חוברת דוגמה למימוש מערכת תקשורת בחומרה (FPGA)

כתב: יפים אוליצקי
ערך ובדק: דוד הירשברג

אוגוסט 2003

הקדמה

חוברת זו מיועדת לסטודנטים המתחילים לבצע פרוייקטים בנושאי מימוש סיפרתי בשפת Verilog וצריבה על כרטיס FPGA של חברת Runcom. כדוגמה נעבור על פרוייקט בנושא מימוש ספרתי של שרשרת תקשורת הכוללת מקודד קונוולוציה ומפענח ויטרבי אשר בוצע ע''י דימיטרי קירז'נר ויפים אוליצקי בהנחייתו של אודי שרוט במעבדה לתקשורת בשנת 2001.

בחוברת זו נעבור על הרקע התאורטי של הפרוייקט, על קוד ה- Verilog של כל חלקי המערכת, ונראה את אופן הביצוע של האלגוריתם. בהמשך נעבור על תוכנות לביצוע סימולציה, ותהליכים הדרושים לצריבה של הקוד על גבי כרטיס FPGA – סינטזה, מיפוי וטעינה.

החוברת אינה מהווה חומר לימוד על שפת Verilog ולא רקע התאורטי על קודי קונוולוציה ואלגוריתם ויטרבי, לכן מומלץ מאוד להשתמש בחומר נוסף בנושאים אלו.

התוכנות שנעבוד אתם:

- **תוכנה לביצוע סימולציה ספרתית - Modelsim SE 5.6**
- **תוכנה לביצוע הסינתזה - Synplify Pro 7.2**
- **תוכנה לביצוע מיפוי של קוד על הכרטיס – Xilinx Project Navigator**
- **תוכנות יעודיות לעבודה עם הכרטיס**

החוברת כוללת הסבר מלא על העבודה עם תוכנות אלה, ומומלץ להכיר את התוכנות בעזרתה.

<u>תוכן עיניינים</u>

1. רקע תאורטי והסברים על מימוש הדוגמה – עמוד 4

- **תאור כללי של שרשרת שידור וקליטה.**
- **הסבר על תאוריה של קודי קונבולוציה.**
- **הסבר על תאוריה של אלגוריתם ויטרבי.**
- **ספציפיקציה של קוד קובלוציה ואלגוריתם ויטרבי ממומשים בפרוייקט.**
- **תאור של קוד Verilog של הדוגמה.**

2. סימולציה ספרתית באמצעות Modelsim SE 5.6 – עמוד 13

3. תאור שלבים שונים הדרושים לצריבת התכנון – עמוד 20

- **תהליך סינתזה בעמצאות Synplify Pro 7.2.**
- **תהליך מפוי וחיווט בעמצאות Xilinx Project Navigator**

4. טעינה של התכנון לכרטיס – עמוד 39

- **הסבר על חומרת הכרטיס.**
- **הסבר על תהליך הצריבה.**

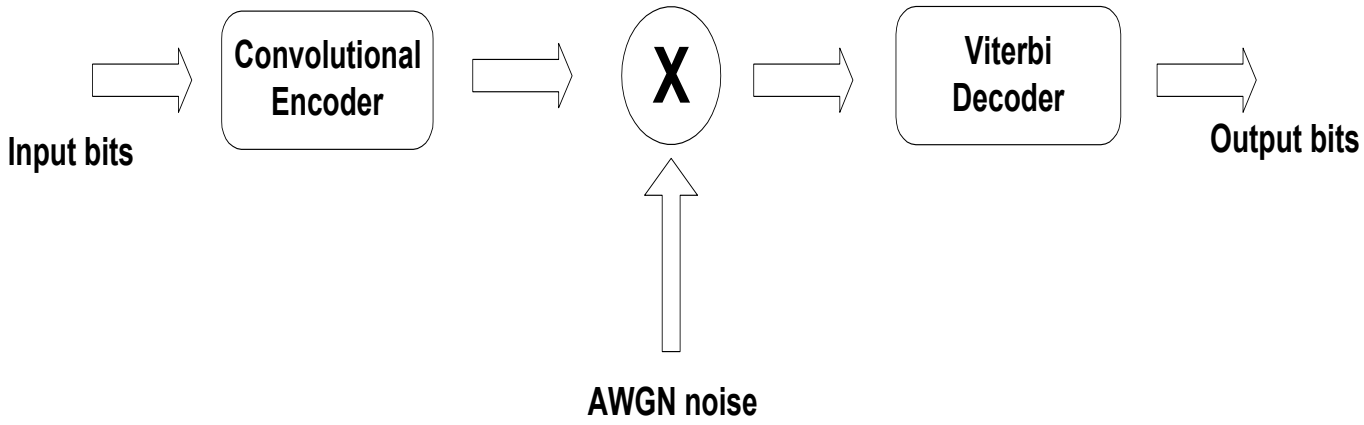
6. בדיקת התכנון – עמוד 48

1. רקע תאורטי והסברים על מימוש הדוגמה

<u>תיאור מערכת מקודד קונבולציה עם מפענח ויטרבי</u>

קוד קונבולציה הנו קוד לתיקון שגיאות המבוסס על הטענת ביטי המידע ל Shift register ושליפת המידע על יד קומבינציה לינארית של ביטי המידע ברגיסטר. מצב המידע ברגיסטר מכונה State – מצב המקודד. הפענוח של קוד קונבולציה מבוסס על אלגוריתם המחפש את סדרת השידור הסבירה ביותר וזאת על ידי חיפוש על פני כל הסדרות האפשריות תוך שימוש בתכונות הסריג הנוצר על ידי המעברים בין כל המצבים השונים האפשריים של המקודד. המפענח הנ"ל מכונה בשם מפענח ויטרבי.
לפרטים נוספים ראה פרוהקיס פרק 8.

בחוברת זו מתואר מימוש מקודד ומפענח ויטרבי המורכב משלושה חלקים.
החלק הראשון הינו המקודד, שתפקידו לקודד את ביטי האינפורמציה למילת קוד לפי אלגוריתם קידוד מסוים ולשלוח את מילת הקוד לערוץ. הערוץ הינו החלק השני של המערכת והנו ערוץ AWGN. הערוץ ממומש בתוכנה ולצורך סימולציה של ביצועי המערכת בלבד. החלק השלישי הינו מפענח ויטרבי, אשר תפקידו לפענח את הביטים הנכנסים מהערוץ בעזרת אלגוריתם לפענוח של ויטרבי.

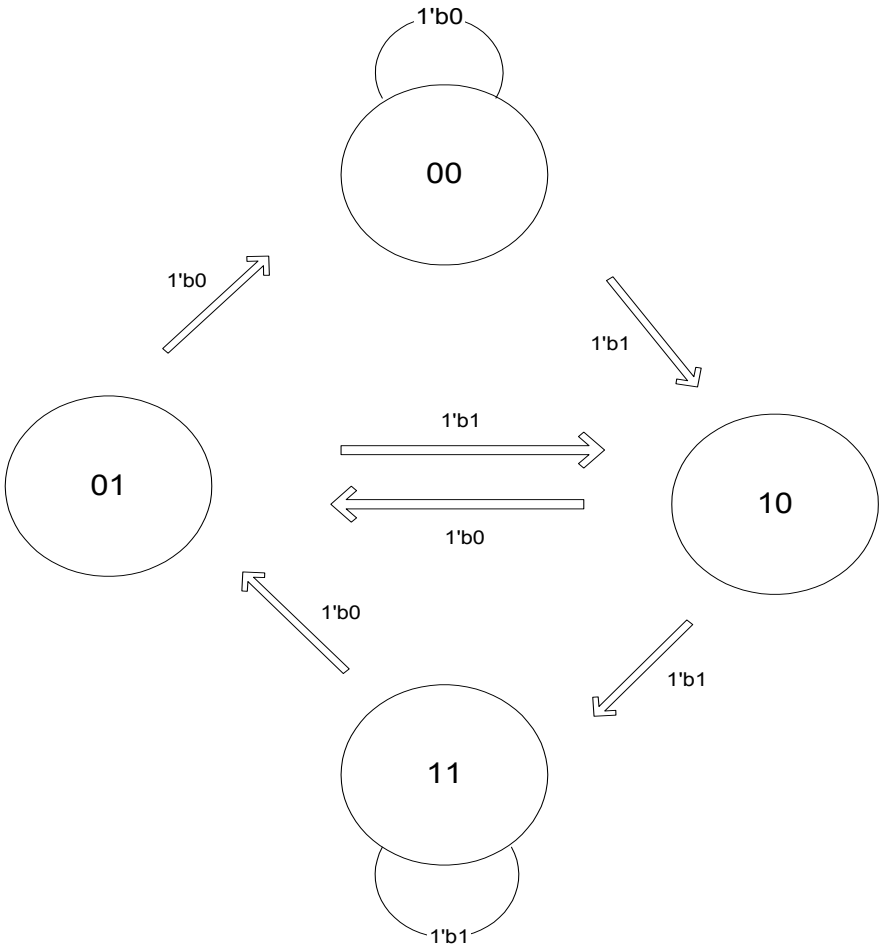


תאור מפורט של רכיבי המערכת

התאור המפורט כולל שילוב של קטעי קוד בשפת ורילוג כפי שמומשו בפרויקט.

מקודד

המקודד הינו מקודד קונבלוציה ועובד לפי דיאגרמת מצבים הבאה:



בכל מחזור שעון המקודד מקבל ביט אחד של האינפורמציה, ולפיו הוא עובר למצב הבא. למקודד שתי יציאות, וכל אחת היא ביט של מצב נוכחי:

```
module coder(b_in, clk, reset, RsigA, RsigB);
input b_in,clk, reset;
```

b_in – ביט של איפורמציה שאותו צריך לקודד

```
output [11:0] RsigA, RsigB;
```

RsigA, RsigB – שני היציאות שהן נקבעות ע''י ביטים של המצב.

הבלוק הראשון של המקודד קובע את הערך של יציאות לפי המצב, לדוגמא עבור מצב 00:

```
case (CurrSt)
2'b00:
begin
RsigA <= 12'b000010000000;
RsigB <= 12'b000010000000;
```

כאשר מספר 12'b000010000000 מייצג 0 ו12'b000100000000 מייצג 1.

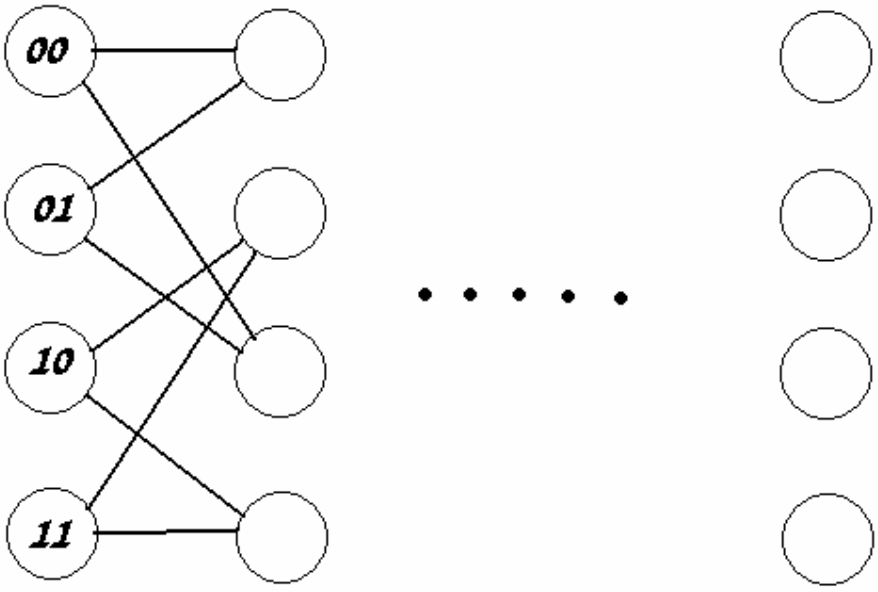
הבלוק השני מממש את מכונת המצבים של המקודד כבורר.

```
case ({CurrSt , b_in})
{2'b00,1'b0}: CurrSt <= 2'b00;
{2'b00,1'b1}: CurrSt <= 2'b10;
{2'b01,1'b0}: CurrSt <= 2'b00;
{2'b01,1'b1}: CurrSt <= 2'b10;
{2'b10,1'b0}: CurrSt <= 2'b01;
{2'b10,1'b1}: CurrSt <= 2'b11;
{2'b11,1'b0}: CurrSt <= 2'b01;
{2'b11,1'b1}: CurrSt <= 2'b11;
endcase
```

מפענח
רקע תאורטי

המפענח הוא החלק המסובך במערכת. נתאר כעת את אופן פעולת האלגוריתם.
אלגוריתם ויטרבי הינו אלגוריתם אופטימלי במובן סבירות מירבית לפענוח רצף מידע לקוד קונבולוציה. על מנת לשערך את רצף נתונים הסביר ביותר האלגוריתם מנצל את מבנה המקודד המיוצג בדיאגרמת ה-trellis המתארת את טבלת המעברים האפשריים בין מצבי המקודד.
דוגמא ל-trellis של מפענח עבור שתי ביטים :

חלון של מפענח ויטרבי עבור כניסה בת שתי סיביות



CurrSt	b_in	Ns
2'b00	1'b0	2'b00;
2'b00	1'b1	2'b10;
2'b01	1'b0	2'b00;
2'b01	1'b1	2'b10;
2'b10	1'b0	2'b01;
2'b10	1'b1	2'b11;
2'b11	1'b0	2'b01;
2'b11	1'b1	2'b11;

בכל רגע t=i+1 ובכל מצב S(1+i) אלגוריתם ויטרבי מחשב מטריקות. מטריקה הינה השגיאה הניצברת במהלך התפתחות ה trellis ומשמשת כמדד למסלול המתאר את רצף המידע ששודר בסבירות הגבוה ביותר . חישוב מטריקה נעשה באופן הבא:

$$M(Si+1,Si) = M(Si) + F(Ei = (Si+1,Si))$$

לכל המעברים האפשריים Ei = (Si,S(i+1)) .
כאשר M (Si) הינה מטריקה המצטברת עד מצב i ו F(Ei) הינה המטריקה המתווספת במעבר Ei .

עבור קוד בינארי שתואר בפרק זה ,ישנם תמיד שני מסלולים המתמזגים במצב S(i+1) . מתוך שני מסלולים הניכנסים למצב,יבחר המסלול עבורו מתקבלת מטריקה מצטברת מקסימלית :

$$M(Si+1) = \max_{Si} M(Si+1,Si)$$

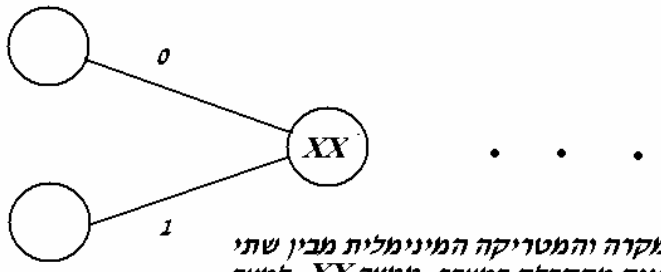
מטריקה מקסימלית M(i+1) והמסלול המתאים (שנוצר בעיקבות בחירת Si עבורו מתקבלת מטריקה המצטברת במעבר) ,(X(Si+1 נשמרים בשביל המשך החישוב.
המטריקה המתווספת האופטימלית תחושב באופן הבא :

$$F(Ei) = \sum_{j=1}^{j=n} Vi,j * Ui,j$$

המטריקות המתקבלות הן הקורלציה בין הביטים המשודרים במעבר בין מצב למצב הבא Uj,i מכל אחד מהגנרטורים במקודד לבין מידע הנקלט במקלט. Vj,i

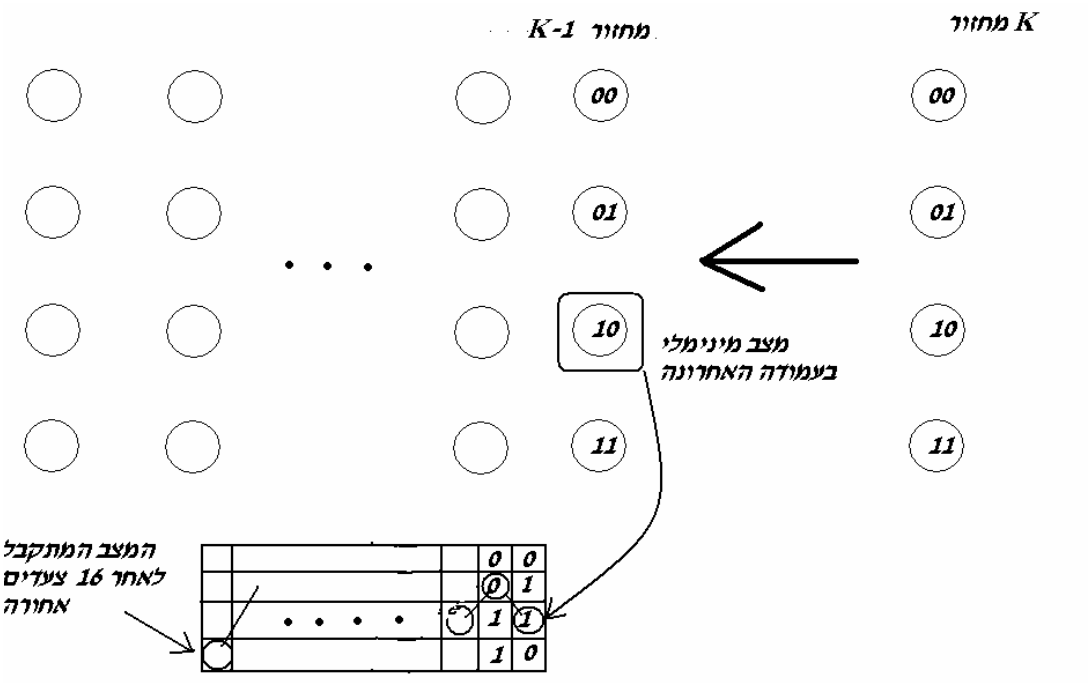
ממוש של המפענח בפרוייקט

בפרוייקט ממומש מפענח ויטרבי עבור שתי סיביות כניסה למשדר (שתי סיביות קידוד עבור כל בית כניסה). עם כך ישנם ארבעה מצבים אפשריים עבור המפענח. פענוח מתחיל ממצב 00, כאשר מטריקה של מצב 00 מאותחלת ל0, ושל שאר המצבים לאינסוף. בכל מחזור שעון המפענח מקבל שני ביטים מערוץ תקשורת רועש ומתחיל בתהליך הפענוח. המטרה של תהליך הפענוח היא להתאים את רצף הביטים שהתקבלו לאחד המצבים הקיימים ובכך לסנן את הרעש ולפענח את הביט המקורי שהיה בכניסה.
במסגרת תהליך הפענוח מחשבים תחילה את כל המטריקות של כל מצב הבא בעזרת מטריקות של המצבים הנוכחיים וביט כניסה. ישנם שתי דרכים אפשריות להגיע לכל מצב ועל מנת לחשב את המטריקה של המצב לוקחים את הדרך המינימלית. כך כל מחזור שעון בונים עמודה נוספת בטבלת מעברים, אשר בנויה בצורת אוגר הזזה .כל משבצת בעמודה מכילה סימן על איזה מבין המצבים האפשריים נותן מטריקה מינימלית



במקרה והמטריקה המינימלית מבין שתי האפשרויות מתקבלת במעבר ממצב XX למצב העליון, נשמר במשבצת מתאימה 0 אחרת 1

ציור סכמתי של סריקת הטבלה



תאור קוד Verilog

עבור כל מצב אפשרי, המפענח מבצע הפרש בין המטריקה שהתקבלה מהערוץ והערך של המצב. ערך זה מהווה למעשה הסתברות של המצב והוא נקרא מטריקה של המצב. בכל מחזור שעון לגבי כל מצב אנו מחפשים את המטריקה המינימלית מבין שתי המטריקות אפשריות (לכל מצב ניתן להגיע השני מצבים בלבד) ורושמים את המעבר הסביר ביותר בטבלת מעברים.
להלן דוגמא של קטע קוד המבצע את הפעולות:

```
//state0

if( met0_reduced + absRsigA0 + absRsigB0 < met1_reduced +
absRsigA1 + absRsigB0 )
begin
met0 <= met0_reduced + absRsigA0 + absRsigB0 ;
//upper state          mem0 <= 0;
end
else begin
met0 <= met1_reduced + absRsigA1 + absRsigB0 ;
//lower state          mem0 <= 1;
end
```

כאשר:
met0_reduced – מטריקה של מצב 0
absRsigA0 – הפרש בין RsigA (כניסה למודול) ל-0
AbsRsigA1 – הפרש בין RsigA (כניסה למודול) ל-1
met0 – מטריקה החדשה של מצב 0
mem0 – מקום עבור מצב 0 בטבלת מעברים

יחד עם עידכון המטריקות המפענח מוצא את המצב בעל המטריקה המינימלית ומצב זה נכנס לאחד מ בלוקי אחסון המסלולים אשר סורקים את טבלת מעברים. בכל path block N קיימים משתנים:
PathN – באיזה מצב בלוק נמצא כעת
IndexN – מספר צעדים אחורה בטבלת מעברים שהבלוק ביצע.
mem_index15[path15] – הביט בטבלת מעברים שמכיל מידע על המסלול, כלומר מהיה המצב הקודם הסביר ביותר.
ההתקדמות לעורך הטבלה נעשית ע''י הבורר הבא:

```
case ({path15, mem_index15[path15]})
3'b000: path15 <= 2'b00;
3'b001: path15 <= 2'b01;
3'b010: path15 <= 2'b10;
3'b011: path15 <= 2'b11;
3'b100: path15 <= 2'b00;
3'b101: path15 <= 2'b01;
3'b110: path15 <= 2'b10;
3'b111: path15 <= 2'b11;
default: path15 <= 2'b00;
endcase
```

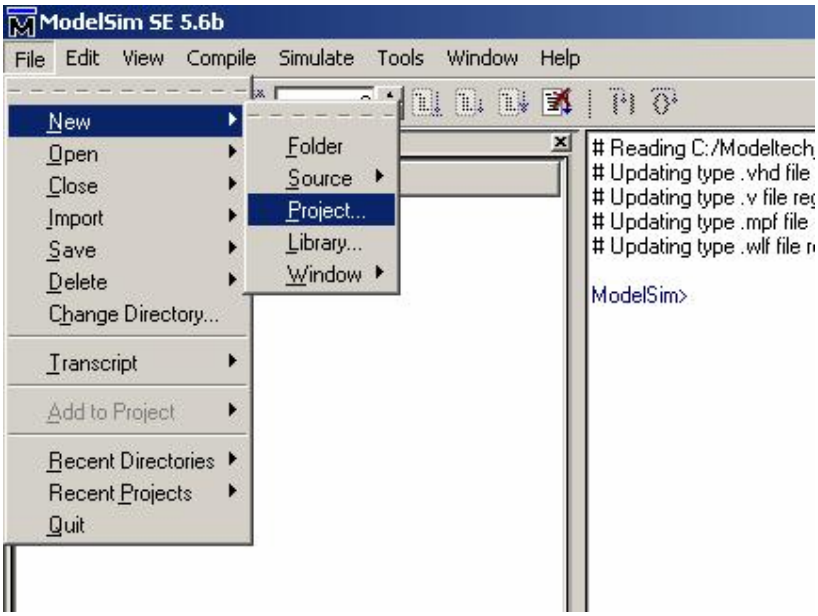
כלומר לפי המצב הנוכחי והמסלול אנו עוברים למצב הקודם.
כאשר IndexN מגיע ל-0 פירוש הדבר שהבלוק הגיע לסוף הטבלה וניתן לחשב את סיבית המעבר – למעשה סיבית האינפורמציה.

הבלוק שמבצע חישוב ביט היציאה נמצא לאחר הpath blocks . דבר ראשון שעושה המודול, הוא בודק איזה מ הpath blocks הגיע לסוף הטבלה, ולאחר מכן בודק את המצב שהתקבל ב path block המתאים ומצב שהתקבל במחזור הקודם . אם המעבר בין שני המצבים הללו אפשרי אז ביט מעבר הוא ביט יציאה ואם לא error עולה.

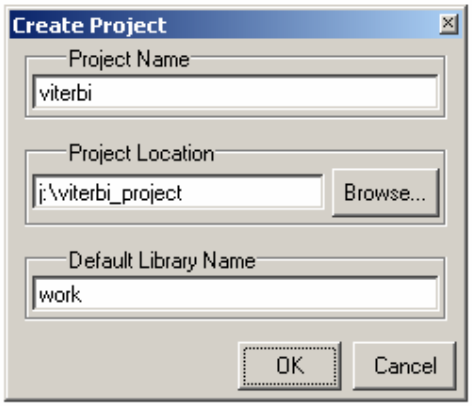
2. סימולציה ספרתית באמצעות Modelsim SE 5.6.

לאחר כתיבת הקוד עלינו לבדוק את נכונותו ע''י סימולציה. לביצוע הסימולציה אנו משתמשים בתוכנת Modelsim 5.6 המותקנת בחוות ה-PC במעבדה. לא מומלץ לעבוד עם גרסה 5.2 המותקנת כיום בחוה הפקולטית!
Modelsim הוא כלי סימולציה מתקדם לתכנות בשפות Verilog ו VHDL, המאפשר ביצוע של סימולציה לוגית, סימולציה זמנית ועבודה עם קבצים לצורכי קלט/פלט.

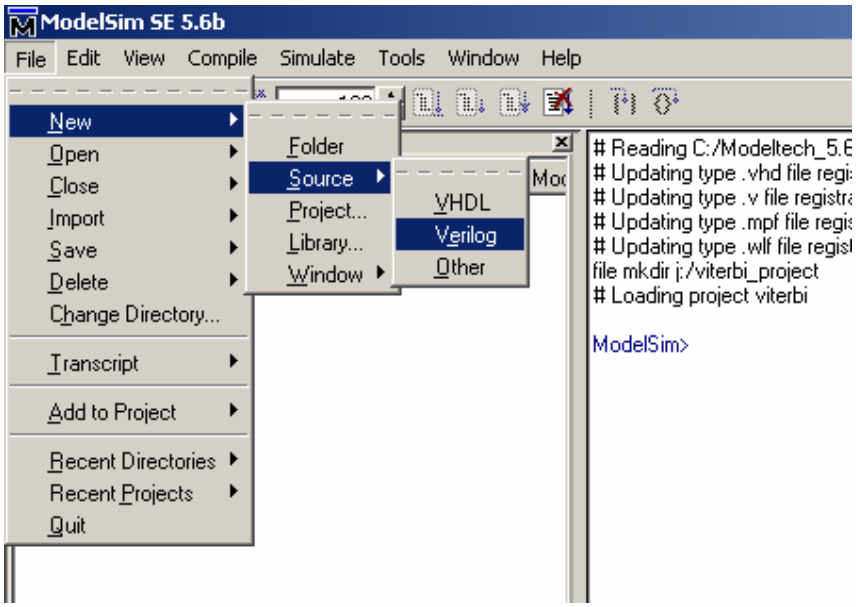
על מנת להתחיל לעבוד עם התוכנה יש ליצור פרוייקט חדש. לשם כך יש להכנס לתפריט
File => New => Project



יש לבחור את שם הפרוייקט ואת המקום בדיסק שבו הפרוייקט יישמר:

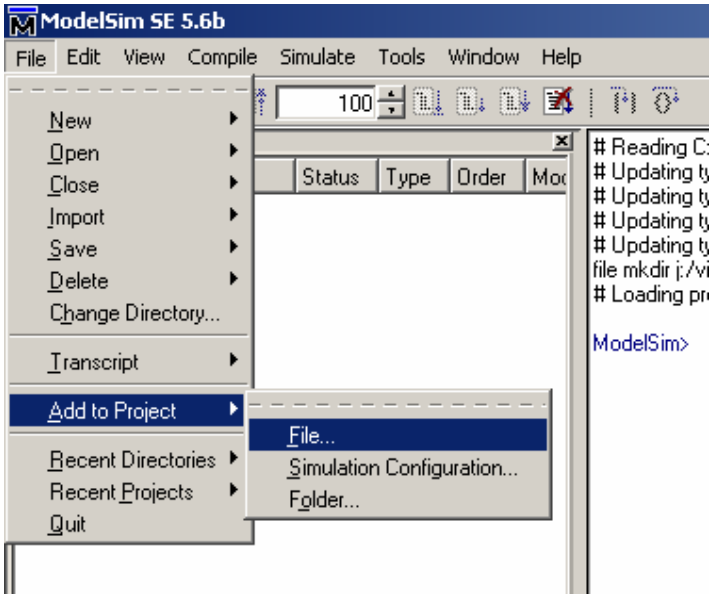


כעת יש להוסיף את קבצי הקוד לפרוייקט. תוכנת Modelsim מכילה עורך טקסט מאוד נוח, ניתן להשתמש בו לכתיבת הקוד. על מנת ליצור קובץ Verilog חדש יש להכנס לתפריט
File => New => Source => Verilog

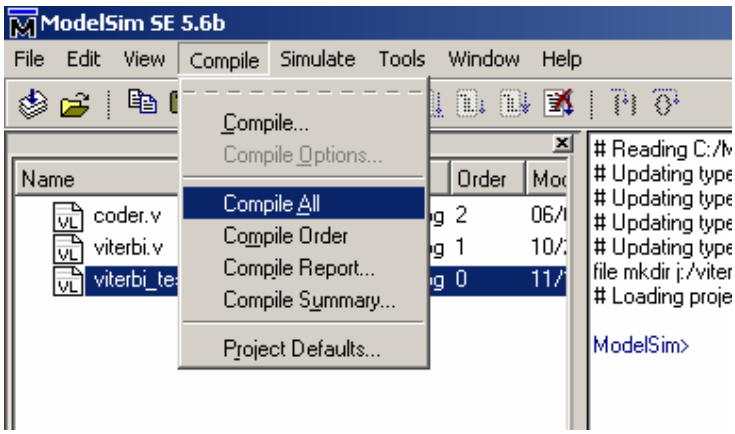


וכדי להוסיף קובץ קוד קיים

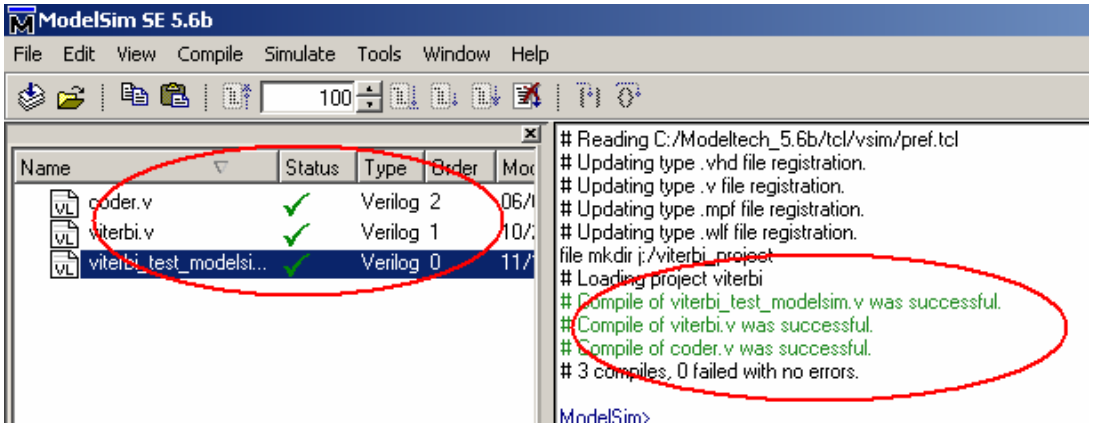
File => Add to Project => File



לאחר שכל קבצי המקור מוכנים ומוגדרים בפרוייקט יש לקמפל אותם.
קומפילציה של הפרויקט מבוצעת על ידי בחירה של התפריט
Compile => Compile All

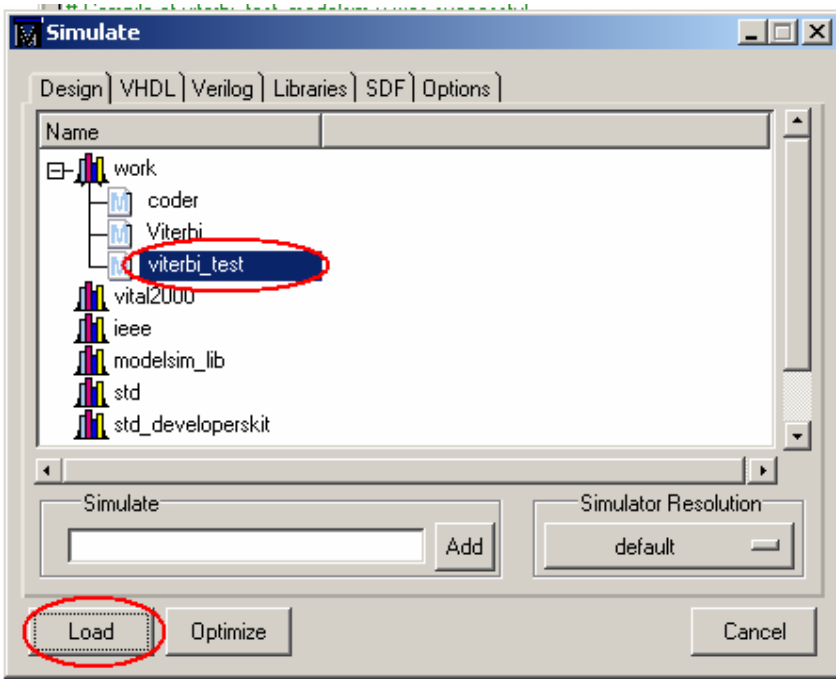


כתוצאה מהקומפילציה נקבל את הסטטוס הבא בחלון Simulation Prompt(החלון הימני של Modelsim)

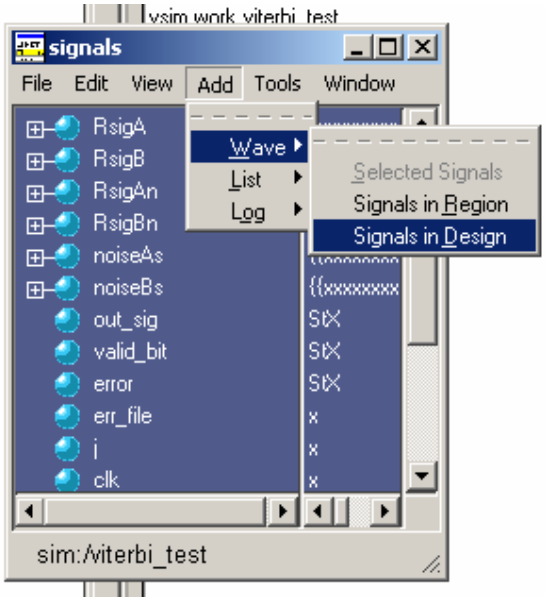


במידה ויש שגיאות יוצגו הודעות בצבע אדום יש לבצע את התיקונים הנדרשים עד למעבר קומפילציה בהצלחה.

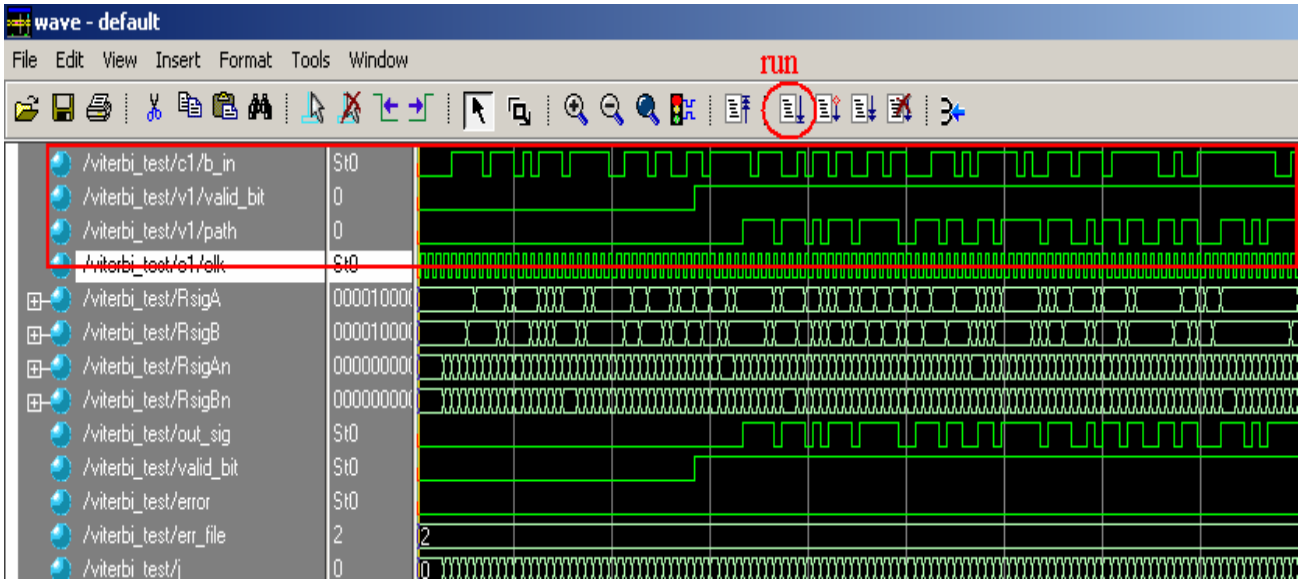
כעת התוכנית מוכנה להרצת הסימולציה. יש להכנס ל Simulate בתפריט הראשי, לבחור את ה top module של התכנון וללחוץ על Load כדי לטעון את התכנון לסימולטור.



בכדי לבדוק את נכונות התכנון יש לבדוק את צורת האותות המתקבלים מהתכנון בעזרת תצוגת האותות הגרפית של הסימולטור (Wave).
לפני הצגת האותות יש לקבוע לסימולטור אלו אותות יוצגו. ניתן לבצע זאת על ידי בחירת View => Signals בתפריט הראשי.
פעולה זו פותחת את החלון המוצג להלן. בכדי לבחור את כל האותות בתכנון יש לבחור את התפריט
Add => Wave => Signals in Design



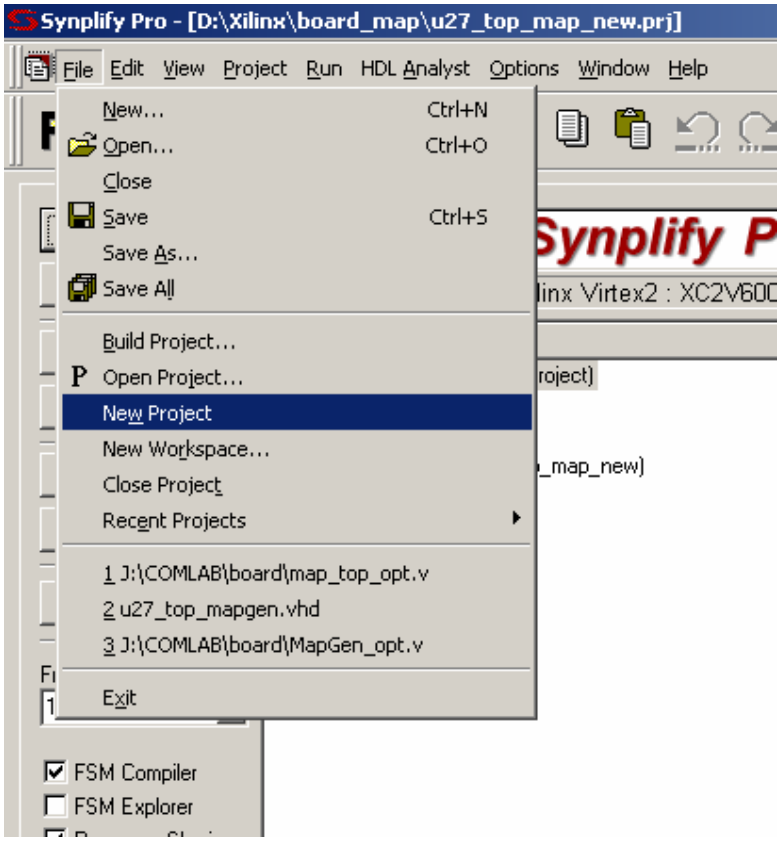
יש להריץ את הסימולציה. בחלון Wave יש ללחוץ על כפתור run .
כל לחיצה מריצה למשך זמן מסוים שנקבע כפרמטר בהגדרות
הסימולטור. בחלון הראשי של Modelsim היתן לכתוב run XXX
כאשר הפרמטר XXX קובע את משך הסימולציה. לפי צורת הגלים של
האותות בחלון ה Wave ניתן לבדוק שהתכנון מבצע את תפקידו.



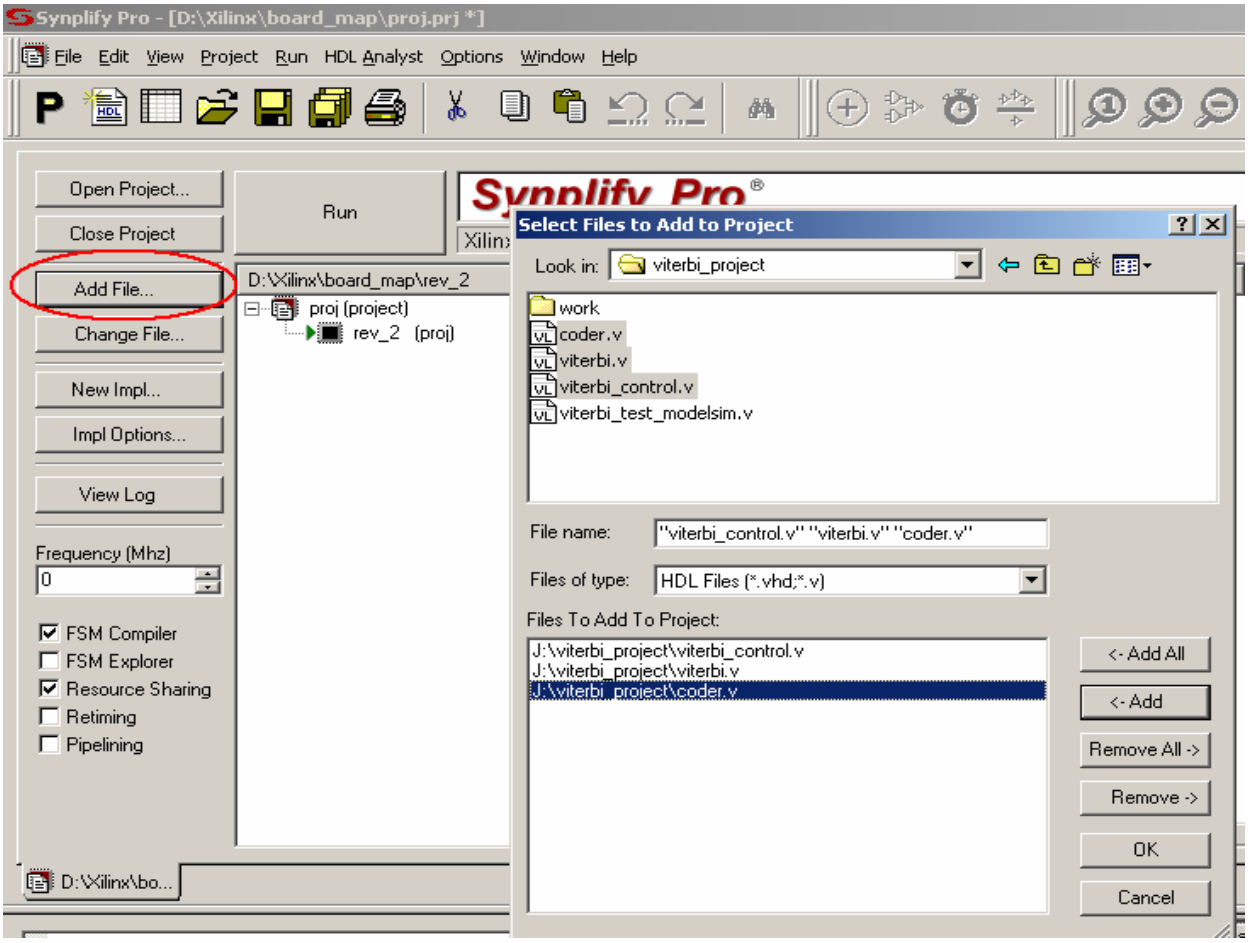
3. תאור שלבים שונים הדרושים לצריבת התכנון

סינתזה

תוכנת סינתזה מקבלת את קבצי הקוד בשפת Verilog , ומבצעת את ההמרה משפת Verilog לבלוקים פרימיטיבים (שערים, פליפ-פלופים, זכרונות) הקיימים ברכיב ה FPGA. לאחר הסינתזה ניתן לראות את המעגלים החשמליים הממומשים לתכנון.
לביצוע סינתזה משתמשים בתוכנת Synplify Pro 7.2. על מנת להתחיל את העבודה עם התוכנה יש ליצור פרוייקט חדש:
File => New Project

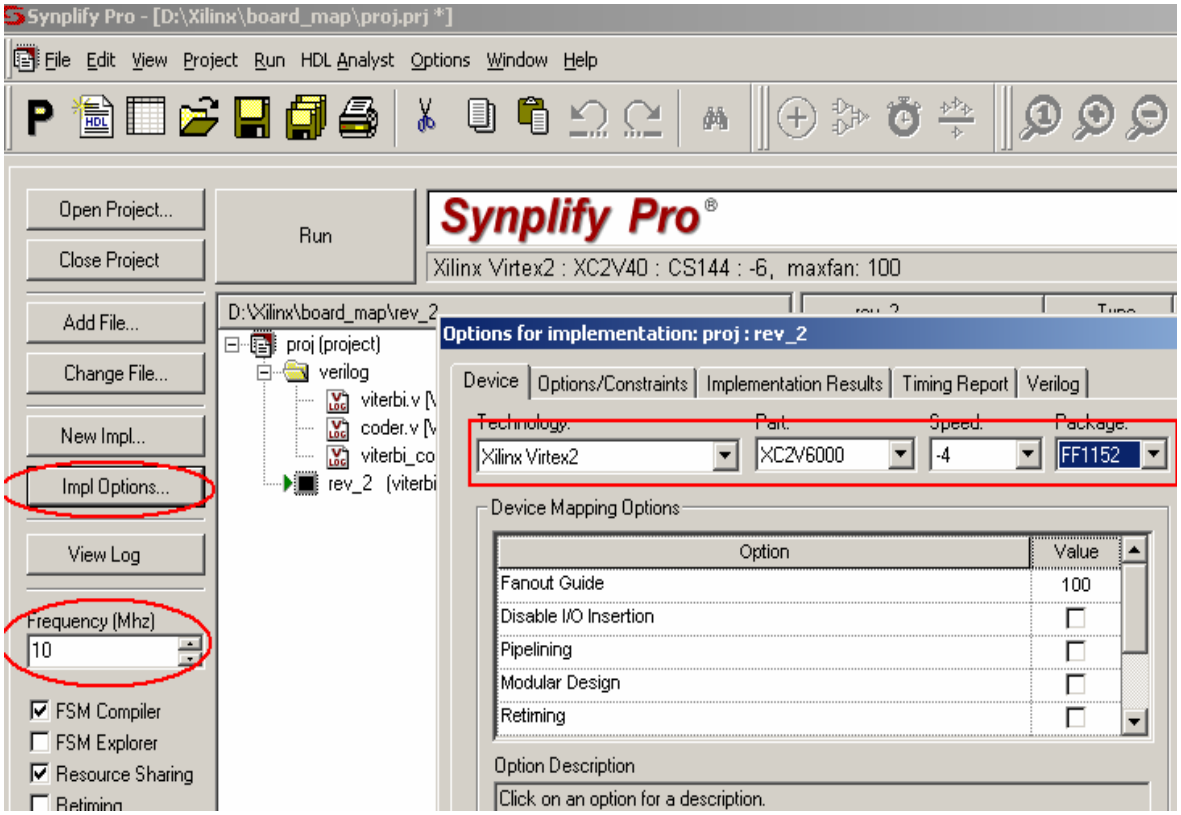


כעת יש להוסיף את קבצי הקוד לפרויקט:
Add File ולבחור את הקבצים ע''י לחיה על Add.



בתוך הפרויקט הקבצים של Verilog חייבים להיות מסודרים כך שה top module חייב להיות בסוף הרשימה.

כעת יש לבדוק ולעדכן את ההגדרות של הפרוייקט:

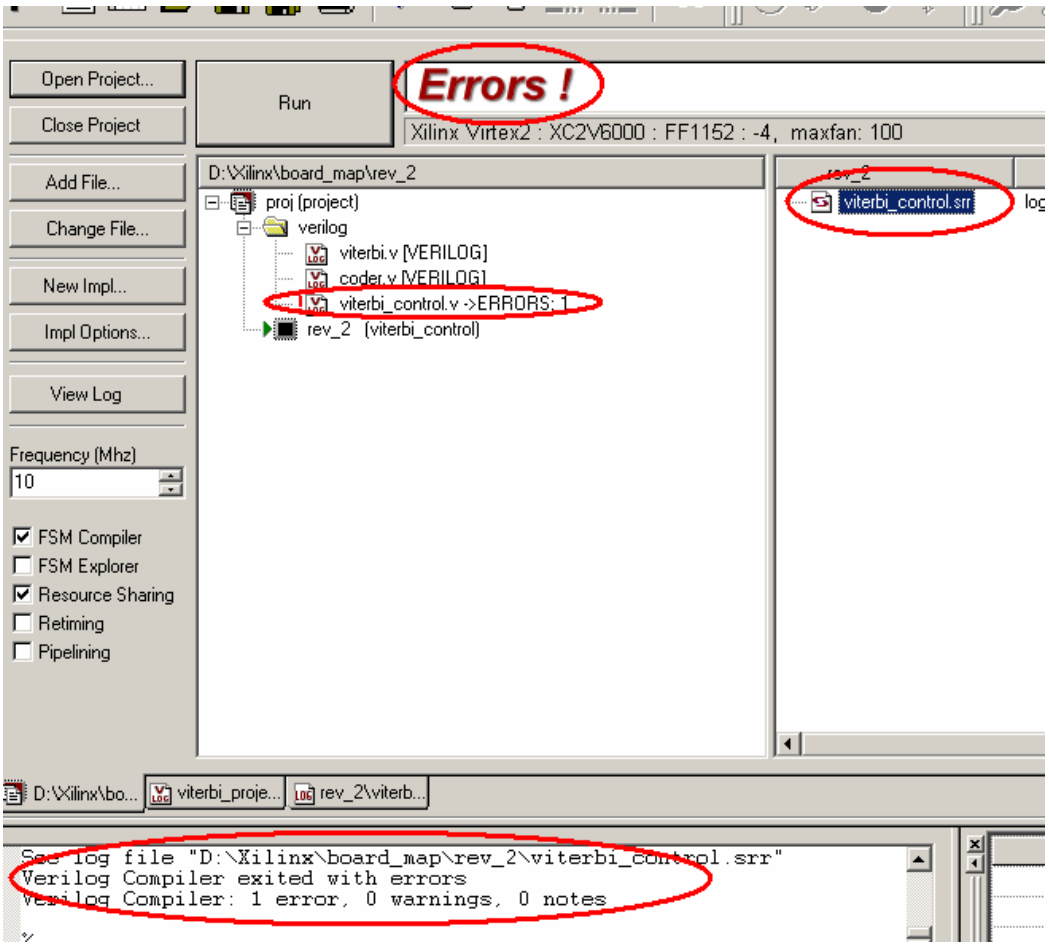


ב Implementation Options יש לבחור את דגם הרכיב אתו נעבוד כמתואר בציור. יש להכניס תדר עבודה למשל 10 MHz בדוגמה. הסינטסייזר יתן התראה במידה ולא יכול לייצר מימוש העומד בקצב הנ"ל.

להרצת הסינתזה יש ללחוץ על כפתור Run.

הסינתזה מבצעת אף היא קומפילציה של הקוד. ייתכן מצב שהקוד עבר את הסימולציה ב Modelsim אבל ייכשל בקומפילציה של Synplify. הדבר נובע מהעובדה שלא כל פקודות Verilog ניתנות לסינטזה וחלקן משמשות רק לסימולציה (קוד התנהגותי – Behavioral). יש להעזר בהוראות המתאימות בספרות לכתיבת קוד הניתן לסינטזה.

את השגיאות והערות ניתן לראות בחלון המצב בפינה השמאלית תחתונה של חלון ה Synplify :



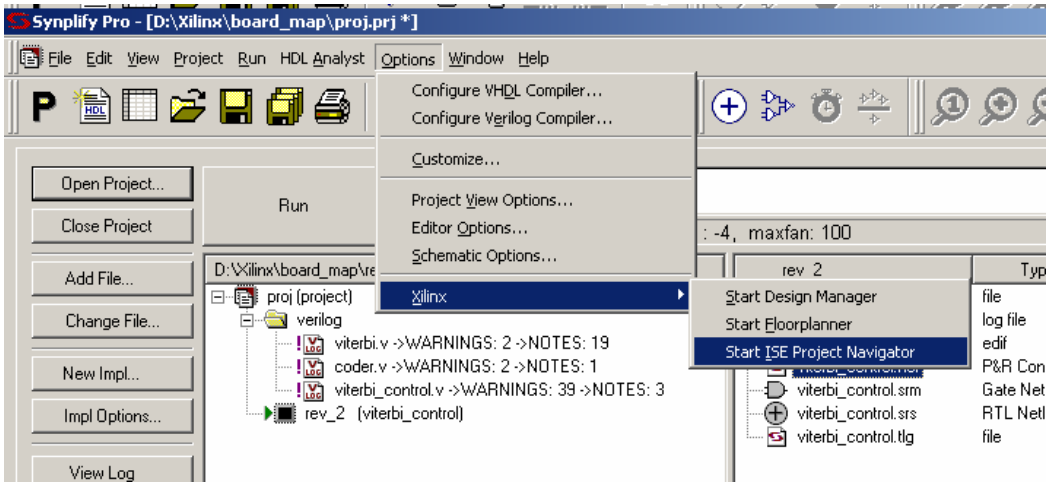
על מנת לקבל מידע על השגיאות יש לפתוח את קובץ *.srr המופיע בחלון הימני. אם תהליך הסתיים בהצלחה ניתן לעבור לשלב הבא – Place&Route.

שם לב:
מומלץ מאוד לעבור על ההערות warnings שהתוכנה נותנת ולנסות לתקן אותם.
בחלון הימני יפיעו שמונה קבצים. הקבצים האלה מכילים מידע על הפרוייקט. קבצים *.srm ו *.srs מכילים את ה netlist – כלומר תאור

סכמטי של החומרה. למעשה ל Place&Route אנו צריכים קובץ *.edf בלבד.

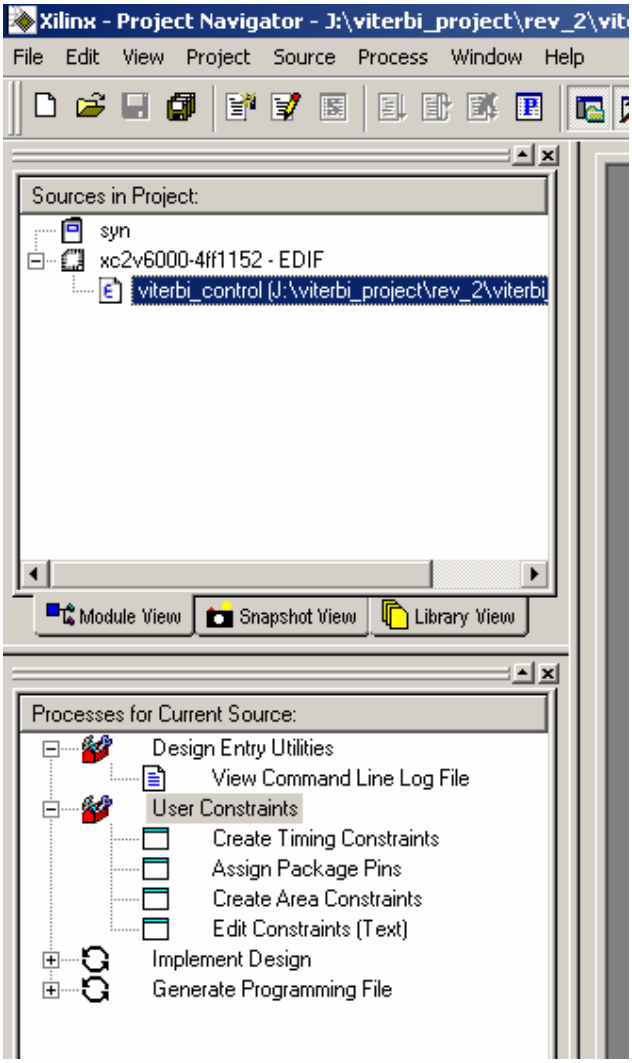
Place & Route

בשלב הבא תוכנה של חברת Xilinx (יצרנית ה FPGA) מבצעת מיקום של היחידות הפרימיטביות ברכיב וחיווט היחידות (Place & Route) על מנת להריץ את תהליך P&R מתוך Synplify יש לבחור את התפריט Options => Xilinx => Start Project Navigator



מיפוי

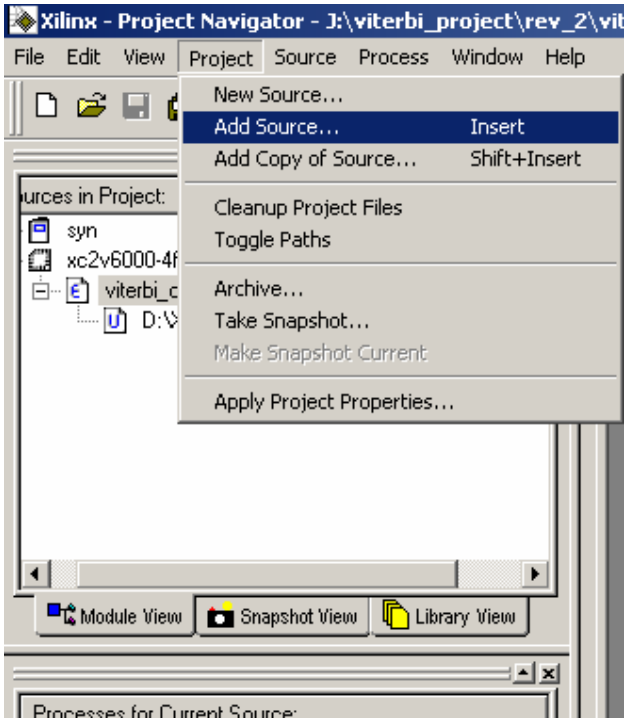
החלון הראשי של Xilinx Project Navigator נראה כך:



החלון העליון מכיל רשימה של קבצים של הפרוייקט. כרגע הוא מכיל רק קובץ אחד ה *.edf שקיבלנו מהסינתזה. החלון התחתון מכיל רשימת הפעולות שנדרש לבצע בתהליך ה P&R. על מנת לקבל קובץ מוכן לצריבה (קובץ הנטען לרכיב ה FPGA ומכיל את התכנון) עלינו לבצע את השלבים האלה אחד אחרי השני.

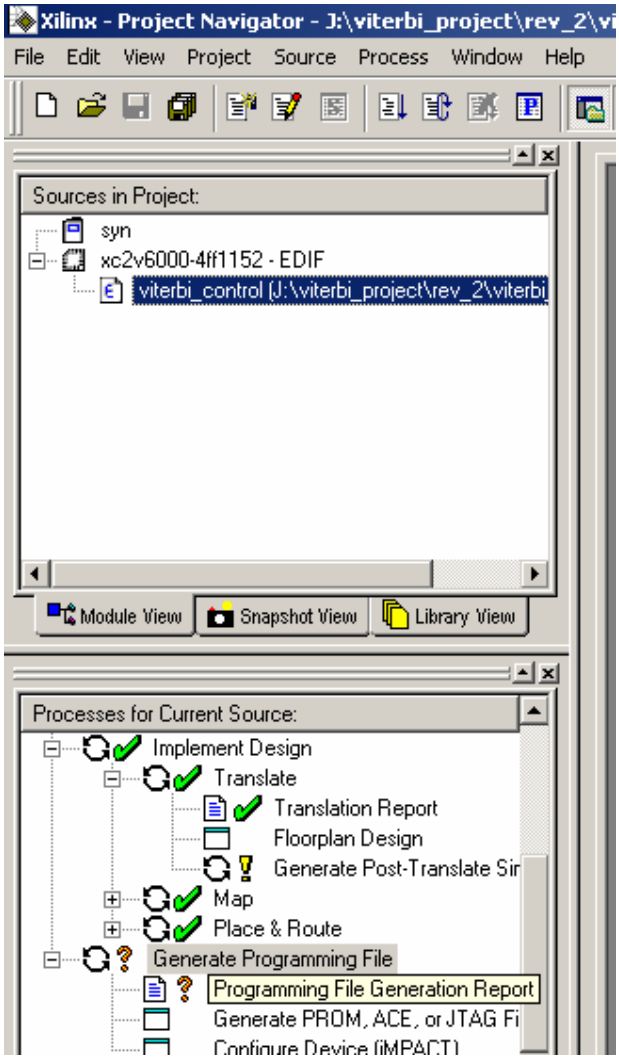
קודם כל עלינו להוסיף לפרוייקט קובץ אילוצים *.ucf . קובץ זה מכיל בין השאר את מיקום ההדקים הנדרשים לכל אות בכניסה ובמוצא של הרכיב. מיקום ההדקים ברכיב חשוב לקבלת תקשורת בין ה PC ל FPGA וכן לצרכי ניפוי ובדיקת התכנון בעזרת נתח לוגי כי שיוסבר בפרוט בפרק הבא:

על מנת להוסיף את קובץ האילוצים לפרוייקט יש לבחור את התפריט Project => Add Source...



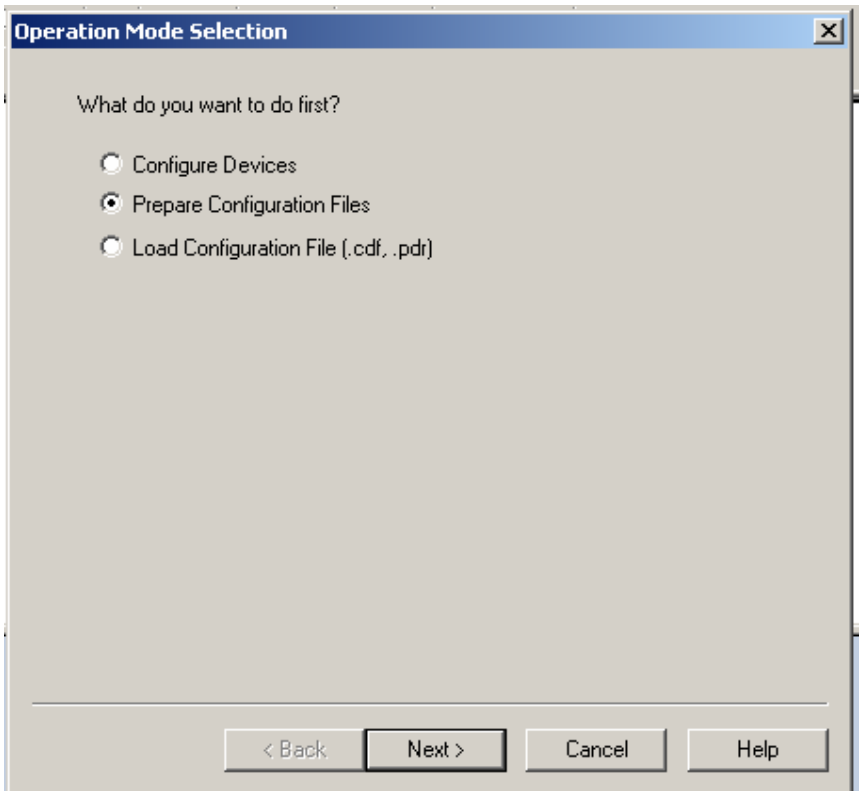
לאחר הוספת *.ucf אנו יכולים להמשיך בתהליך ע''י לחיצה על Process => Run.

בסיום התהליך (העזר בסבלנות, בתכנון גדול משך זמן הריצה יכול לקחת זמן) מתקבל חיווי ✔ בצבע ירוק על כל התהליכים עד וכולל Place & Route

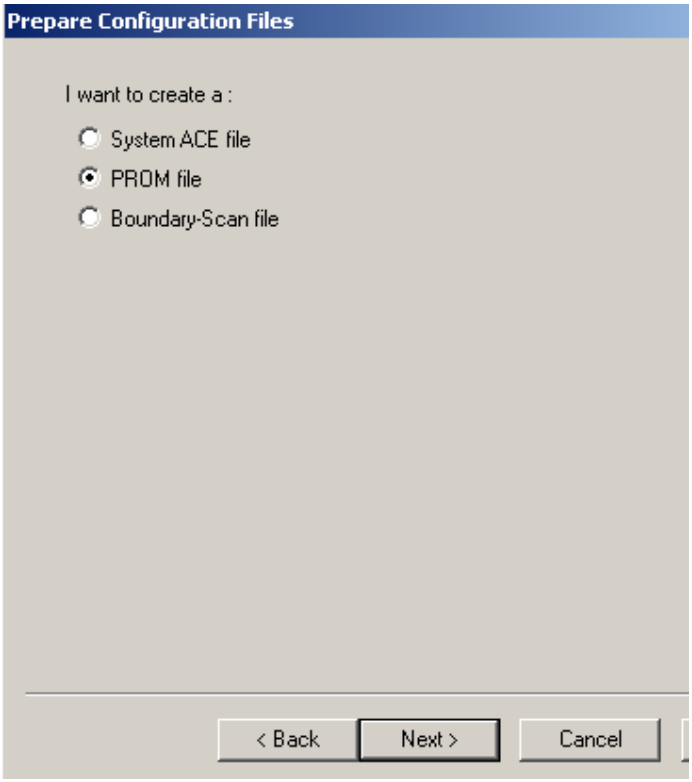


שלב Generate Programming File מייצר את קובץ נתוני הצריבה של הרכיב. יש לבחור את התהליך Generate PROM, ACE or JTAG File

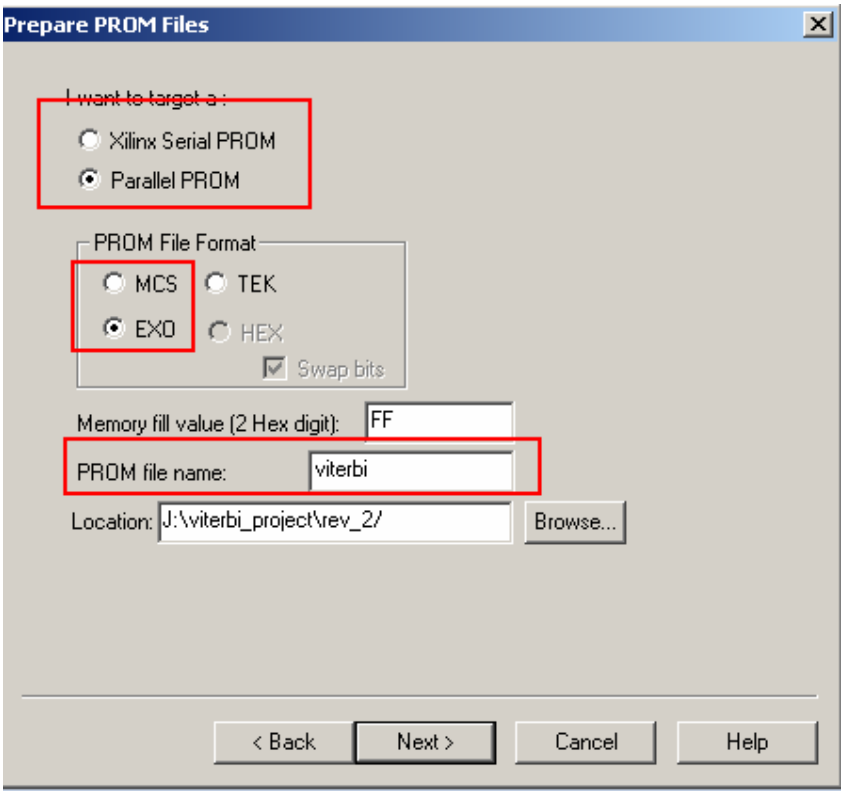
בחלון הנפתח יש לבחור באפשרות השניה: Prepare Configuration Files



בחלון הבא נבחר ב PROM File.

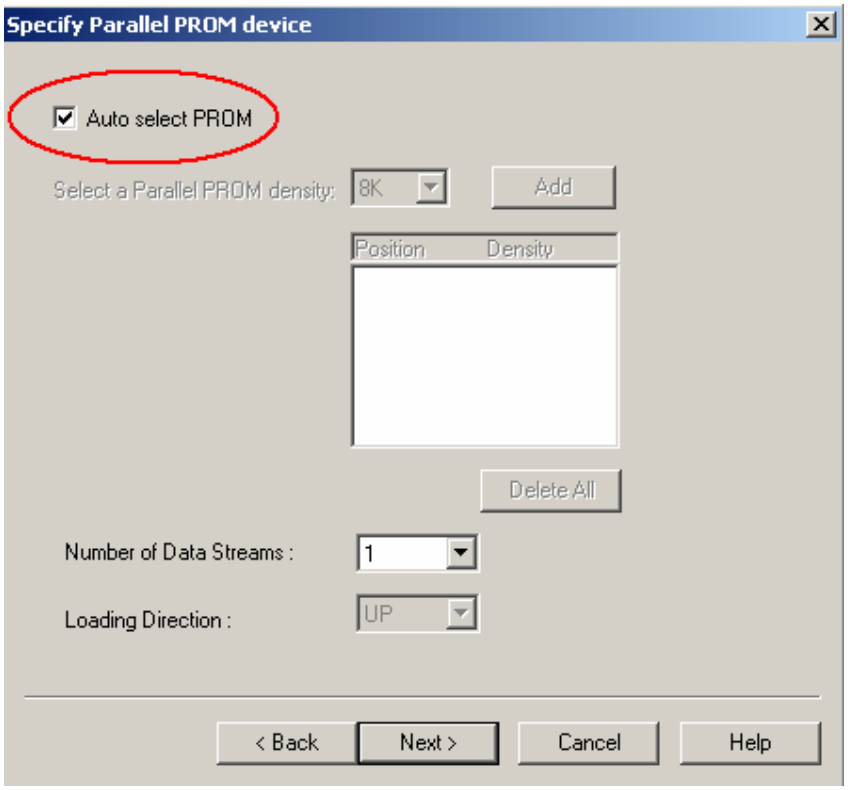


בחלון הבא יש לבחור את האפשריות לפי המתואר בציור שלהלן:



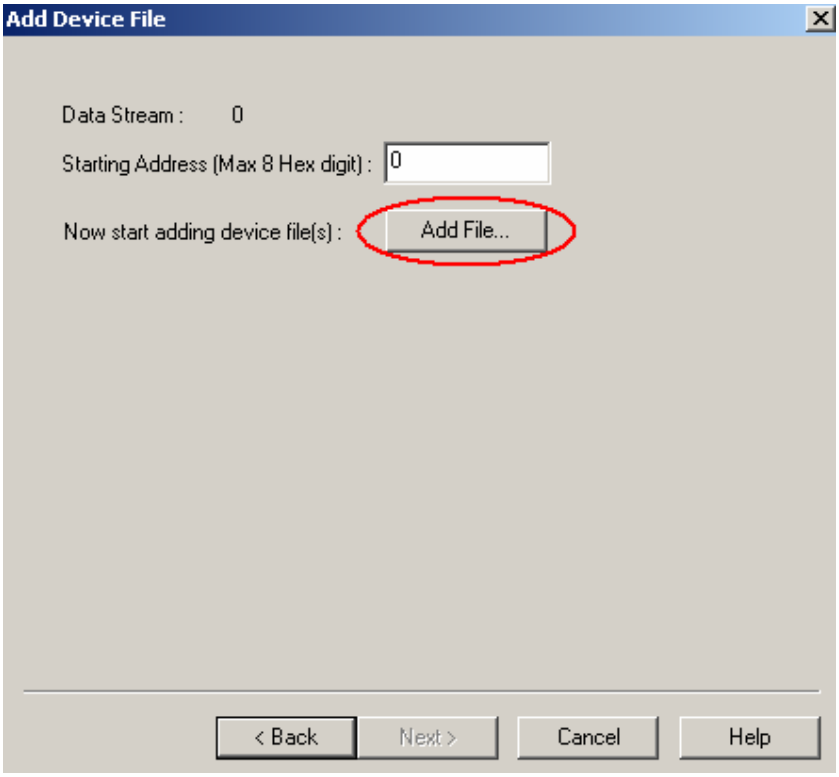
בחלק העליון יש לבחור ב Parallel PROM , ב PROM File Format יש לסמן EXO , ובחלק השלישי יש לבחור שם לקובץ הצריבה שייוצר (רצוי להשתמש בשם הפרויקט) וללחוץ Next.

בחלון הבא Specify Parallel PROM Device יש לבחור ב Auto Select:

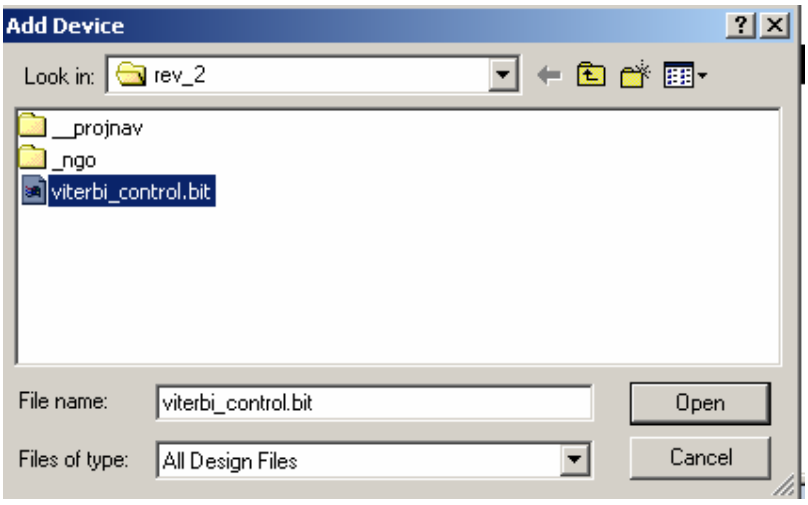


יש ללחוץ פעמיים על Next

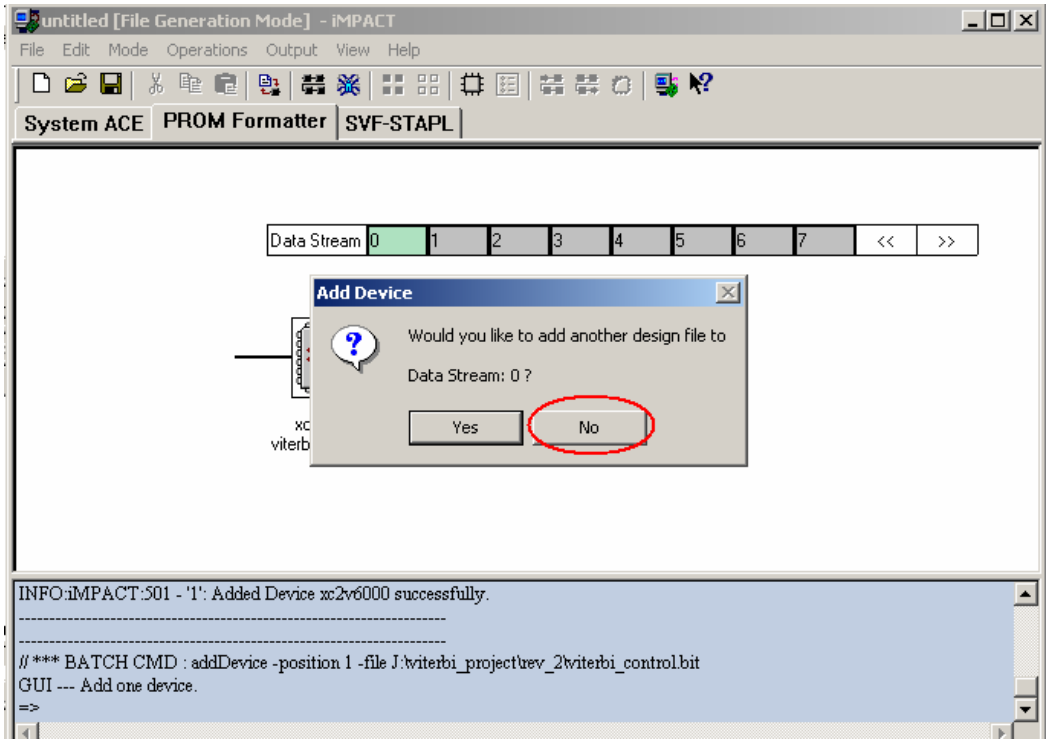
ואחר כך ללחוץ על הכפתור Add File



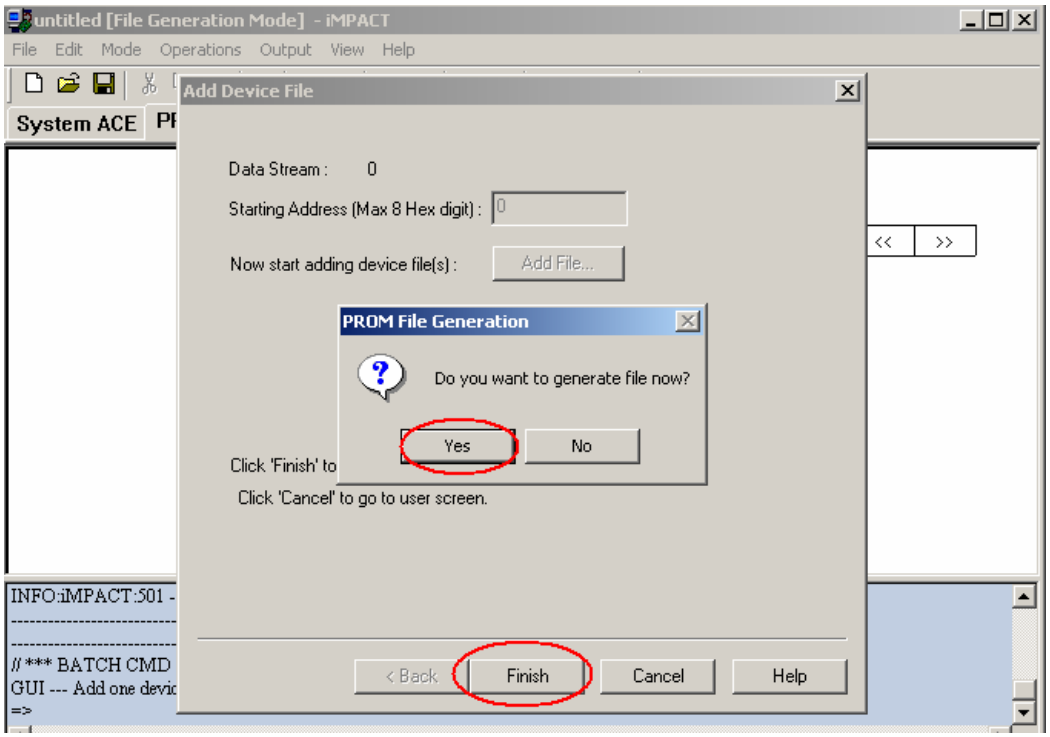
לאחר מכן יש לסמן את קובץ בפרוייקט שממנו יווצר ייוצר קובץ הצריבה (*.exo . הקובץ שנוצר ע''י Place&Route הינו הנו קובץ הקלט ליצירת קובץ הצריבה ובעל הסיומת *.bit ולכן יש לבחור את קובץ *.bit המתאים בחלון הנ''ל.



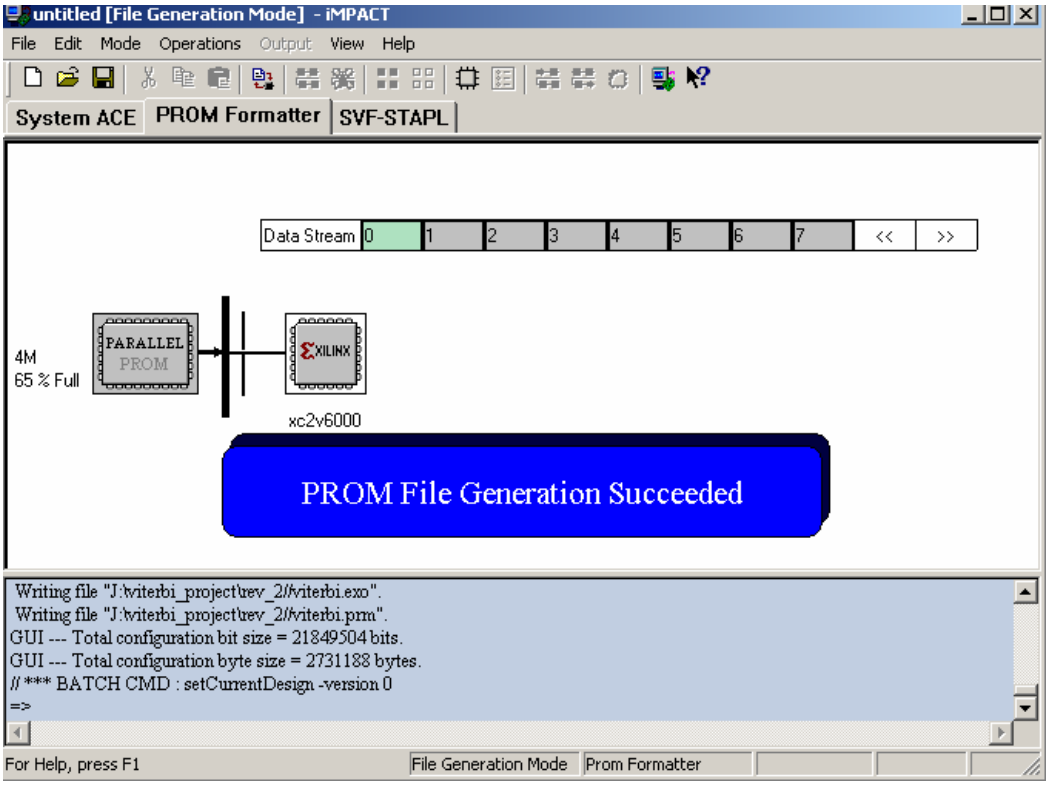
בחלון הבא המוצג להלן יש לסמן "No"



לעשות Finish בחלון הבא ו Yes בחלון PROM File Generation

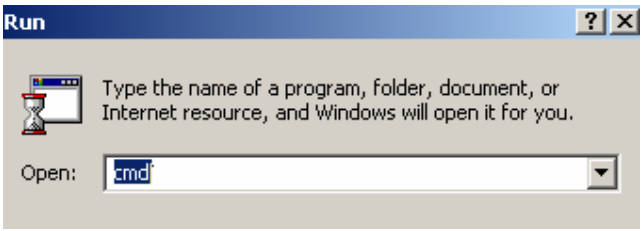


אם עברנו בהצלחה את כל התהליך הנ"ל נקבל את החלון הבא:



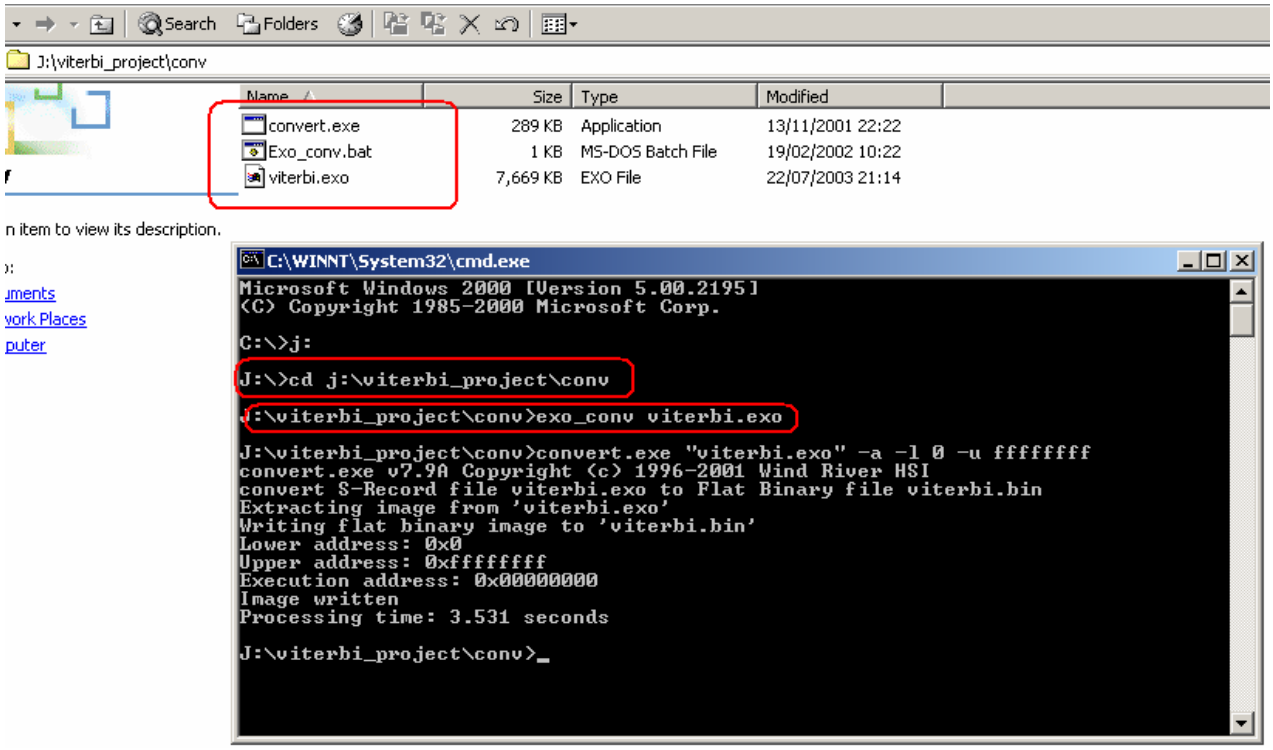
כעת סיימנו את ייצור קובץ הצריבה וניתן לסגור את כל החלונות של Project Navigator.

שלב זה נדרש לבצע על המחשב המחובר לכרטיס.
יש לתאם שעות העבודה על המחשב הנ"ל עם המדריך במעבדה.
קובץ exo.* שנוצר מכיל את כל מידע לצריבה. אין צורך להעתיק אתו על המחשב הנ"ל, מספיק ליצור מחיצה על הדיסק ברשת (J:) ולשם לשים את ה exo.*
התוכנה בכרטיס צורבת את ה FPGA בעזרת פורמט שונה במעט ולכן כעת עלינו להמיר לפורמט המתאים לכרטיס. נעשה זאת בעזרת תוכנית המצורפת לכרטיס בשם Exo_conv.bat . יש להעתיק אותה ממחשב Comlab52 למחיצה שלכם ברשת.
התוכנית נמצאת ב D:\runcom_board
על מנת להשתמש ב exo converter יש לשים את 2 קבצים – convert.exe ו exo_conv.bat לתיקיה אחת עם קובץ exo.* שאותו רוצים להמיר. יש להכנס ל Ms-Dos ע"י Start => Run ולכתוב cmd כדי לעבור ל סביבת DOS:

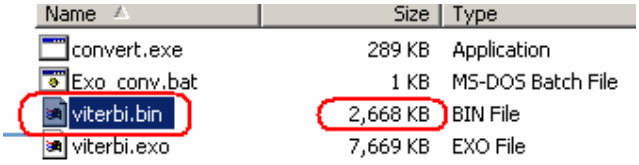


לעבור לתיקיה ע''י cd drive:\dir name

ולהריץ את converter ע''י exo_conv *.exo



כתוצאה נוצר קובץ *.bin :

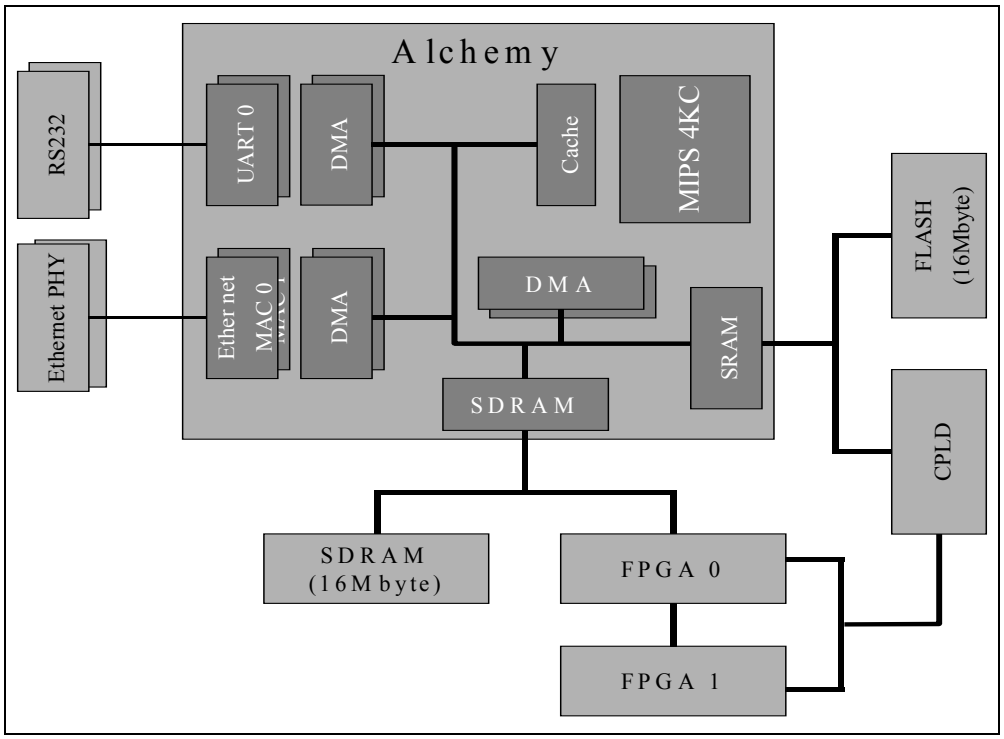


יש לשים לב שגודל הקובץ איננו משתנה וצריך להיות בגודל של 2.668 Mb !!!

4. טעינה של התכנון לכרטיס

כרטיס FPGA

כרטיס ה FPGA הנו כרטיס שנבנה על ידי חברת RUNCOM ומכיל 2 FPGA של חברת Xilinx.
רכיבי FPGA הינם רכיבים המכילים לוגיקה הניתנת לתכנות. הרכיבים הנמצאים בכרטיס הנם XC2V6000 המכילים לוגיקה בסדר גודל של כ 6 מיליון שערים לוגיים מחולקים למערך של 88x96 בלוקים למימוש לוגיקה כללית, 144 בלוקים של מכפלים ו 144 בלוקים של 18Kbit RAM . גודל רכיבי ה FPGA גדלים מעת לעת. בעת רכישת הכרטיס, רכיבי ה FPGA הנ"ל היו מהגדולים בשוק. בציור הבא נתונה סכימת המלבנים של הכרטיס.



FPGA Board General View

כפי שניתן לראות בציור הכרטיס בנוי סביב מעבד RISC בשם "Alchemy" המכיל בתוכו ליבה (Core) של מעבד מסוג MIPS.

לאחר ביצוע Reset הכרטיס מריץ תוכנת monitor הנמצאת בזכרון הלא נדיף (Flash) . תוכנת המוניטור מתקשרת עם הסביבה בעזרת פרוטוקול RS232 וסט פקודות מינימלי המאפשר:

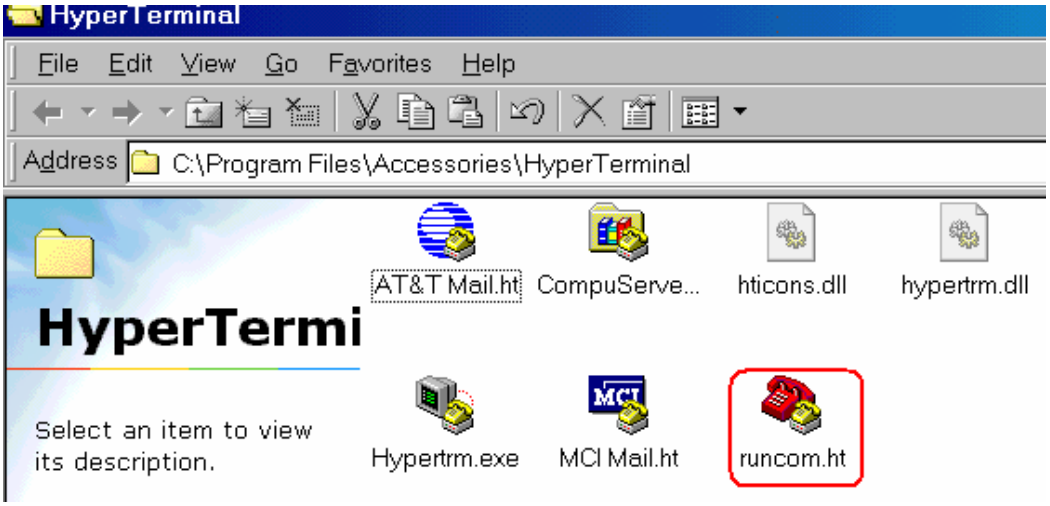
1. **טעינה של קובץ צריבת ה FPGA לזכרון הפלאש 2**
2. **צריבת ה- FPGA מזכרון הפלאש.**
3. **כתיבה וקריאה של נתונים מזכרון (רגיסטרים) הממוקמים בתוך ה FPGA.**

<u>הערה חשובה:</u> יש לזכור כי כל פעם שאין אספקת מתח לכרטיס הצריבה של ה FPGA נמחקת אך קובץ הצריבה בפלאש עדיין שמור בכרטיס.

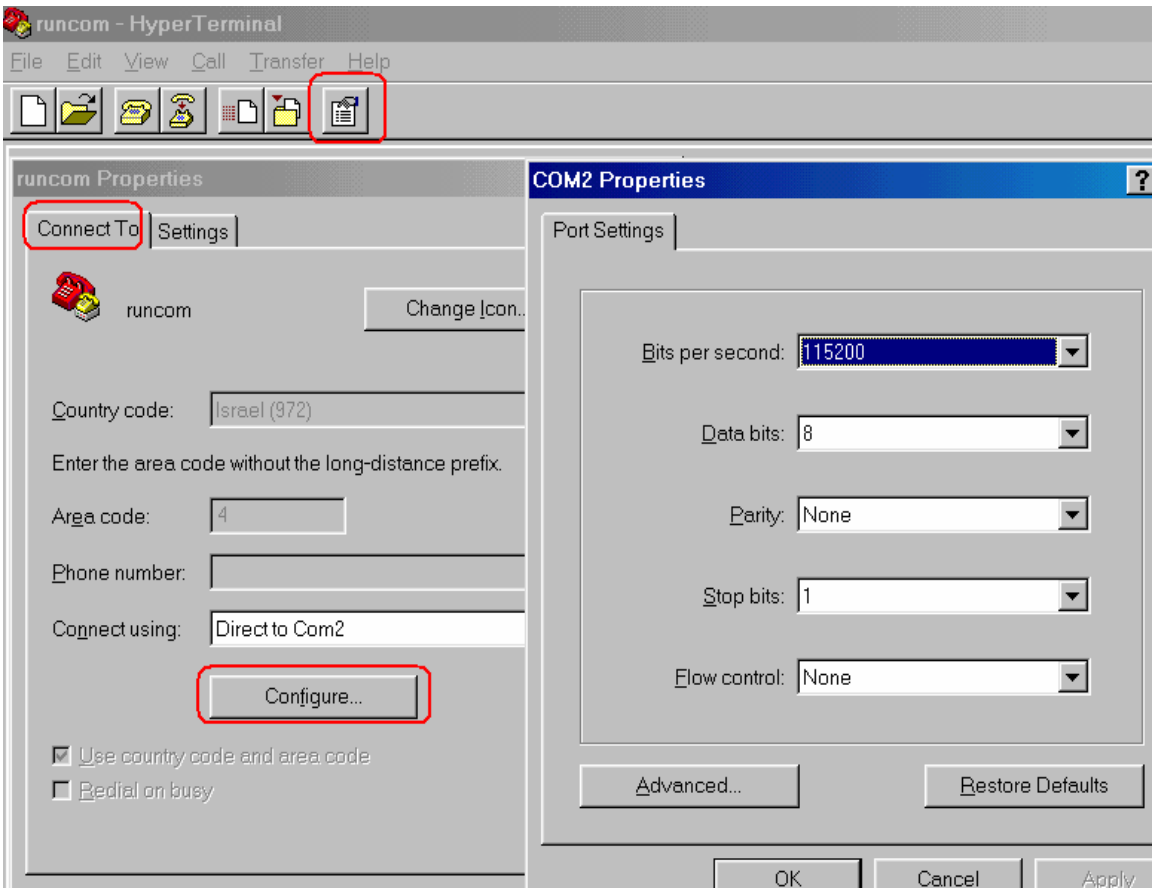
במידה ותכנון ה FPGA כולל את המודול cpu_interface ניתן בעזרת המוניטור לבצע פעולות של כתיבה וקריאה מרגיסטרים ב FPGA כאשר פקודת טקסט ב PC נשלחת בתקשורת הטורית למוניטור, תוכנת המוניטור יוזמת פקודת קריאה או כתיבה על ה BUS של המעבד, ואילו המודול Cpu interface ב FPGA מזהה את הטרנזקציה על BUS וממיר אותה למחזור קריאה או כתיבה סטנדרטי לשאר המודולים הממומשים בתוך FPGA.

טעינה של קובץ צריבת ה FPGA

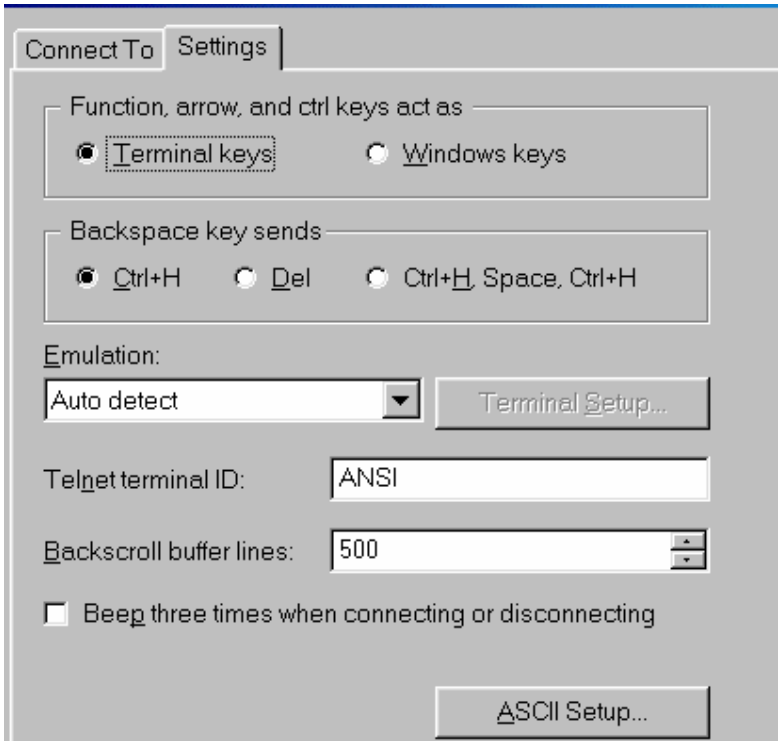
תהליך הטעינה הוא השלב האחרון בפרוייקט.
התהליך נעשה בעזרת תוכנת HyperTerminal של Windows , אשר מתקשרת בממשק טורי לתוכנה הרצה בכרטיס.
כדי להפעיל את התוכנה יש להכנס ל:
Start >> Accessories >> Communications >> HyperTerminal
ושם להפעיל את runcom.ht:



לפני השימוש בתוכנה יש לבדוק את ה Properties ולוודא שמאפייני החיבור מוגדרים נכון. יש ללחוץ על הסמל שמופיע בצד שמאל של החלון – יד עם דף, ובחלון הנפתח לבחור בלחצן Configure. יש להשוות את ההגדרות המופיעות שם עם הגדרות שבציור.



לאחר מכן עוברים ל Settings ובודקים גם שם:



אם כל ההגדרות נכונות תוכנת HyperTerminal תציג את התפריט הראשי של ממשק הכרטיס הנראת כך:

```
********************* The learning (1) version **

                  FPGA tests

Type  Mode: s - single request,
            f - i/o file/complex record,
            h - Help,
            c - change system clock:
```

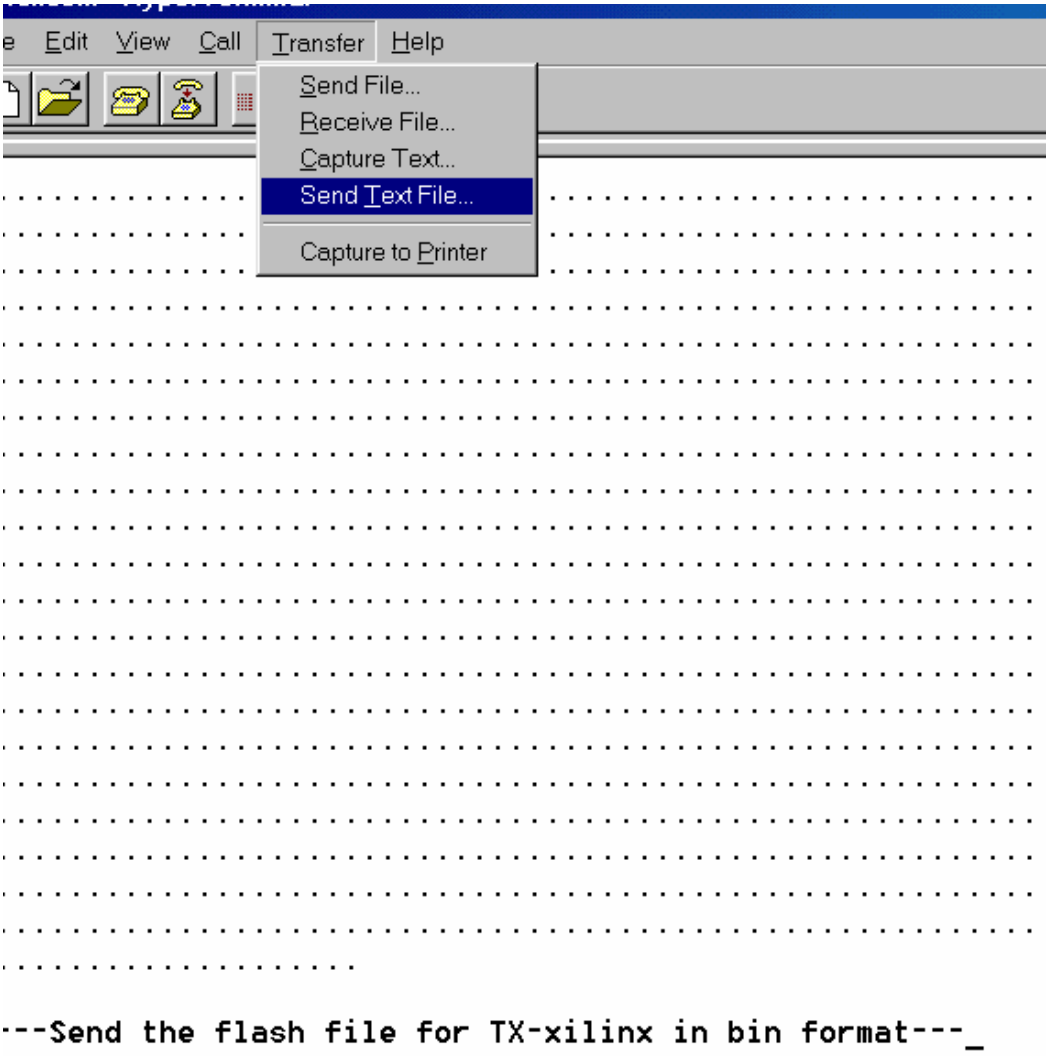
ברצוננו לטעון תכנון לזכרון הפלש של הכרטיס, לכן יש לבחר באפשרות השניה – file/complex record i/o כלומר יש ללחוץ על "f".
בתפריט הבא ברצוננו לשלוח קוד לרכיב.

בחר באפשרות השניה – "l" - Load code to FPGA/Flash

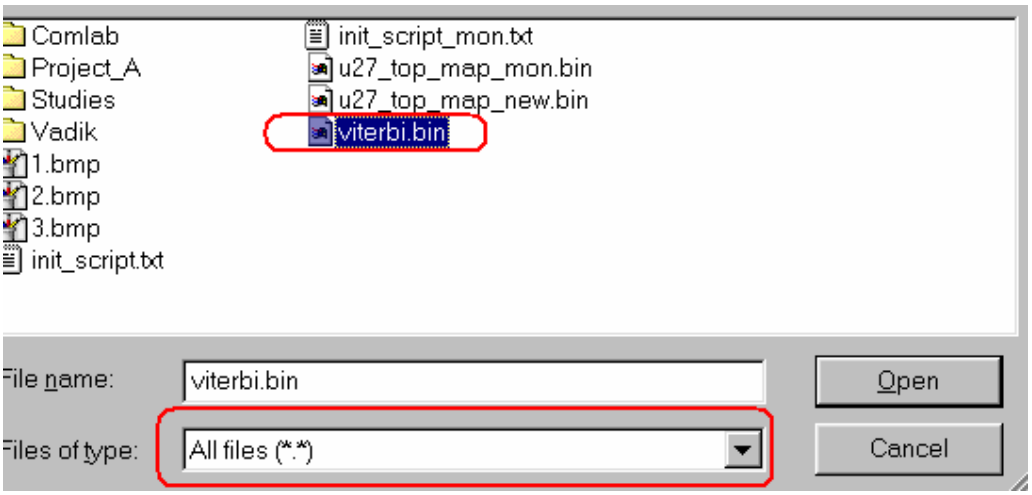
ובתפריט השלישי נבחר לשלוח קובץ ממחשב ... program FLASH m –

```
Type Mode: s - single request,
           f - i/o file/complex record,
           h - Help,
           c - change system clock: f

         Type: c - test from file,
               1 - load code to FPGA/Flash
               '-' - back: 1

      Type: f - load FPGA from Flash,
            m - program FLASH from PC : m

File for FPGA 1 - 1,
          FPGA 2 - 2,
          Exit - ESC: 1

Please,wait a minute.....................................
.........................................................
```

כאשר תתקבל ההודעה: Send the flash file יש להיכנס לתפריט transfer:



לבחור ב Send text file . בחלון הנפתח יש לשנות את סוג הקבצים
ל all files ולסמן את קובץ *.bin.



וללחוץ open.
כאשר מתבצעת הטעינה בשורת מצב בתחתית העמוד נוספות נקודות.

```
---Send the flash file for TX-xilinx in bin format--....
```

כאשר הטעינה מסתיימת התכנון טעון לתוך זכרון הפלש בכרטיס.
כעת ברצוננו לטעון את התכנון מהפלש לתוך ה FPGA.
חוזרים על אותו תהליך אך בוחרים ב

F - load FPGA from FLASH

```
Type Mode: s - single request,
           f - i/o file/complex record,
           h - Help,
           c - change system clock: f

         Type: c - test from file,
               l - load code to FPGA/Flash
               '-' - back: l

       Type: f - load FPGA from Flash,
             m - program FLASH from PC: f

File for FPGA 1 - 1,
          FPGA 2 - 2,
          Exit - ESC: 1

Please,wait a minute.....................................
.........................................................
```

5. בדיקת התכנון

ניתן לבדוק את התכנון בשני דרכים:

א. דרך פרוטוקול התקשורת ניתן לכתוב ולקרא רגיסטרים שמומשו בתכנון. לדוגמה עם בכתובת מסוימת ניתן הקלט לתכנון ובכתובת אחרת נמצא הפלט. ניתן לכתוב ולקרא מהמחשב ולבדוק האם זוג הנתונים ממש את הנדרש.

ב. ניתן להוציא אותות מהתכנון לפינים ב FPGA המחוברים למחבר מיוחד המכונה מיקטור. מחבר זה מיועד להתחבר ללוג'יק אנלייזר המאפשר לראות את צורות הגלים של האותות הנ''ל. על סמך ניתוח של צורות הגלים ניתן לבחון האם התכנון מבצע את הנדרש.

את המיפוי בין האותות למיקטור ניתן לבצע על ידי מיפוי אות להדק מסוים של ה FPGA בקובץ אילוצי התכנון ucf.* כאשר המיפוי בין הדק ב FPGA להדק במחבר המיקטור ניתן בשרטוטי חיווט הכרטיס. פנה לאחראי המעבדה לקבלת מידע מפורט בנושא.