PIN_NUMBERS of DATA(4) is "46" ;
Attribute PIN_NUMBERS of DATA(5) is "45" ;
Attribute PIN_NUMBERS of DATA(6) is "40" ;
Attribute PIN_NUMBERS of DATA(7) is "39" ;

Attribute PIN_NUMBERS of A0 is "67" ;
Attribute PIN_NUMBERS of WR is "66" ;

Attribute PIN_NUMBERS of PUSH is "17" ;

Attribute PIN_NUMBERS of IO1 is "18" ;
Attribute PIN_NUMBERS of IO2 is "19" ; -- 19 or 20

Attribute PIN_NUMBERS of reset is "41" ;
Attribute PIN_NUMBERS of clk is "23" ;
```

איור 4: קובץ אילוץ הפינים של ערכת הפיתוח (Ultra37K.ctl)

צריבה על הלוח - ISR

לפני כל תכנון או שימוש בשיטת ISR לצריבת רכיבים, בין אם לתרגול ובין אם לתכנון עצמי של פרוייקט חדש, כדאי לקרוא קודם את המסמך המופיע באתר של CYPRESS (וגם באתר הקורס):

Design Considerations for In-System Reprogrammable (ISR) Programming of Cypress CPLDs.

המאמר מתאר כיצד לתכנן מערכת ISR, את צורת החיבור של JTAG לרכיב ועוד פרטים רבים הקשורים בנושא ISR. מכל מקום, סדר תכנון צריך להיות כדלקמן:

1. כתיבת המודל בשפת VHDL - Text Editor.
2. סימולציה בקוד מקור (אופציה) - Active HDL.
3. קומפילציה לבדיקת שגיאות תחביר - WARP.
4. סינתזה לרכיב נבחר וקבלת קובץ VHDL לסימולציית RTL - WARP.
5. השמה לרכיב, קבלת קובץ JED - WARP.
6. סימולציה בקוד RTL מסונתז - Active HDL.

למעשה, בעבודה תחת WARP השלבים 3-5 נעשים בצורה שקופה למשתמש בצורה סדרתית, כך שקובץ JED מתקבל באופן אוטומטי במקרה של סינתזה מוצלחת. אם שלב 6 עבר בהצלחה, כלומר, סימולציה בקוד RTL מסונתז עונה על הדרישות עוברים לשלב הצריבה. אנו נעבוד בטכנולוגיית ISR המאפשרת צריבת הרכיב על הלוח ללא הוצאתו מן המעגל, דבר החשוב ביותר בשלב הפיתוח של אב טיפוס. יתר על כן, עלותו של כבל הצריבה בטכנולוגית ISR נמוכה ממאה דולר, דבר הזמין לכל מתכנן.

הצריבה יכולה להעשות גם בצורב תעשייתי (לדוגמא: BP Microsystem, Data IO). היתרון בעבודה עם צורב תעשייתי הוא קצב הצריבה המהיר בפס יצור, לאחר שמעגל האב טיפוס נבדק ואושר. עלותו של צורב כזה יכולה להגיע לעשרות אלפי דולרים והוא אינו זמין לכל אחד.

סדר הצריבה בטכנולוגיית ISR:

1. המעגל חייב להיות כבוי.
2. חיבור כבל הצריבה למחשב.
3. חיבור מחבר JTAG למעגל.
4. טעינת תוכנת הצריבה - ISR Programming Software.
5. הדלקת המעגל.
6. הכנסת הנתונים לתוכנה וצריבה בפועל.

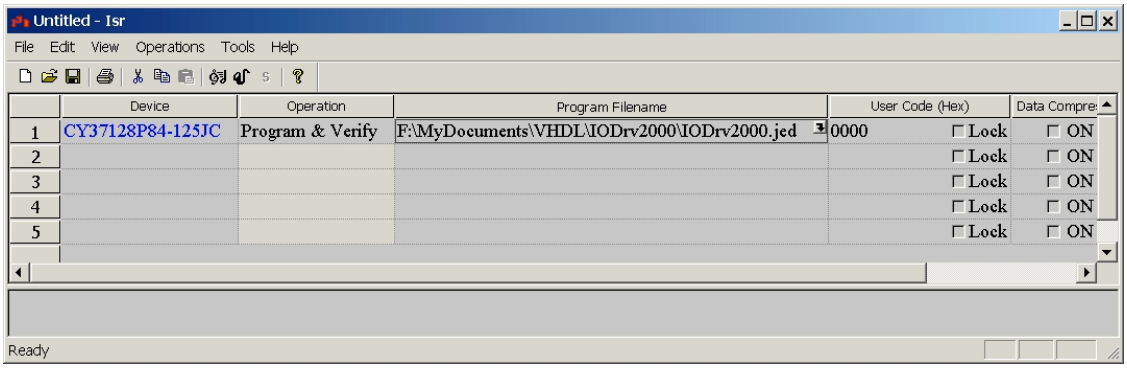
חשוב מאד לשמור על סדר הפעולות שהוצג למעלה. אי שמירה על הסדר עלולה לגרום לנזק בלתי הפיך לרכיב, במיוחד בצריבה הראשונה שלו. הרכיב מגיע מהחברה בקונפיגורצית מוצאים לא ידועה, דבר העלול לעמוד בסתירה לארכיטקטורה על הלוח. שמירה על הסדר הנ"ל תבטיח את שלמות הרכיב באופן ודאי.

בשלב זה נעבור לתאור תוכנת הצריבה בטכנולוגיית ISR של חברת CYPRESS המסופקת חינם באתר האינטרנט של החברה.

תוכנת הצריבה - ISR Programming Software

ראשית כל צריך להתקין את תוכנת הצריבה במחשב. לאחר ההתקנה ניתן לגשת לתוכנה בשני אופנים: 1) הדרך הרגילה, דרך תפריט התחל וכן הלאה; 2) מתוך תוכנת WARP, דרך תפריט Tools->Cypress.

החלון המרכזי של תוכנת הצריבה ניראה באיור 5.



איור 5: תוכנת הצריבה ISR Programming Software של CYPRESS

על המשתמש למלא את העמודות בערכים המתאימים כפי שרואים בדוגמא. באופן כללי, ניתן לבצע כמה מהלכים יחדיו ולצרוב כמה רכיבים המשורשרים על המעגל המודפס. במעבדה אנו נצרוב רכיב בודד, ולכן נמלא שורה בודדת לפי הפורמט הבא:

- Device - בוחרים את הרכיב שעל הלוח.
- Operation - בוחרים את הפעולה לביצוע (מחיקה, צריבה וכדי).
- Filename - קובץ הצריבה JED שהתקבל בשלב הסינתזה.

כמעט גמרנו. כל שנותר לעשות זה לבצע Compose מתוך תפריט Operation, דבר הממיר את קובץ JED לקובץ JAM המתאים לצריבה ב JTAG. ולסיום, לבצע PLAY. בכל התהליך יש לעקוב אחר ההודעות בחלון למטה. אם התקבלה ההודעה שהרכיב נצרב בהצלחה הסתיימה הפעולה ויש לכם רכיב מוכן.

זהו, במזל טוב רכיב חדש נולד. הרכיב שעל הלוח אמור לפעול בהתאמה מלאה לסימולציה בקוד RTL שהתקבלה בשלב הקודם. אם זאת, במעגל אמיתי ישנם מאפיינים שלא נלקחו בחשבון בסימולציה ועשויים להיות שינויים מסוימים (בעיקר בעבודה בתדר גבוהה) ביחס לסימולציה. לשם כך אנו נבחן את המעגל בזמן אמת במעבדה.

בפרק הבא הקורא ימצא אוסף משימות בתכנון לוגי ברמות שונות, החל בלוגיקה צרופית וכלה במכונות מצבים. בכל תרגיל יצויינו המבואות, המוצאים והפורטים המתאימים להם ב ENTITY. על התלמיד לבצע את המשימה עפ"י השלבים שצויינו למעלה, להגיעה לסימולציית RTL תקינה, לצרוב את הרכיב ולבדוק את המודל על כרטיס הפיתוח.

תרגילים

תרגיל 1 - הכי קל שיש

כניסות: מפסקי הזזה, לחצן PUSH.

יציאות: LEDS.

- כתוב מודל המחווה את מצב המפסקים ע"י LEDS. כלומר, כל LED ידלק או יכבה בהתאם למצב המפסק המתאים לו.
- כתוב מודל המחווה ב LEDS (B_0 - B_2) את הפונקציות הבינאריות OR, AND, XOR, של שני הביטים הנמוכים במפסקים S_0 ו S_1.
- הרחב את המודל כך שלחיצה על לחצן PUSH הופכת את החיווי (NOT המצב המקורי).

תרגיל 2 - לוגיקה צרופית (מקודדים ומפענחים)

כניסות: מפסקי הזזה SW.

יציאות: LEDS.

- כתוב מודל למפענח בינארי 3 ל 8 כאשר הכניסות הן: S_0 - S_2 והמוצאים הם LEDS.
- כתוב מודל למקודד בינארי 8 ל 3 כאשר הכניסות הן: S_0 - S_7 והמוצאים הם: B_0 - B_2.

תרגיל 3 - לוגיקה צרופית (מפענח Seven Segment)

כניסות: מפסקי הזזה SW, לחצן PUSH.

יציאות: תצוגת מקטעים SEGA, SEGB.

- כתוב מודל המחווה את מצב המפסקים ע"י SEGA ו SEGB בקוד BCD. כלומר, כל תצוגת מקטעים תציג את הספרה המתאימה לקוד BCD המיוצג ע"י ארבעה מפסקים. הנקודה העשרונית בתצוגה תופעל ע"י הלחצן PUSH.

תרגיל 4 - לוגיקה צרופית (אריתמתיקה)

כניסות: מפסקי הזזה SW, לחצן PUSH.

יציאות: תצוגת מקטעים SEGA, SEGB, LEDS.

- כתוב מודל המקבל בכניסה שני מספרים בקוד בינארי בעל ארבעה ביטים (מפסקי ההזזה מחולקים לשני ניבלים) ומוציא בתצוגת LEDS את הסכום הבינארי שלהם.
- הרחב את המודל כך שלחיצה על לחצן PUSH הופכת את החיבור לחיסור.
- הרחב את המודל כך שהסכום וההפרש יופיעו בתצוגת המקטעים (SEGA ו SEGB) בקוד BCD.

<u>תרגיל 5 - לוגיקה סדרתית (מונים)</u>

כניסות: מפסקי הזזה SW, לחצן PUSH, שעון CLK.

יציאות: תצוגת מקטעים SEGA, SEGB, LEDS.

- כתוב מודל למונה בינארי עולה, באורך שישה ביטים, המשנה את ערכו בכל שנייה. מוצא המונה יצויין ע״י LEDS (B_0 - B_5). שם לב שהשעון הפנימי הוא בתדר של 10MHz ולכן צריך ראשית כל לחלק אותו בצורה כל שהיא.
- הרחב את המודל כך שמפסק ההזזה S_0 הופך את הספירה מעולה ליורד.
- הרחב את המודל כך שלחיצה על לחצן PUSH מאפסת את המונה.
- הרחב את המודל כך שהמוצא יופיע בתצוגת המקטעים (SEGA ו SEGB) בקוד BCD.

<u>תרגיל 6 - לוגיקה סדרתית (אוגר הזזה)</u>

כניסות: מפסקי הזזה SW, כניסה דיגיטלית IO_2, לחצן PUSH, שעון CLK.

יציאות: LEDS, יציאה דיגיטלית IO_1.

- כתוב מודל לאוגר הזזה, באורך שמונה ביטים, המשנה את ערכו בכל שנייה. מצב האוגר יצויין ע״י LEDS (B_0 - B_7). שם לב שהשעון הפנימי הוא בתדר של 10MHz ולכן צריך ראשית כל לחלק אותו בצורה כל שהיא. הכניסה לאוגר תהא S_0 והמוצא IO_1. בדוק ע״י LEDS וסקופ את פעילות האוגר.
- הרחב את המודל כך שלחצן PUSH ישמש ככניסת אפשור (enable).
- כתוב מודל לאוגר הזזה, באורך 50 ביטים (ראה איור 6), המשנה את ערכו בכל מחזור שעון בסיסי. הכניסה לאוגר תהא IO_2 והמוצא IO_1. חבר בכניסת האוגר (IO_2) מחולל תדר ברמות TTL. כוון את התדר לערך של 100 KHz גל רבועי. בדוק במוצא האוגר (IO_1) מה התדר ומה הפזה שלו ביחס למבוא.
- מה ההשהייה המבוקרת הקטנה ביותר הניתנת להשגה במעגל זה? בדוק זאת!.

איור 6: תאור סכמתי של אוגר ההזזה באורך 50 ביט

<u>תרגיל 7 - לוגיקה סדרתית (מחלק תדר)</u>

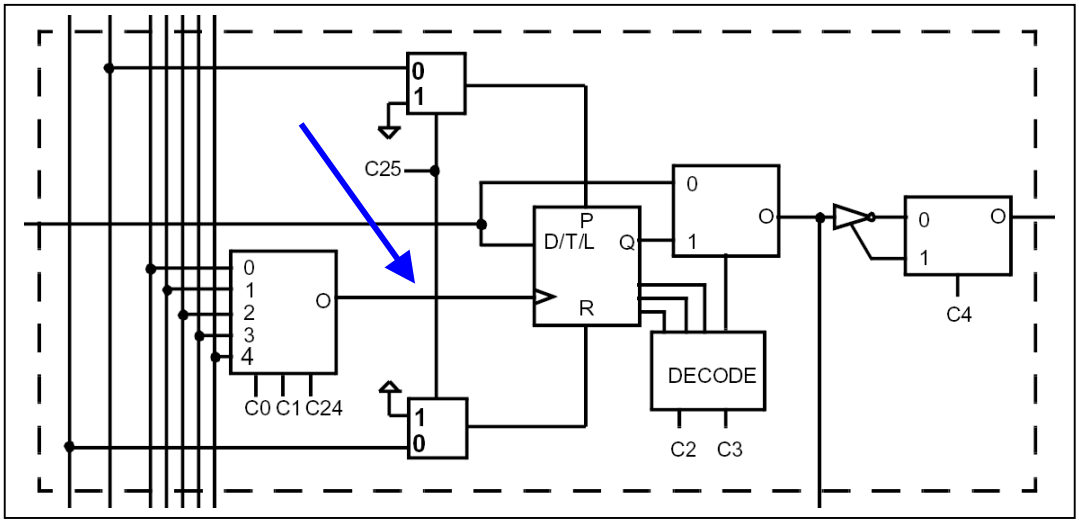
כניסות: מפסקי הזזה SW, כניסה דיגיטלית IO_2.

יציאות: יציאה דיגיטלית IO_1.

- כתוב מודל למחלק תדר מבוקר, למקדמי חלוקה של 1-256, המקבל את ערך החלוקה מהמספר הבינארי המיוצג ע"י מפסקי ההזזה S_0 - S_7. הכניסה למחלק תהא IO_2 והמוצא IO_1. חבר בכניסת המחלק (IO_2) מחולל תדר ברמות TTL. כוון את התדר לערך של 100 KHz גל רבועי. בדוק במוצא המחלק (IO_1) מה התדר ומה הפזה שלו ביחס למבוא. שנה את ערכי המפסקים ובדוק את ההשפעה על המוצא.

הארה חשובה: בארכיטקטורת ה CPLD ישנם 4 כניסות המיועדות לשעון (ראה איור 7), אשר שניים מתוכם מחווטות בפועל: CLK לרגל 23 ו IO_2 לרגל 20. בארכיטקטורה הנוכחית, רק הכניסות היעודיות (אחת מן הארבע) יכולה לשמש כשעון, במובן של התיחסות לתכונת EVENT שלה במודל VHDL. במילים אחרות, רק לכניסות אלה נוכל לבדוק אירוע של עליה או ירידה. עם זאת, פינים אלה יכולים לשמש רק ככניסה ולא כיציאה. בכדי להנות משני העולמות, בכרטיס הפיתוח IO_2 חווט גם לפין 19 על מנת שיוכל לשמש כיציאה (בתרגילים אחרים) וגם לפין 20 על מנת שיוכל לשמש כשעון חיצוני.

עד כה לא נעשה כל שינוי בקובץ אילוץ הפינים. **בכדי לממש את הכניסה כשעון, צריך לאלץ, בקובץ אילוץ הפינים, את IO_2 לפין 20 במקום פין 19.**



איור 7: כניסות השעון לכל macro cell ברכיב

<u>תרגיל 8 - מכונת מצבים אלגוריתמית (טיימר)</u>

כניסות: מפסקי הזזה SW, כניסה דיגיטלית IO_2, שעון CLK.

יציאות: יציאה דיגיטלית IO_1, תצוגת מקטעים SEGA, SEGB.

- כתוב מודל לקוצב זמן (טיימר) מבוקר, המקבל שני זמני השהייה T_1 ו T_2 המיוצגים ע"י מפסקי ההזזה S_0 - S_3 ו S_4 - S_7 בהתאמה. T_1 במילי שנייה, יהיה המספר הבינארי המיוצג כסיפרת BCD ע"י הניבל הנמוך של המפסקים, ובהתאמה T_2 במילי שנייה, יהיה המספר הבינארי המיוצג כסיפרת BCD ע"י הניבל הגבוה של המפסקים. הכניסה לקוצב תהא IO_2 והמוצא IO_1. במצב רגיל המוצא במצב '0'. מרגע שמתגלית עליה בכניסה, ממתינים זמן T_1. לאחר ההמתנה המוצא עולה למצב '1' למשך זמן T_2 וחוזר למצב '0'. חבר בכניסת הקוצב (IO_2) מחולל תדר ברמות TTL. כוון את התדר לערך של 100 Hz ורוחב פולס של 100µs. בדוק במוצא הקוצב (IO_1) את האות המתקבל. שנה את ערכי המפסקים ובדוק את ההשפעה על המוצא.
- הרחב את המודל כך שהזמנים T_1 ו T_2 יופיעו בתצוגת המקטעים (SEGA ו SEGB) בקוד BCD.

<u>תרגיל 9 - מכונת מצבים אלגוריתמית (גלאי סדרות)</u>

כניסות: מפסקי הזזה SW, כניסה דיגיטלית IO_2, שעון CLK.

יציאות: יציאה דיגיטלית IO_1, LEDS.

- כתוב מודל לגלאי סדרות מבוקר, המגלה סידרה באורך K של '1' המופיע בכניסה IO_2. מספר ה-'1' K, יהיה מיוצג ע"י מפסקי ההזזה S_0 - S_7. הכניסה לגלאי תהא IO_2 והמוצא IO_1. במצב רגיל המוצא במצב '0'. ברגע שמתגלית סדרה מתאימה, המוצא עולה למצב '1' ונשאר במצב זה כל עוד K הביטים האחרונים היו '1'. חבר בכניסת הגלאי (IO_2) מחולל תדר ברמות TTL. כוון את התדר לערכים שונים. בדוק במוצא הגלאי (IO_1) את האות המתקבל. שנה את ערכי המפסקים ובדוק את ההשפעה על המוצא.
- הרחב את המודל כך שבגילוי סדרה ה LEDS יהבהבו בקצב של שתי הבהובים בשניה.

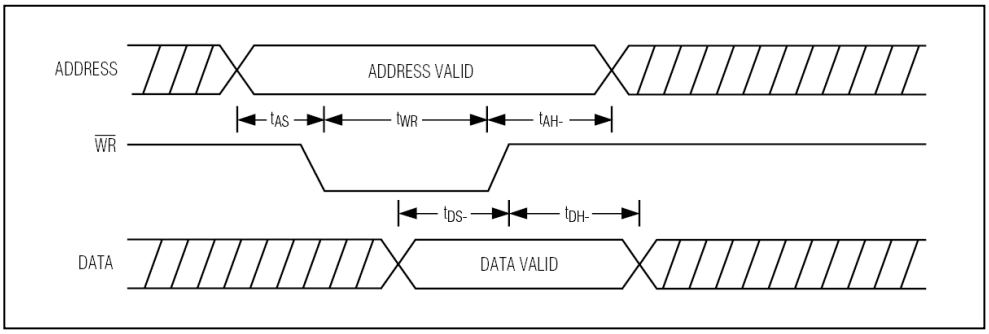
הארה חשובה: בתרגילים אלה המערכת היא סנכרונית ולכן משתמשים בשעון הפנימי. דבר זה מונע את האפשרות להשתמש בו זמנית גם בכניסה IO_2 כבשעון. **צריך לאלץ, בקובץ אילוץ הפינים, את IO_2 לפין 19.**

<u>תרגיל 10 - בקרים (לממיר DAC)</u>

כניסות: מפסקי הזזה SW, שעון CLK.

יציאות: יציאה אנלוגית $AOUT_0$, תצוגת מקטעים SEGA, SEGB.

- כתוב מודל למחולל אותות אנלוגיים (משולש ושן מסור). תדר המחולל יקבע ע"י המספר K המיוצג ע"י מפסקי ההזזה S_0 - S_7. עוצמת האות תהא תמיד 0V - 5V. עליך להשתמש בממיר DAC תקני הקיים על הלוח - MAX5102 מתוצרת MAXIM שאת דיאגרמת הזמנים שלו ניתן לראות באיור 8. חבר סקופ למוצא האנלוגי ($AOUT_0$) ובדוק את האות המתקבל. שנה את ערכי המפסקים ובדוק את ההשפעה על המוצא.
- הרחב את המודל כך שיוציא מתח בשתי היציאות האנלוגיות בפזה של 90 מעלות ביניהן. חבר סקופ למוצאים האנלוגיים ($AOUT_0$ ו $AOUT_1$) ובדוק את האות המתקבל. בדוק את הפזה בין הערוצים. שנה את ערכי המפסקים ובדוק את ההשפעה על המוצא.



איור 8: כניסות השעון לכל macro cell ברכיב

הארה חשובה: את דפי הנתונים המלאים של הרכיב ניתן למצוא באתר של MAXIM. אם זאת, ניתן להניח שכל הזמנים המצויינים למעלה הם מסדר גודל של 50ns - 100ns.