**המעבדה ל - VLSI**

# חוברת הדרכה על הרצת סימולציות בעזרת NCSIM

**גואל סמואל**
**אמנון סטניסלבסקי**

## הקדמה

**התחברות ממחשבי מרחוק ראה את אתר המעבדה:**
https://vlsi.eelabs.technion.ac.il/

## הקדמה

הכלי xmsim הוא סימולטור מבית Cadence אשר בעזרתו מבצעים סימולציות על קבצי verilog,systemverilog,systemc ו- VHDL. רצוי להריץ את הכלי רק על מחשבי ה-linux החדשים של המעבדה.

**xmhelp** : פקודה זאת מספקת הסברים נוספים של שגיאות שמתקבלות מהכלים השונים.
דוגמא :

```
xmhelp xmvlog CUVWSP
```

**xmls** : פקודה זאת מציגה רשימת המודולים שנמצאים בספריות המקומפלות. לדוגמא :

```
xmls -library worklib
xmls module_name
```

שימוש באופציה –command מראה גם את פקודת הקומפילציה של המודול.

xmrm : מוחקת מודול מהספרייה.

**חשוב : ניתן להפעיל את כל סביבת הסימולציה באמצעות הכלי nclaunch. חוברת הסבר מופיע באתר המעבדה.**

### הכנות מקדימות

**חשוב** : יש לעדכן את קובץ ה-cshrc. פתיחת הקובץ לעדכון ע"י הפקודה :

```
nedit ~/.cshrc &
```

יש לוודא שהשורה הבא מופיעה בקבוץ ואם לא, יש להוסיף אותה :

```
source /users/iit/cadence/source_cadence2324
```

ראשית, מומלץ ליצור ספרייה חדשה אך ורק לצורך סימולציה. בתוכה יש ליצור מקום שהסימולאטור יקמפל לתוכו את הקבצים. לשם כך ניצור עוד ספריה חדשה (בתוך הספרייה הראשונה שייצרנו). לדוגמא ייצרנו את הספרייה project/~ על מנת שנריץ מתוכה סימולציות. כעת נעבור לספריה זאת :

```
cd ~/project
```

העתק קובץ בשם cds.lib בעזרת הפקודה הבאה :

```
cp /users/iit/cadence/xmsim2324/cds.lib .
```

כעת העתק קובץ בשם hdl.var בעזרת הפקודה :

```
cp /users/iit/cadence/xmsim2324/hdl.var .
```

צור את הספרייה worklib בעזרת :

```
mkdir worklib
```

הקובץ בשם cds.lib המכיל את השורות הבאות :

```
include …/tools/inca/files/cds.lib
DEFINE worklib . /worklib
```

השורה הראשונה מייבאת מספר הגדרות של הכלי ואילו השורה השנייה מודיעה לכלי שהספרייה ששמה worklib נמצאת במקום ~/simulation/worklib

הקובץ בשם hdl.var מכיל :

```
DEFINE WORK worklib
```

כדי לאפשר עבודה גם עם קבצי VHDL עם אותו worklib שהוגדר בקובץ cds.lib עבור verilog.

**השלבים לביצוע סימולציה:**

**א. יצירת קובץ המכיל את המעגל**

רשום קובץ Verilog בעזרת ה- texteditor (xemacs למשל) המכיל את התכנון בשם design.v למשל.

```
module design (input a,b,c, output logic s1,c1);
 always @(a,b,c)
  begin
   s1 = a ^ b ;
   c1 = s1 ^ c ;
  end
 endmodule
```

**ב. יצירת קובץ Verilog המכיל את אותות הכניסה למעגל**

פתח את הקובץ design_test.v. קובץ זה משמש כ- testbench ומטרתו היא לדחוף את אותות הכניסה להלן דוגמא של מבנה הקובץ :

```
module design_test;
  logic a, b, c;
  logic s1, c1;

  initial begin
   $monitor("a=%b, b=%b, c=%b, c1=%b, time=%t",a,b,c,c1,$time);

   {a,b,c} = 3'b000;
   #10 {a,b,c} = 3'b001;
   #10 {a,b,c} = 3'b011;

   #10 $finish;
  end

  design U1 (.*);
endmodule
```

## ג. ביצוע קומפילציה

עבור קוד verilog יש לקמפל את כל הקבצים בעזרת הפקודות :

```
xmvlog -sv -nowarn DLCPTH -nocopyright -work worklib design.v
xmvlog -sv -nowarn DLCPTH -nocopyright -work worklib design_test.v
```

אופציות :

- sv– : עבור קוד systemverilog
- work worklib- : אם worklib לא מוגדר ב- cdslib יש להוסיף את ההגדרה בפקודה
- linedebug- : לביצוע debug
- update- : למניעת קומפילציה של מודולים מעודכנים.

עבור קוד VHDL יש להשתמש בפקודה :

```
xmvhdl design.vhd
```

תוצאות הקומפילציה נשמרת בספרית העבודה (למשל worklib).

## ד. ביצוע אלבורציה - elaboration

לאחר קומפילציה יש לבצע elaboration של התכנון כלומר קישור המודולים וביצוע preprocessing (החלפת ה- generics בערכיהם ועוד).

אם הקובץ העליון ביותר (top level) הוא קובץ verilog :

```
xmelab -access +rwc -work worklib  design_test
```

כאשר design_test הוא שם המודול העליון.

אם הקובץ העליון ביותר (top level) הוא קובץ VHDL :

```
xmelab -access +wrc TOP_LEVEL_NAME:ARCHITECTURE_NAME
```

כאשר TOP_LEVEL_NAME הוא שם ה- entity של ה- top level ו- ARCHITECTURE_NAME הוא שם הארכיטקטורה הרצויה שלו.

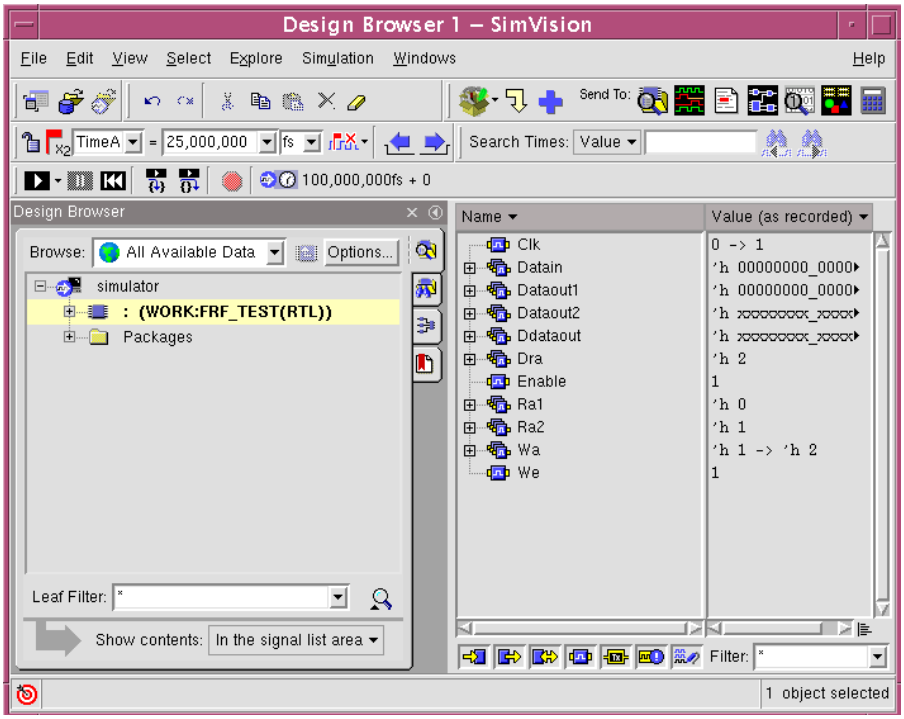
אופציות :

- work mylib– : אם worklib לא מוגדר ב- cdslib יש להוסיף את ההגדרה בפקודה
- access +wrc- : מאפשר גישה מלאה לאובייקטים בזמן debug

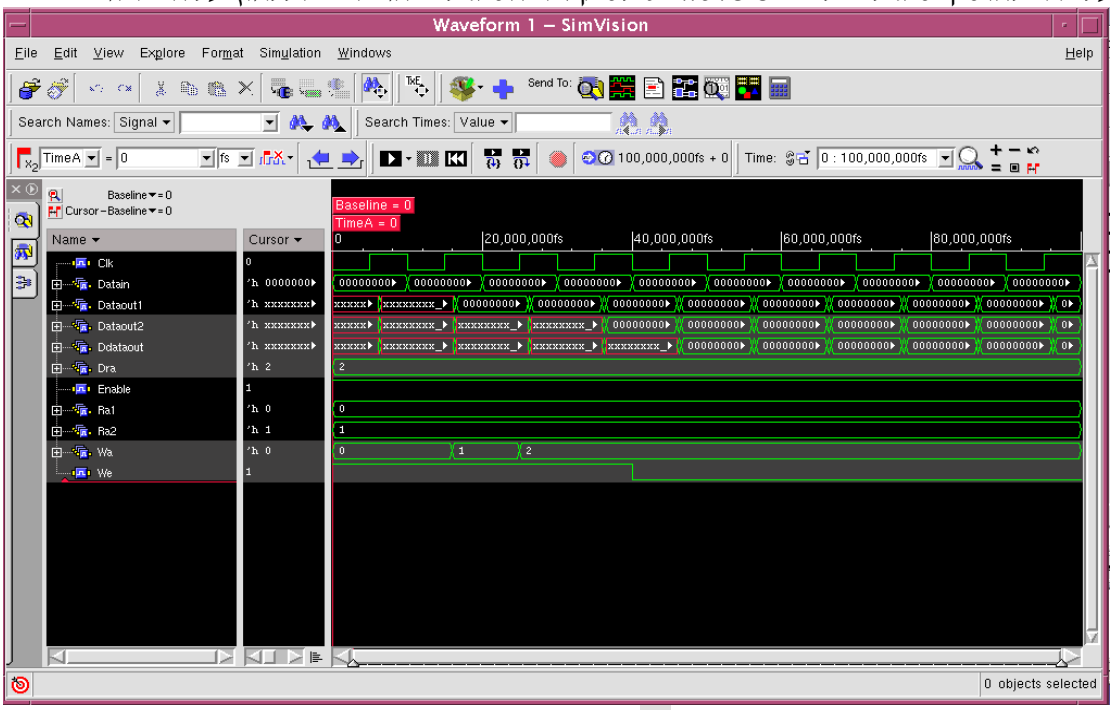
## ה. תפעול הסימולאטור

```
xmsim -gui design_test &
```

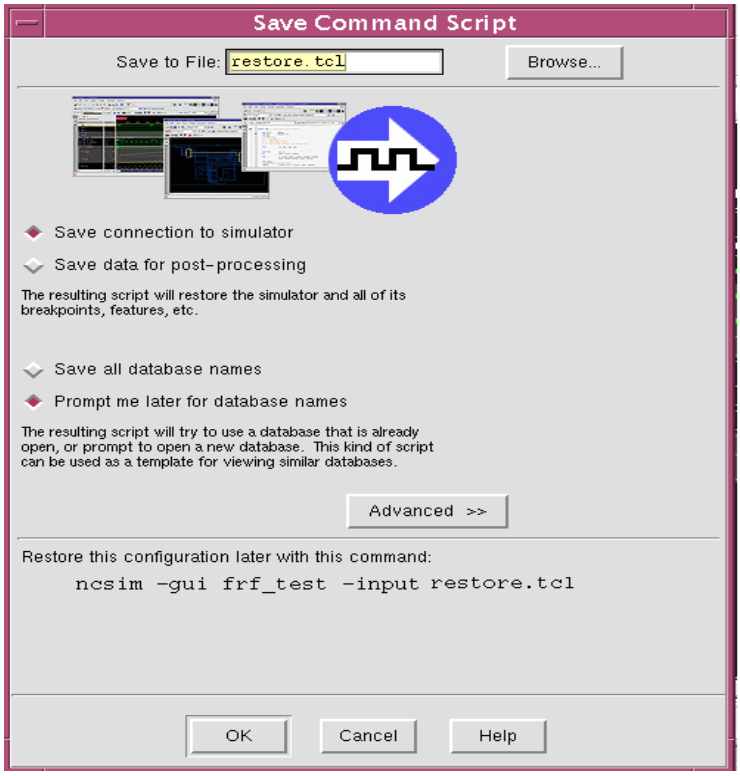
כאשר design_test הוא שם entity/module של ה- top level. יפתח המסך הבא :




על מנת להוסיף סיגנלים ל- waveform יש לסמן את הסיגנלים הרצויים וללחוץ על הכפתור :




- להרצת הסימולציה ניתן ללחוץ על ▶ או לרשום בחלון הראשי run TIME (לדוגמא run 50 ns יריץ את הסימולציה 50 ננו-שניות)
- על מנת להגיע לשינוי הבא בסיגנל יש לסמנו וללחוץ על
- על מנת לשמור תמונה של ה- waveform יש ללחוץ על file->print window…
- על מנת לשמור את מצב ה- wavform יש ללחוץ file->save source
-




ובפעם הבאה אם נריץ את הפקודה xmsim -gui TOP_LEVEL -input restore.tcl - יפתח ה-waveform כפי שהוא נראה בזמן השמירה (מכיל את הסיגנלים הרצויים ובזום הרצוי)

### ה. הפעלת התהליך בפקודה אחת : xrun

ניתן לבצע את כל הפקודות הנ"ל בעזרת פקודה אחד בלבד :

```
xrun -gui -sv -access +wrc design.v design_test.v
```

הפקודה xrun לא מסוגלת לזהות את ה- top level entity בעבודה עם קבצי VHDL ולכן לא ניתן להפעיל את התהליך בפקודה אחת במקרה של שימוש בקבצי VHDL.

### שימוש ב- package ב- VHDL :

אם יש לנו מספר רב של קבצי VHDL – ניתן לפצל אותם למספר ספריות. במקרה שכזה – ניצור ספרייה עם השם הרצוי ונוסיף משפט מתאים בקובץ ה- cds.lib.

לדוגמא : ייצרנו קובץ בשם mypack.vhd שבו מוגדר package וברצוננו לשים את הקובץ המקומפל ב- pkglib אז נקפל בעזרת הפקודה :

```
xmvhdl -work pkglib mypack.vhd
```

בקובץ cds.lib יש להגדיר את מיקום הספריה בעזרת הפקודה :

```
DEFINE pkglib ./PKG_LIB
```

כמובן, אם PKG_LIB לא קיים, יש לצור אותו עם :

```
mkdir ./PKG_LIB
```

אם נניח שקובץ mypack.vhd המכיל את השורה הבאה :

```
package mypkg is …
```

כל קובץ שירצה להשתמש ב- package מהספרייה הזו יוסיף את הפקודה library NAME כאשר NAME היא השם שנתנו לספרייה בקובץ ה- cds.lib. כל entity שירצה להשתמש ב- package זה יוסיף את שתי השורות הבאות לפני הגדרתו :

```
library PKG_LIB;
Use PKG_LIB.mypkg.all;
```

## שימוש ב- package ב- SystemVerilog :

אם יש לנו מספר רב של קבצי SystemVerilog – ניתן לפצל אותם למספר ספריות. במקרה שכזה – ניצור ספרייה עם השם הרצוי ונוסיף משפט מתאים בקובץ ה- cds.lib.
לדוגמא : ייצרנו קובץ בשם mypack.sv שבו מוגדר package וברצוננו לשים את הקובץ המקומפל ב- pkglib אז נקפל בעזרת הפקודה :

```
xmvlog -work pkglib mypack.sv
```

בקובץ cds.lib יש להגדיר את מיקום הספריה בעזרת הפקודה :

```
DEFINE pkglib ./PKG_LIB
```

כמובן, אם PKG_LIB לא קיים, יש לצור אותו עם :

```
mkdir ./PKG_LIB
```

אם נניח שהקובץ mypack.sv מכיל את השורה הבאה :

```
package PkgName;
…
```

כל module שירצה להשתמש ב- package זה יוסיף את השורה הבאה אחרי משפט ה- module :

```
import PkgName::*;
```