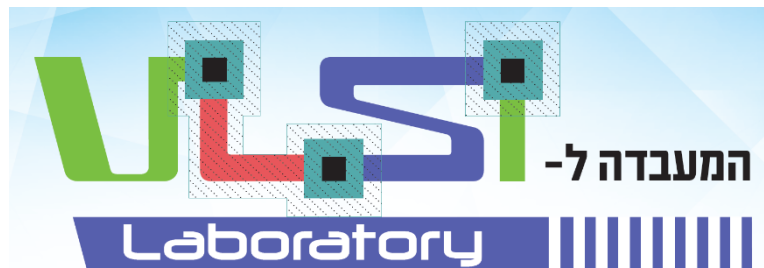




# המעבדה ל – VLSI



מעבדה 2, 3

ניסוי בסינתזה ותכנון (BackEnd) Layout  
של מעגלי VLSI

מהדורה חדשה - הערות נא לשלוח ל-goel@ee

כל הערה תתקבל בברכה!

עדכון אחרון - 30/06/2022 09:56

<http://www.ee.technion.ac.il/vlsi/>

מסמך זה כתוב בלשון זכר ע"מ להקל על הכתיבה אך מתייחס לנשים ולגברים כאחד. עמכם הסליחה.



## הנחיות בטיחות לסטודנטים במעבדות לאלקטרוניקה

### כללי:

תמצית הנחיות בטיחות מובאת לידיעת הסטודנטים כאמצעי למניעת תאונות בעת ביצוע ניסויים ופעילות במעבדות לאלקטרוניקה של הפקולטה להנדסת חשמל. מטרתן להפנות תשומת לב לסיכונים הכרוכים בפעילויות המעבדה, כדי למנוע סבל לאדם ונזק לציוד. אנא קיראו הנחיות אלו בעיון ופעלו בהתאם להן.

### מסגרת הבטיחות במעבדה:

- ♦ אין לקיים ניסויים במעבדה ללא קבלת ציון עובר בקורס הבטיחות.
- ♦ לפני התחלת הניסויים יש להתייצב בפני מדריך הניסוי לקבלת הנחיות בטיחות ותדריך ראשוני.
- ♦ אין לקיים ניסויים במעבדה ללא השגחת מדריך.
- ♦ מדריך הניסוי אחראי להסדרים בתחום פעילותכם במעבדה; הטו אוזן קשבת להוראותיו ונהגו על פיהן.

### עשו ואל תעשו:

- ♦ יש לידע את המדריך על מצב מסוכן וליקויים במעבדה או בסביבתה הקרובה.
- ♦ לא תיעשה במזיד ובלי סיבה סבירה פעולה העלולה לסכן את הנוכחים במעבדה.
- ♦ אסור להשתמש לרעה בכל אמצעי או התקן שסופק או הותקן במעבדה.

- ♦ היאבקות, קטטה והשתטות אסורים. מעשי קונדס מעוררים לפעמים צחוק אך הם עלולים לגרום לתאונה.
- ♦ אין להשתמש בתוך המעבדה בסמים או במשקאות אלכוהוליים, או להיות תחת השפעתם.
- ♦ אין לעשן במעבדה ואין להכניס דברי מאכל או משקה.
- ♦ יש לכבות מכשירי טלפון ניידים לפני הכניסה למעבדה.
- ♦ בסיום הפעולות יש להשאיר את השולחן נקי ומסודר.

### **בטיחות חשמל:**

- ♦ מדריך הניסוי עבר הכשרה בבטיחות חשמל והינו בעל תעודת חשמלאי בדרגה הנדרשת. היעזרו בו ובגורמים מקצועיים אחרים במעבדה, בעת חירום.
- ♦ בשולחנות המעבדה מותקנים בתי תקע ("שקעים") אשר ציוד המעבדה מוזן מהם. אין להפעיל ציוד המוזן מבית תקע פגום.
- ♦ אין להשתמש בציוד המוזן דרך פתילים ("כבלים גמישים") אשר הבידוד שלהם פגום או אשר התקע שלהם אינו מחוזק כראוי.
- ♦ אסור לתקן או לפרק ציוד חשמלי כולל החלפת נתיכים המותקנים בתוך הציוד; יש להשאיר זאת לטפול הגורם המוסמך.
- ♦ אין לגעת בלוח החשמל המרכזי, אלא בעת חירום וזאת לצורך ניתוק המפסק הראשי.

### **בטיחות אש, החייאה ועזרה ראשונה:**

- ♦ מדריך הניסוי עבר הכשרה בבטיחות אש, החייאה ועזרה ראשונה. העזרו בו ובגורמים מקצועיים אחרים במעבדה, בעת חירום.
- ♦ במעבדה ממוקם מטף כיבוי אש ותיק עזרה ראשונה, זהו את מקומו.
- ♦ אין להפעיל את המטפים ואין להשתמש בציוד העזרה הראשונה, אלא בעת

חירום ובמידה והמדריך וגורמים מקצועיים אחרים במעבדה אינם יכולים לפעול.

### יציאות חירום:

- ♦ במעבדה ישנה פתח יציאה אחת והיא משמשת כפתח היציאה גם בשעת חירום.
- ♦ בארוע חירום הדורש פינוי, כגון שריפה, יש להתפנות מיד מהמעבדה.

### דיווח בעת אירוע חירום:

- ♦ יש לדווח **מיידית** למדריך ולצוות המעבדה.
- ♦ המדריך או איש מצוות המעבדה ידווחו מיידית לקצין הביטחון בטלפון; 2740, 2222, נייד; 050-544575. **במידה ואין הם יכולים** לעשות כך, ידווח אחד הסטודנטים לקצין הביטחון.
- ♦ לפי הוראת קצין הביטחון, או כאשר אין יכולת לדווח לקצין הביטחון, יש לדווח, לפי הצורך; משטרה 7-100, מגן דוד אדום 7-101, מכבי אש 7-102 וגורמי בטיחות ו/או ביטחון אחרים. בנוסף לכך יש לדווח ליחידת סגן המנמ"פ לענייני בטיחות; 3033, 2146/7.
- ♦ בהמשך, יש לדווח לאחראי משק ותחזוקה; 4776, 052-419917.
- ♦ לסיום, יש לדווח לאחראי האקדמי; 4661, לעוזר למנהל; 4678, לאחראי ההנדסי; 4668, 4671 ולאחראי האדמיניסטרטיבי; 3276.

## תוכן עניינים

2	פרק 1 - הקדמה
2	פרק 2 - מבוא
2	2.1 הקדמה - שפת SystemVerilog
2	2.2 מושגים בסינתזה ובתזמון
2	2.2.2 סינתזה
2	2.2.3 תזמון ואילוץ תזמון (Timing Constraints)
2	2.2.4 שיפור הבדיקות של תכנון : Design For Testability (DFT)
2	2.3 כלי סינתזה – Design Vision
2	2.4 כלי ה- Innovus – Layout
2	ממשק המשתמש של Linux
2	2.4 רקע בסיסי למערכת ההפעלה Linux
2	הכנה ניסוי מספר 1
2	פרק 3 – דוחות הכנה
2	1. אנליזה של setup time ו- hold time
2	2. שיפור הבדיקות : Design For Testability (DFT)
2	3. Logical Equivalence Checking - LEC
2	4. סינתזה בעזרת script
2	הכנה ניסוי מספר 2
2	1. מבנה קובץ ה- LEF
2	2. קובץ mmmc.view וקובץ אילוץ תזמון
2	3. שלבי מימוש ה- layout
2	פרק 4 – ביצוע הניסויים
2	ביצוע ניסוי מס' 1
2	1. הכנת סביבת הסינתזה
2	2. סינתזה של המעגל
2	3. שיפור הבדיקות : Design For Testability (DFT)
2	4. LEC
2	ביצוע ניסוי מס' 2
2	1. תכנון ה- Floorplan
2	2. מיקום הפינים והגדרת רשתות האספקה
2	3. מיקום התאים הסטנדרטיים ובנית עץ השעון (Clock Tree Synthesis CTS)
2	4. חיווט : קווי האספקה והתכנון כולו
2	5. אנליזת תזמון Sign-Off והספק

חוברת זו מהווה תדריך והכנה לניסוי תכנון Backend של מעגלי VLSI במעבדה ל-VLSI. הניסוי מתבצע על גבי תחנות Linux ותוכנת Synopsys ו-Cadence לתכנון VLSI.

### מטרת הניסוי:

1. הכרה של כלי סינתזה וכלי Layout אוטומטיים וכן כל קבצי הטכנולוגיה הדרושים להפעלת התהליך הדרושים לכלים.
2. הכרה בסיסית של מבנים מסוימים בשפת SystemVerilog.
3. התנסות בסינתזה עם ה- Design Compiler של חברת Synopsys ושימוש ביכולות המתקדמות שלי הכלי.
4. התנסות בסיסית עם Logical Equivalence Checking.
5. התנסות ב- Layout אוטומטי באמצעות כלי ה- Innovus של חברת Cadence ושימוש ביכולות המתקדמות שלי הכלי.

### מבנה הניסוי:

הניסוי מורכב מ- 2 פגישות. כל פגישה אורכה ארבע שעות. לפני כל ניסוי יש להכין דו"ח מכין ולהגישו ב- labadmin לפני תחילת הניסוי.

### חלק א':

1. סינתזה ראשונית והכרה של ספריות ומודלים הדרושים לתהליך.
2. סינתזה תוך כדי אופטימיזציה של התזמון בשיטות שונות של הכלי design\_vision.
3. שילוב של Design For Testability – DFT.
4. ביצוע Logical Equivalence Checking על המעגל המסונתז.

### חלק ב':

1. סינתזה בעזרת סקריפט.
2. מימוש Layout אוטומטי כולל Floor Planning, Power Grid Design, Standard Cell Placement, Clock Tree Synthesis, Final Route. בדיקת חוקיות ה- layout.
3. אנליזה של setup time ו- hold time (עם min/max delay) כולל פתרון הבעיות.
4. ביצוע אנליזה הספק (Power Analysis).

### דרישות הניסוי:

- קריאת חוברת הניסוי בעיון רב (אפילו יותר מפעם אחת).
- הגשת דו"ח הכנה לניסוי לפי שאלות מפרק דו"ח הכנה.
- בוחן הכנה לניסוי.
- ביצוע הניסוי על תחנת עבודה.
- הגשת דו"ח סיכום שבועיים לאחר ביצוע חלק ב' של הניסוי.

### דרישות דו"ח סיכום:

#### הגשת:

- דו"ח מכין לשני חלקי הניסוי.
- דו"ח סופי לשני החלקים עם תשובות לכל השאלות שנשאלות במהלך הניסוי.

**"הסטודנט מתבקש למלא את טופס המשוב האלקטרוני הנמצא בקישור <http://www2.ee.technion.ac.il/Labs/EE Labs/>, הטופס ממלא באופן אונימי. אנו זקוקים לתגובותיכם על מנת לתקן ולשפר כמו גם לשבח".**

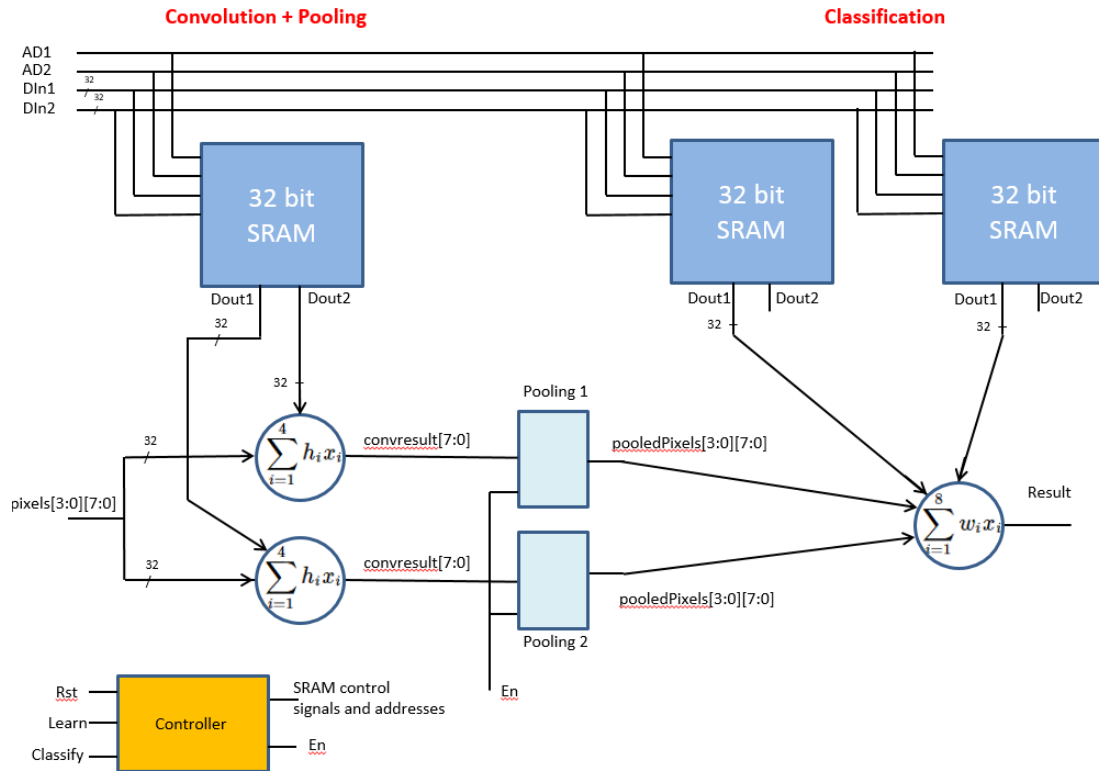
## פרק 2 - מבוא

תהליך תכנון טיפוסי של מעגל VLSI מורכב מהשלבים הבאים :

1. הגדרת המערכת ותכנון הארכיטקטורה
2. מימוש המערכת בשפה עלית כגון VHDL או Verilog.
3. סימולציות
4. סינתזה
5. בניית ה-Layout

מטרת הניסוי היא לבצע את השלבים של הסינתזה וה-Layout (כלומר תהליך ה-Backend Design) על תכנון קיים שמממש מאיץ למערכת לומדת. הניסוי יכלול הכרה והפעלה של כלי סינתזה ו-Layout מתקדמים. העבודה תבצע על מימוש מאיץ של מערכת לומדת שממומשת בניסוי אחר של המעבדה, כלומר ניסוי 98 - תכנון ארכיטקטורה למאיץ עבור מערכת לומדת ממומשת ב-Systemverilog. **חשוב : ניסוי 98 אינו מהווה קדם לניסוי זה.**

להלן סכמת המלבנים של המערכת :



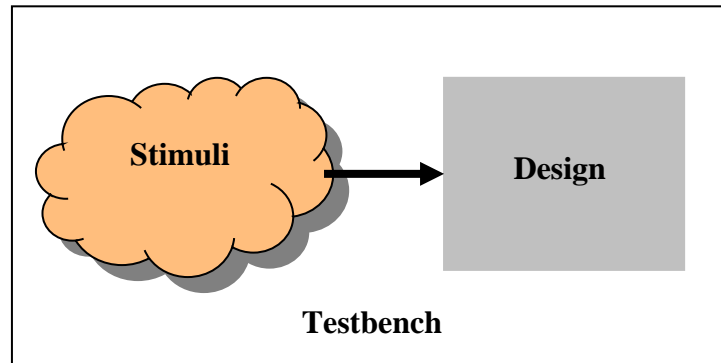
איור מס' 1 – הארכיטקטורה של מאיץ של מערכת לומדת

כפי שניתן לראות, המערכת מורכבת משלש יחידות SRAM, שני Convolution Neurons, שתי יחידות Pooling, Fully Connected Neuron אחד והבקר. הסבר מלא על התכנון מופיע בחוברת של ניסוי 98 אבל לצרכים של הניסוי הזה, אין צורך להבין את הפונקציונליות של המעגל. חשוב רק להכיר את המבנה הלוגי שמתאורת לעיל. לפני שנמשיך נתחיל במבוא קצר של שפת SystemVerilog.

## 2.1 הקדמה - שפת SystemVerilog

בפרק זה נתאר רק את המבנים של השפה שדרושות לביצוע הניסוי. המטרה העיקרית של העבודה עם שפת HDL היא מימוש התכנון וסביבת הסימולציה. הסברים נרחבים על השפה ניתן למצוא ב-

<http://webee.technion.ac.il/vlsi/Projects/Manuals/SV.pdf>



איור מס' 2

בשפת SystemVerilog התכנון ממומש באמצעות משפט module שניתן לתאר באופן הבא :

```
module name ( )
  interface // input and output definition
  variables //declare variables and type
  functional behaviour :
    // assign statements
    // always statements
    // instantiation of other module
endmodule
```

המודול מכיל את הגדרת הממשק (כניסות ויציאות), הגדרת משתנים ואת תיאור ההתנהגות הלוגית של הרכיב. ה- testbench (איור 2) מכיל הצבה של התכנון עצמו וקוד (stimuli) שיוצר את אותות הכניסה עבור התכנון. הסברים מפורטים יובאו בהמשך.

### אינסטנסיאציה – instantiation (הצבה)

אינסטנסיאציה היא הצבת עותק של מודול קיים וחיבורו ללוגיקה הנוספת במודול. לדוגמה שער and יכול להיות בנוי משער nand ומשער not המחוברים ביניהם. להלן דוגמה של משפט אינסטנסיאציה :

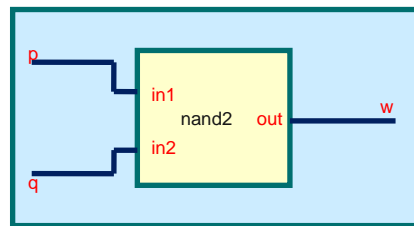
```
nand nand_U1 (.out(w), .in1(p), .in2(q));
```

המשפט הנ"ל יוצר עותק של יחידה מסוג nand בשם nand\_U1 ומחבר את הכניסות והיציאות שלו (out,in1,in2) לקווים של היחידה המכילה אותו (w,p,q). לא חובה לציין את שמות הכניסות והיציאות של ה- nand, לדוגמה :

```
nand nand_U1 (w,p,q);
```



במקרה זה החיבור יתבצע לפי סדר רישום האותות, כלומר w יחובר לכניסה או ליציאה הראשונה בהגדרה של ה- nand וכן הלאה כפי שמופיע באיור הבא :



איור מס' 3

אם במודול שבו מבצעים את האינסטנציאציה מופיעים סיגנלים בשמות זהים לכניסות וליציאות של הרכיב (במקרה הזה nand) אז ניתן לבצע אינסטנציאציה מבלי לציין אף שם. לדוגמא :  
nand U1 (\*);

ה "\*)" מציין ש- pins של ה- nand יחובר לרשתות (חוטים) בעלי שם זהה.

## כתיבת testbench

לאחר מימוש מודול צריך לבדוק אותו באמצעות סימולציות. לשם כך יש להכין סביבה פשוטה המכונה testbench. הסביבה היא למעשה מודול שמכיל הצבה של המעגל הנבדק בתוספת קוד שיוצר ערכים שמשתנים עם הזמן עבור הכניסות. מקובל לממש את הקוד באמצעות משפטי always ו/או initial לפי נוחיות המשתמש. בניסוי זה לא נעשה שימוש ב- testbench.

## 2.2 מושגים בסינתזה ובתזמון

נכיר מספר מושגים מעולם הסינתזה ואנליזת זמנים.

זהו התהליך שבו מתורגם מעגל המתואר בשפת HDL ( hardware description language) לדוגמה SystemVerilog למימושו באמצעות שערים לוגיים. כלי סינתזה

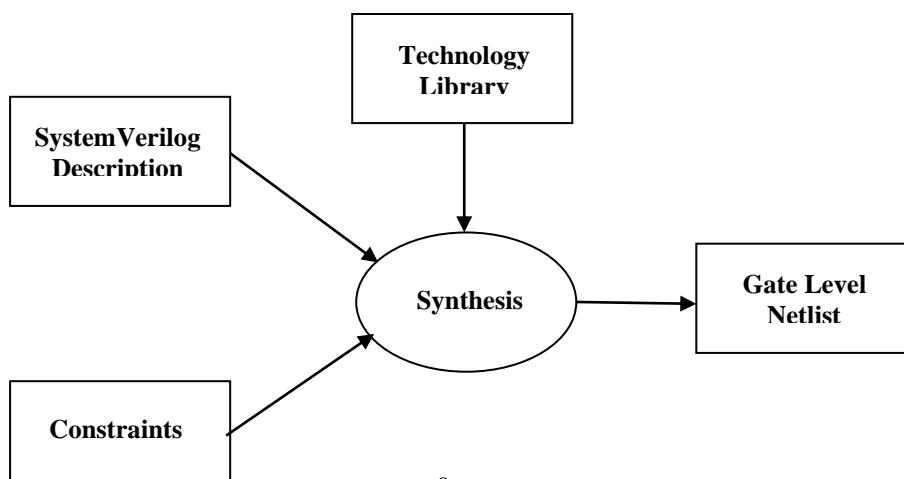
### 2.2.2 סינתזה

מקבל כקלט :

1. תיאור SystemVerilog של התכנון
2. ספרית התאים שעומדת לרשותו
3. אילוצי המשתמש

הפלט הוא מימוש המעגל באמצעות תאי הספרייה. להלן איור של מערכת הסינתזה :

איור מס' 4 : מערכת סינתזה



## ספריית השערים

ספריית השערים, או תאי הספרייה, היא מסד נתונים הכולל מספר רב של קבצים המתארים שערים לוגיים, פליפ-פלופים, רגיסטרים, ועוד. לכל סוג של שער או תא, יהיו מספר קבצים הכוללים את כל הנתונים הדרושים כדי להשתמש בתא. הנתונים האלה כוללים: הפונקציה הלוגית של השער, שמות הפינים של הכניסה והיציאה, ההשהיות שלו מכל כניסה ליציאה, המימוש שלו ב LAYOUT ועוד. לחלק גדול מהשערים קיימות מספר גרסאות בגדלים שונים: למשל SMALL הבנוי מטרני במידות W מינימלי, LARGE, MEDIUM ו XL. השימוש בגודל מאפשר לסינטזה שימוש בשער יותר מהיר לפי דרישות התיזמון. כל תהליך ייצור מגיע עם הספרייה שלו.

מפעל הייצור של הסיליקון מפתח עבור הלקוחות שלו את הספרייה, כדי להקל על הלקוח לפתח את הרכיב. לעיתים קיימות מספר ספריות שונות לאותו תהליך ייצור המאפשרות רכיבים דלי הספק, או עתירי ביצועים. תאי הספרייה נבנים ע"י יצרן התהליך בצורה האופטימלית ביותר המאפשרת מקסימום יעילות ואיכות. תאי הספרייה מאופיינים ע"י הייצורן מה שמבטיח את הדיוק של המודל שלהם.

במהלך הניסוי הזה אנו משתמשים בספריות של חברת Tower Semiconductors הישראלית הממוקמת במגדל העמק בטכנולוגיית CMOS 0.18u.

התוצר של הסינטזה הוא הסכימה של המעגל. בשפה המקצועית היא נקראת NET LIST. זהו קובץ שכולל את כל הפינים של הכניסות והיציאות, את כל קווי הסיגנלים הפנימיים, ואת תאי הסיפרייה שמשמשים בהם, והחיבוריות שלהם אל כל הסיגנלים. הסינטזה מממשת את הפונקציות הלוגיות ועושה להם מיטוב (אופטימיזציה וצימצומים לוגיים) ובהמשך גם מנסה לשפר את ההשהיות כך שהמעגל יעבוד בזמן המחזור הרצוי.

על מנת לאפשר ביצוע תהליך הסינטזה, יש לספק לכלי את הספריות שמאפיינות את התאים של הטכנולוגיה. ניתן לעשות זאת בעזרת הפקודות הבאות:

```
set link_library " dw_foundation.sldb\  
/tools/kits/tower/PDK_TS18SL/FS120_STD_Cells_0_18um_2005_12/DW_TOWER_  
_tsl18fs120/2005.12/synopsys/2004.12/models/tsl18fs120_typ.db  
dpram32x32_cb.db"
```

```
set target_library  
"/tools/kits/tower/PDK_TS18SL/FS120_STD_Cells_0_18um_2005_12/DW_TOWER_  
tsl18fs120/2005.12/synopsys/2004.12/models/tsl18fs120_typ.db dpram32x32_cb.db"
```

כל תא ספרייה מגיע עם סט של קבצים. שניים מהם הם בסיומת db ו lib. קבצי ה- db הם קבצים בינריים בפורמט של חברת Synopsys. התוכן של הקובץ הבינרי הוא בלתי קריא, ולכן נקבל את קובץ ה- lib (Liberty) בפורמט טקסטואלי של חברת Cadence שמכיל את אותה אינפורמציה אבל בפורמט טקסט. קובץ זה מאפיין את התאים מבחינה לוגית, מבחינת זמני תגובה ועוד. נתונים אלה חיוניים לביצוע הסינטזה. דוגמא של קובץ lib:

INVERTER

```
cell (INVX1) {  
  cell_footprint : inv;  
  area : 3;  
  cell_leakage_power : 0.0341756;  
  pin(A) {  
    direction : input;
```

```

capacitance : 0.0160164;
rise_capacitance : 0.0160164;
fall_capacitance : 0.0159693;
rise_capacitance_range ( 0.0160164, 0.0160164) ;
fall_capacitance_range ( 0.0159693, 0.0159693) ;
}
pin(Y) {
direction : output;
capacitance : 0;
rise_capacitance : 0;
fall_capacitance : 0;
rise_capacitance_range ( 0, 0) ;
fall_capacitance_range ( 0, 0) ;
max_capacitance : 0.402017;
function : "(!A)";
timing() {
related_pin : "A";
timing_sense : negative_unate;
cell_rise(delay_template_5x5) {
index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
values ( \
"0.147583, 0.217035, 0.355377, 0.907001, 1.73349", \
"0.224219, 0.291274, 0.428044, 0.977438, 1.80305", \
"0.365232, 0.445422, 0.582152, 1.12428, 1.94149", \
"0.46186, 0.55044, 0.700788, 1.23784, 2.0566", \
"0.75585, 0.872701, 1.05674, 1.62712, 2.42849");
}
rise_transition(delay_template_5x5) {
index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
values ( \
"0.09712, 0.164583, 0.299488, 0.839163, 1.64852", \
"0.117867, 0.173862, 0.30016, 0.839221, 1.64879", \
"0.174063, 0.229801, 0.335188, 0.839268, 1.64864", \
"0.212233, 0.270216, 0.376329, 0.849921, 1.64867", \
"0.322205, 0.398154, 0.51726, 0.945327, 1.66712");
}
cell_fall(delay_template_5x5) {
index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
values ( \
"0.166552, 0.247549, 0.408769, 1.04944, 2.00953", \
"0.234229, 0.313057, 0.472061, 1.11143, 2.07233", \
"0.366262, 0.455096, 0.610195, 1.24173, 2.19801", \
"0.456648, 0.553252, 0.718877, 1.34254, 2.29658", \
"0.732451, 0.855462, 1.05488, 1.6961, 2.63335");
}
fall_transition(delay_template_5x5) {
index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
values ( \
"0.103393, 0.176429, 0.322563, 0.906541, 1.78269", \
"0.119174, 0.182264, 0.32239, 0.906789, 1.78367", \
"0.176141, 0.235883, 0.351112, 0.906842, 1.78324", \
"0.215097, 0.277743, 0.390555, 0.911993, 1.78282", \
"0.333403, 0.409933, 0.534675, 0.99795, 1.79573");
}
}
internal_power() {
related_pin : "A";
}

```

```

rise_power(energy_template_5x5) {
  index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
  index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
  values ( \
    "0.389002, 0.393869, 0.398499, 0.402791, 0.40412", \
    "0.442829, 0.430625, 0.423648, 0.413675, 0.40862", \
    "0.617376, 0.577985, 0.535474, 0.47171, 0.448634", \
    "0.768465, 0.717378, 0.652041, 0.547541, 0.493993", \
    "1.29533, 1.2207, 1.10425, 0.86946, 0.7254");
}
fall_power(energy_template_5x5) {
  index_1 ("0.06, 0.18, 0.42, 0.6, 1.2");
  index_2 ("0.025, 0.05, 0.1, 0.3, 0.6");
  values ( \
    "0.011845, 0.00625, 0.001276, 0.005383, 0.006277", \
    "0.022784, 0.017685, 0.013616, 0.008902, 0.007861", \
    "0.173984, 0.138231, 0.101902, 0.054497, 0.035459", \
    "0.311874, 0.256871, 0.196965, 0.106444, 0.069818", \
    "0.81705, 0.723427, 0.595846, 0.372799, 0.246518");
}
}
}
}
}

```

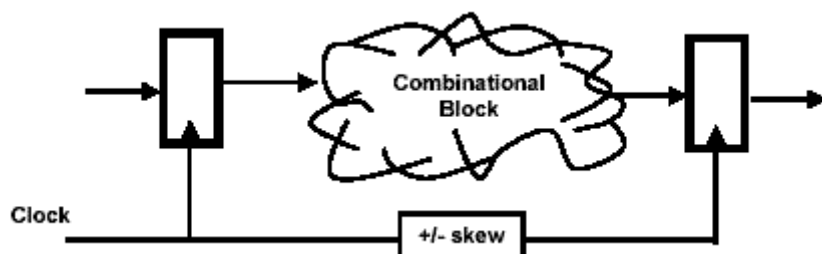
לא נעמיק בהסברים מפורטים על קובץ זה.

### 2.2.3 תזמון ואילוץ תזמון (Timing Constraints)

בסעיף זה, מופיע הסבר על מושגים שונים בנושא התזמון ובהמשך יובא הסבר על הפקודות הרלוונטיות בכלי הסינתזה. סביבת הפיתוח כוללת כלי שמנתח מעגל מבחינת התזמון שלו נקרא **timing verifier (TV)**. בבדיקה טיפוסית של מעגל סינכרוני, יש לבדוק את כל המסלולים הבאים :

- א. מסלולים בין הכניסות של המעגל לכניסות של רגיסטרים שבמעגל.
- ב. מסלולים מיציאות של רגיסטרים שבמעגל ליציאות המעגל.
- ג. מסלולים מיציאות של רגיסטרים שבמעגל אל הכניסות של רגיסטרים אחרים במעגל.

ה - **critical path** הוא המסלול בעל ההשהיה הארוכה ביותר .



איור מס' 5

#### Slack

אם במהלך העבודה עם TV מגדירים שעון בעל זמן-מחזור T למערכת, הכלי יבדוק האם המעגל יכול לעבוד בקצב הזה, כלומר האם ההשהיה של המסלול הקריטי  $T_c$  קצר ממחזור השעון. אם הוא קצר יותר אז אומרים שלמסלול הקריטי, slack חיובי אחרת ה- **slack** שלילי!!.

$$\text{Slack} = T - T_c$$

ברור שתמיד צריך להגיע למצב שבו ה-slack יהיה חיובי. כמוכן שתיאור פשטני זה מזניח נתונים חשובים כגון **setup time** ו-**hold time** (ראה המשך).

### Max delay requirement

$$\text{Longest\_path\_delay} + T_{\text{clock\_to\_q}} < T_{\text{cycle}} - T_{\text{setup}} + T_{\text{skew}}$$

קיום דרישה זאת מבטיח שהמידע מגיע לפני שהשעון עלה.

### Min delay requirement

$$\text{Shortest\_path\_delay} + T_{\text{clock\_to\_q}} > T_{\text{hold}} + T_{\text{skew}}$$

קיום דרישה זאת מבטיח שהמידע לא מגיע מהר מידי. אסור שיקרה מצב שבו המידע בכניסה של ה-FF השני משתנה לפני שהשעון מגיע ובכך ננעל המידע החדש ולא הישן כפי שהיה צריך. קיימים שני פתרונות לבעיה זאת :

1. להוסיף השהייה במסלול הנתונים
2. לעכב את השעון שמגיע ל-FF הראשון

## 2.2.4 שיפור הבדיקות של המעגל : Design For Testability (DFT)

בתהליך הייצור קורה הרבה פעמים שנופלים פגמים בחומר בין אם זה בטרנזיסטורים, או בקווי ההולכה ובחיבורים. ברוב המקרים הפגמים האלה מתבטאים בשלוש צורות התנהגות:

1. STUCK AT 1
2. STUCK AT 0

3. קצר בין סיגנלים הגורם להם לשנות מצב בייחד.

קיימים גם מנגנוני כשל נוספים מורכבים יותר, אך לא נתעסק בהם הפעם.

לאחר הייצור נדרש לבדוק את השבב בצורה מקיפה כדי לוודא שהוא פועל בצורה מושלמת והוא נקי מפגמים. אחת הבדיקות החשובות היא בדיקה פונקציונלית. צריכים לוודא שאף צומת אינו מתנהג באחת משלוש הצורות הנ"ל. על מנת לבצע בדיקה זאת, יש לספק לשבב כניסות שיגרמו לכל אחד מהצמתים לשנות מצב וגם שאפשר יהיה להבחין בשינוי ביציאות השבב. פעולה זאת קשה ביותר עבור שבבים גדולים בעיקר עבור צמתים הנמצאים עמוק בתכנון.

נוכל לדמיין כל מערכת לוגית כאוסף של פונקציות לוגיות ללא זיכרון, ורכיבי FF שאוגרים את תוצאות הביניים בין מחזור שעון למחזור שעון. ראה דוגמה באיור מס' 5. כדי לבדוק כל מכלול כזה של פונקציות נרצה להיות מסוגלים להכניס את כל סט הצירופים האפשריים בכניסות, ולדגום את כל היציאות. קוראים לזה *controllability* ו-*observability*.

מה שמאפשר לנו לעשות את זה בקלות יחסית זה ה-*scan*.

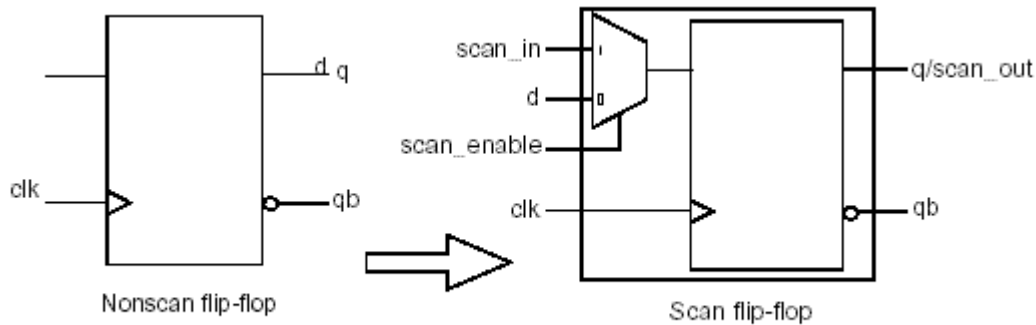
הדרך המקובלת לפתרון הבעיה היא הוספת חומרה לתכנון שמטרתה היא לשפר את הבדיקות שלו. *Internal Scan* היא אחת הטכניקות הנפוצות ל-DFT. בשיטה זאת, מחליפים כל ה-FF בתכנון ב-*Scan FF* כפי שמתואר באיור מס' 6. בעבודה רגילה ה-*Scan FF* מקבל את הכניסה הרגילה וב-*test mode* הוא מקבל כניסה מה-*scan\_in* port.

בעזרת ה-*Scan FF*, אנו יוצרים מספר שרשראות שמחברות את כל ה-FF שבמעגל בצורת רגיסטר הזזה דרך כניסת ה-*SCAN* של ה-FF. כל שרשרת כזאת מחוברת לשני פינים חיצוניים של הרכיב. פין אחד להכנסת המידע, ופין אחד במוצא. כאשר מכניסים את הרכיב למצב בדיקה, ניתן להזין פנימה ל-*shift register* את ה"1"-ים וה"0"-ים, ולאחר מכן לשלוף החוצה את תוצאות החישוב של כל רשת לוגית.

*scan\_in* : פורט זה מחובר ליציאה של *Scan FF* אחר. בצורה זאת ב-*test mode* ניתן לצור *shift register* ארוך (*Scan Chain*) המורכב מכל ה-FF שבתכנון. ע"י הפעלה נכונה של המעגל, ניתן

להכניס ל-FF-ים כל ערך רצוי ע"י רצף של פעולות הזזה. אחרי הכנסת ערך רצוי לכל ה-FF-ים מפעילים את השבב במוד רגיל של עבודה במשך מחזור אחד שבסיומו ננעלות התוצאות באותם ה-Scan FF. כעת ניתן להוציא את התוצאות החוצה ע"י סידרה נוספת של פעולות הזזה. ברור ששינויים אלה מגדילים את שטח המעגל ויכולים לפגוע בביצועים, אבל הם חיוניים לשיפור הבדיקות.

ניתן לבצע הוספת ה- scan chain בשלב הסינתזה בתצורות שונות באופן אוטומטי.



איור מס' 6 : Scan Flipflop

כחלק מתהליך הפיתוח של הפרויקט נשתמש בכלי אוטומטי שינתח את המעגלים הלוגיים, וייצור את הרצפים של הכניסות ל SCAN וכן את הרצפים של היציאות הצפויות (שנקרא להם VECTORS), כך שבקו הייצור נוכל להשתמש במכשיר בדיקה (TESTER) שיריץ את הבדיקות על כל רכיב לפני שהוא יוצא מן המפעל. בצורה כזאת ניתן להגיע לכיסוי בדיקות כמעט מושלם בעזרת הכלים האוטומטיים במינימום של השקעה בעבודה הנדסית, אך במחיר מסוים של תוספת חומרה, עם כל הכרוך בכך.

### 2.3 כלי סינתזה – Design Vision

ה- **dc design compiler** (או **design vision** בגרסתו הגרפית) הוא כלי הסינתזה של חברת **Synopsys**. הכלי מקבל כקלט תיאור **RTL** ויוצר כפלט מעגל ברמת השערים המממש את ה-**RTL**. הכלי מופעל באמצעות הפקודה **design\_vision** שגורם לפתיחה של חלון גרפי המאפשר הרצת פקודות מתוך התפריטים.

כל פקודה שנריץ בעזרת הממשק הגרפי תופיע גם בחלון ה- **terminal** ממנו הרצנו את הכלי. נוכל להריץ גם פקודות על ידי כתיבתם בחלון ה-**terminal**. נוכל גם לרשום את כל הפקודות בקובץ אחד (כל פקודה בשורה נפרדת) ולהריץ את כל הקובץ בבת אחת על ידי **file->execute script...**

ניתן לקבל את קובץ כל הפקודות שהרצנו ע"י **file->save info->design setup**

תהליך של סינתזה מתחלק ל- 3 שלבים :

1) קריאת קבצי ה- **vhdl/systemverilog** ובדיקתם לשגיאות **syntax** : לחץ על **file->read**.

2) אופטימיזציה ומיפוי לשערי ספרייה (סינתזה תלוית טכנולוגיה) : לפני התחלת שלב זה יש להגדיר את אילוצי התכנון (אופציונאלי – ראה בהמשך) ע"י האפשרויות השונות שבתפריט ה- **attributes** או ע"י קריאת קובץ אילוצים בעזרת.

3) קריאת קובץ אילוצים. בקובץ האילוצים מגדירים שת אילוצי התזמון וגם ניתן לבחור את כמות העבודה שהכלי ישקיע בכל שלב ע"י בחירת ה- **effort**. ז"א הכלי יפעיל אלגוריתמים שונים כדי לשנות את המעגל כך שכל המסלולים הלוגיים יעמדו בזמן המחזור הנדרש, וכן המשתמש יכול לבחור אפשרויות מיטוב שונות. ההוראות המדויקות של איך לבצע את זה יופיעו בהמשך בזמן הניסוי.

אילוצים שניתן להגדיר לכלי :

- **create\_clock** : פקודה להגדרת שעון, לדוגמא :

**create\_clock -name "CLK" -period 10 -waveform {0 5} clock**

מגדירה אות שעון על ההדק **CLK** בעל מחזור **10ns**, המתחיל ב-0 ועולה ל-1 בזמן **5ns**. בד"כ נבצע את הפקודה באמצעות התפריט. אם השעון מוגדר, DC ינסה לסנתז מעגל העומד בתדר השעון.

ניתן להגדיר שעון גם בעזרת התפריט ע"י בחירת הדק השעון ולחיצה על **clock** -> **specify** attributes. למציאת הדק השעון יש ללחוץ על הרכיב הרצוי ולשנות את הבחירה (hierarchical) cells ל pins/ports.

- **set\_max\_delay** : קובע אילוץ של השהיה המרבית בין צמתים שונים במעגל. דוגמא :

**set\_max\_delay 10 -to D[\*]**

ה- **dc** ינסה לבנות מעגל כך שהשהיה בין כל צומת במעגל לצמתים ששם מתאים ל- **D[\*]** לא תעלה על **10ns**. אין ודאות שה-**dc** יצליח לבנות מעגל כזה.

- **set\_max\_area** : קובע אילוץ לגבי השטח הכולל של המעגל.

- **set\_input\_delay** : מאפשר למשתמש להגדיר מתי (ביחס לעליית השעון) אותות הכניסה מוכנים.

- **set\_output\_delay** : מאפשר למשתמש להגדיר מתי (ביחס לעליית השעון) אותות היציאה צריכים להיות מוכנים.

- **set\_dont\_touch block\_name** : מורה לכלי לא לסנתז בלוק מסוים.

**הגדרת אילוצי תזמון :**

**set\_input\_delay 0 -clock clk a**

**set\_output\_delay 0 -clock clk z**

- הפקודה הראשונה מגדירה שהשהיה של הכניסות **a** יחסית לשעון **clk** הוא 0, כלומר הכניסה מופיעה בתחילת מחזור השעון.

- הפקודה השנייה מגדירה שהשהיה של יציאות **z** יחסית לשעון **clk** הוא 0, כלומר היציאה תהיה מוכנה לתחילת מחזור השעון.

קבלת נתונים על הביצועים של המעגל :

- שטח : ע"י **design->report area**

- צריכת הספק : ע"י **design->report power**

- תזמון : ע"י **timing->report timing**

## 2.4 כלי ה- Innovus – Layout

ה- **layout** הוא השלב הבא בעבודת הפיתוח שבו אנו עוברים ממצב של סכמה למימוש פיזי של המעגלים. בשלב זה אנו ממקמים את המימוש הפיזי של כל שער בשטח המוקצה לנו בציפ על הסיליקון, ומחברים את כל קווי האותות והספקים. בסיום התהליך יוצר קובץ הכולל את שרטוט כל המסכות שבהן ישתמשו בקו הייצור. כלי ה- **Innovus** של חברת **Cadence** מאפשר בין היתר תכנון ומימוש של המסכות (או **layout**) של מעגל **VLSI** באופן אוטומטי. **Innovus** הוא כלי בעל יכולות רבות כולל:

- **physical synthesis**

- **clock tree synthesis**

- **timing analysis**

- **power analysis**

- **voltage (IR) drop analysis**

- **signal integrity analysis**

- **crosstalk analysis**

מעבר לאנליזות השונות הכלי מסוגל לבצע תיקונים אוטומטיים ב- **layout** של התקלות שמתגלות במהלך בדיקות שונות. בבניית **layout** אוטומטי למעגל, ראשית מבצעים את תכנון ה- **floorplan** כלומר מיקום הבלוקים בתכנון ומיקום הכניסות/יציאות שלהם, לאחר מכן מבצעים את מיקום תאי הספרייה והחיווט שלהם בכל תת בלוק ובין כל תתי הבלוקים.

ראשית נתאר מימוש **layout** של מעגל שכולו מבוצע ברמת היררכיה אחת. תיאור של שימוש בכלי האנליזה השונים יובא בהמשך. באופן כללי ניתן לומר שבניית **layout** שטוח מורכב מהשלבים הבאים :

- א. קריאת קבצי הטכנולוגיה
- ב. קריאת קובץ ה- **verilog** (ברמת שערים כלומר אחרי סינתזה) של המעגל
- ג. אתחול והגדרת ה- **Floorplan** הראשוני
- ד. מיקום התאים
- ה. מימוש רשת האספקה
- ו. חיווט התכנון

הפעלת הכלי מתבצעת בעזרת הפקודה : **innovus**  
קריאת התכנון מתבצע ע"י הפקודה **Design File-> Import**. בחלון שנפתח לחץ על **load** וטען את קובץ **env.globals**, הקובץ מגדיר את הדברים הבאים :  
- **top.v** זהו הקובץ שנוצר ע"י כלי הסינתזה.  
- קבצי **lef** המכילים אפיון גאומטרי של התאים.  
- קובץ ה- **mmmc.view** – ראה הסבר בהמשך.  
- קובץ אילוצי תזמון **Top.sdc**. כאן מגדירים את השעון למשל. קובץ זה נקרא ע"י קובץ ה- **mmmc.view**.

**חשוב** : הכלי מאפשר הצגת התכנון בשלשה ייצוגים שונים :

- א. **Floorplan View** : מציג מיקום הבלוקים
- ב. **Amoeba (placement) View** : מציג מיקום התאים הבסיסיים
- ג. **Physical View** : מציג את כל הפרטים של המימוש

ניתן לעבור בין הייצוגים ע"י לחיצה על שלשת הכפתורים  בצד ימין של המסך.

### קובץ ה- **mmmc.view**

בקובץ זה ניתן להגדיר סטים של תנאים בכל מיני צירופים שישמשו את הסימולטור כדי לחשב תיזמונים, והספקים ועוד.

ישנם מספר גורמים שמשפיעים על מהירות התפשטות האותות במעגל וכן על צריכת ההספק עבור סכמה נתונה. ז"א, לאחר שנקבעו השערים, החיבורים, השעונים, וכל הסכימה, יש צורך לחשב את התיזמון וההספקים בתנאים שונים.

הגורמים העיקריים הבלתי תלויים שמשפיעים על התיזמונים וההספקים הם :

1. מתח העבודה – במתח גבוה, הכל יעבוד יותר מהר וצריכת ההספק תגדל.
2. טמפרטורה – בטמפ' נמוכה, הכל יעבוד יותר מהר וצריכת ההספק תגדל.
3. תהליך הייצור – התהליך אינו דטרמיניסטי אלא אקראי. לדוגמא ריכוז ההשתלות משתנה מפרוסה לפרוסה וגם בתוך הפרוסה עצמה ובהתאם לזאת מהירות הטרנזיסטור משתנה.

דוגמא נוספת, אורך התעלות – יצרן הסיליקון אינו יכול תמיד לייצר את אורך התעלות בדיוק באותה מידה. יש לו מטרה של אורך מסוים, למשל בניסוי שלנו 180 ננו מטר. היצרן מכוון את המכשירים כך שרוב התעלות ייצאו שם. אבל חלק מההתקנים המיוצרים יוצאים עם תעלות ארוכות יותר, או קצרות יותר.

גדלי ה- **R** וה- **C** – כמו בהסבר הקודם – נקבל צירוף של **SLOW RC** שמתאים להתנגדות גבוהה יותר של קווי המתכת, וקיבולים גבוהים יותר גם של ה- **GATE** בטרני. והפוך ב- **FAST RC**.

בעת פיתוח תהליך הייצור, ובניית הסיפריה, הייצרן יאפיין את כל השערים על כל תווך המנעד של הפרמטרים האלה, וייצור בד"כ 3 נק' איפיון – **MIN**, **TYPICAL**, **MAX**. רוב הייצור ייצא



במצב **TYP**, אבל הפיזור של אורכי התעלה ופרמטרים נוספים, יהיה בד"כ גאוסייני. כך שבנקי ה **MAX** יאופיין החומר האיטי יותר ברמה של 3 סיגמה, ובנקי ה **MIN** להיפך. אנו נרצה שהמעגל שלנו ימשיך לתפקד בכל מנעד התחומים האלה. כדי למקסם את הרווחיות שלנו. הסיבה – אנו משלמים מחיר קבוע ליצרן הסיליקון על כל מנה, גם אם היא יוצאת בקצה הטווח.

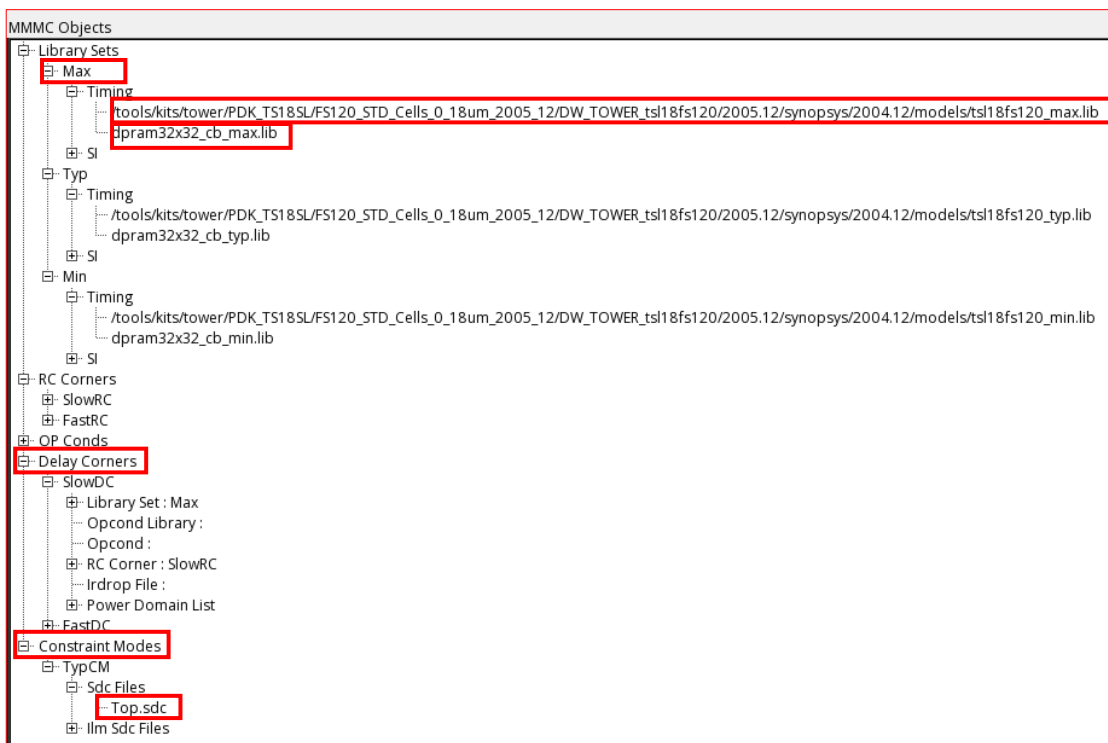
בקובץ **mmmc.view**, נוכל להגדיר לסימולטור מספר רב של צירופים של פרמטרים אלה בשם **"DELAY CORNER"** כך שבזמן הסימולציה נוכל לבחור כל אחד מהם ולבדוק את תפקוד המעגל.

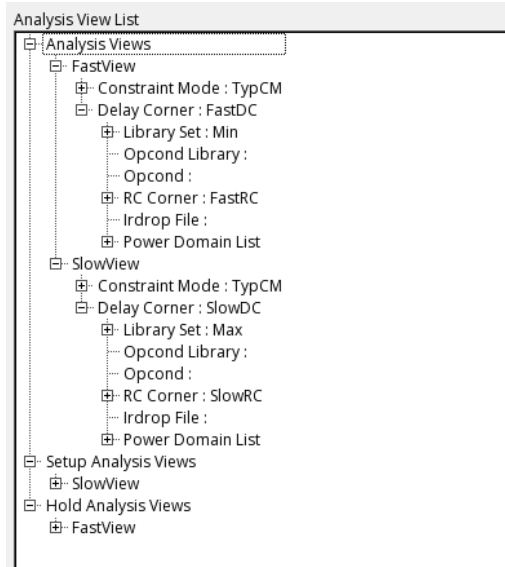
אגב, בתהליכים מודרניים יותר, נוכל לקבל גם איפיונים קיצוניים יותר, ז"א מסיגמה גדול יותר. אלה יקראו בד"כ **FF, SS, FFF, SSS**, וכו'. כמוכן יתכנו גם צירופים שבהם התעלות ב **N** וב **P** הפוכים, אחד מהיר ואחד איטי, ולהיפך.

בד"כ נדרוש לראות שהמסלולים הקריטיים, עדיין עוברים במצב **SLOW** (או **max delay**), ושאינן בעיות **HOLD** או מירוצים במצב **FAST** (או **min delay**) שזה כולל גם את צירוף המתח והטמפ'.

קובץ ה- **mmmc.view** עושה שימוש בקבצי **lib** המכילים אפיון של התאים מבחינה לוגית, תזמונים, הספקים, ועיד. בקובץ זה מוגדרות פינות כפי שהוסבר עבור אנליזת הזמנים. הקובץ מכיל את ההגדרה של **MMMC Objects** ו- **Analysis Views** כפי שמופיע באיור הבא.

בדוק באיור מס' 7 שבו מופיע מבנה קובץ ה- **mmmc.view** כיצד מוגדרים ה- **MMMC Objects** וה- **Analysis Views**.





איור מס' 7 : מבנה קובץ ה- mmmc.view

### שילוב זיכרונות RAM

אם משלבים זיכרונות RAM (כפי שנעשה בניסוי זה) גם עבורם יש לספק:

1. קובץ .lef

2. קבצי lib

קבצים אלו מגיעים כתוספת לקבצי lef ו- lib של התאים הסטנדרטיים.

### בנית ה- layout

השלבים העיקריים בבניית ה- layout הם כדלקמן :

### הגדרת ה- Floorplan

הגדרת צורת ה-Floorplan מתבצעת בעזרת לחיצה על **Floorplan->Specify**. בתפריט זה ניתן להגדיר את המאפיינים של ה-Floorplan:

א. המרחק בין הליבה (core) ל- I/O pads. במקרה שלנו נקבע מרחק זה כ-20.

ב. הגדרת גודל השבב: ראשית יש לעבוד עם גודל שבב שנקבע ע"י ברירת המחדל. אם יסתבר שהגודל לא מתאים, ניתן לשנותו בעזרת השדות **Die Size by Width Height**.

### הגדרת רשתות האספקה של התאים

יש להגדיר לכלי איזה pin של כל תא יחובר לרשתות האספקה. פעולה זאת מבוצעת בעזרת הפקודה **Power -> Connect Global Nets**. אנו נבצע זאת באופן אוטומטי בעזרת **script** מתאים.

### מיקום תאי RAM

מבצעים זאת באופן ידני בעזרת ה- **icon** של פקודת **move** (בצד שמאל למעלה). בד"כ תאי ה- RAM נמצאים בצד ימין למטה. חשוב להשאיר מקום פנוי מסביב לתאי הזיכרון עבור החיווט. הסבר מפורט בזמן הניסוי.

## הגדרת רשתות האספקה VDD ו-VSS

ברשתות האספקה אנו נרצה להשתמש במוליכים רחבים ככל שניתן כדי להקטין את התנגדותם. ככל שההתנגדות קטנה יותר כך יהיה פחות מפל מתח, והשערים יקבלו מתח יציב יותר ומדויק יותר. אבל מפאת מגבלות מקום משתמשים בחוטים עבים כאשר הזרם גבוה ובחוטים דקים יותר כאשר הזרם נמוך (ממש ליד התא למשל).

לשם כך בונים שתי טבעות (עבור **VDD** ו- **VSS**) הממומשים בחוטים עבים (ז"א רחבים, שכן הגובה לא ניתן לשינוי) מסביב לכל היחידה שלנו. בנוסף מוסיפים עוד כמה חוטים עבים שחוצים את התכנון באופן אנכי או אופקי.

החיבור של כל התאים לרשתות האלו יתבצע באמצעות חוטים דקים. הסבר מפורט בזמן הניסוי איך ניתן לממש את רשתות האספקה. אנו נרצה לצמצם עד כמה שניתן את מפלי המתח על קווי האספקה – גם **VDD** וגם **VSS**. זאת ניתן לעשות ע"י עיבוי הקווים האלה. אם יוצר מפל מתח – בין אם ב **VDD** או ב **VSS** זה יקטין את מתח העבודה של השער, ויגדיל את ההשהיה שלו, בצורה שלא תהיה ממודלת ע"י הסימולטור. כמוכן, במצבים של זרם גבוה מדי בקו צר, יתכן מצב של "פיוז" והקו ישרף.

ניתן לממש את רשתות האספקה בעזרת שתי פקודות :

- א. **Power->Power Planning->Add Rings**
- ב. **Power->Power Planning->Add Stripes**

בעזרת **Power->Power Planning->Add Rings** ניתן להוסיף טבעות אספקה סביב כל בלוק או סביב כל הליבה.

בעזרת **Power->Power Planning->Add Stripes** ניתן להוסיף רצועות נוספות של קווי האספקה.

### מיקום התאים הסטנדרטיים

זה השלב שבו ממקמים באופן אוטומטי את התאים בצורה אופטימלית ככל שניתן. על מנת לגרום ל- **Innovus** למקם תאים תוך כדי אופטימיזציה בתזמון הכללי יש לספק את האילוצים המתאימים.

את מיקום התאים מבצעים בעזרת הפקודה :

**Place->Standard Cells**

### עץ שעון מאוזן – Clock Tree Synthesis (CTS)

תפקיד עץ השעון הוא להבטיח שהשעון מגיע לכל הפליפי פלופים באותו זמן פחות או יותר. להלן דוגמה של רצף הפקודות שמממשות עץ שעון. ראשית מגדירים את סיגנל השעון :

```
create_ccopt_clock_tree -name top -source clk
```

לאחר מכן מגדירים אלו תאים יכולים להשתתף בבניית עץ השעון :

```
set_ccopt_mode -cts_inverter_cells {invbd2 invbd4 invbd7 invbda invbdf invbdk}
```

```
set_ccopt_mode -cts_buffer_cells {bufbd1 bufbd2 bufbd3 bufbd4 bufbd7}
```

בשלב הבא מגדירים את זמני עליה וירידה מכסימליים :

```
set_ccopt_property target_max_trans 220ps
```

וגם את ה- skew המכסימלי

```
set_ccopt_property target_skew 0.2
```

הפקודות שיוצרות את העץ הן :

```
create_route_type -name RT_trunk_leaf -top_preferred_layer M4 - \
bottom_preferred_layer M3 -preferred_routing_layer_effort high
set_ccopt_property route_type RT_trunk_leaf -net_type leaf
set_ccopt_property route_type RT_trunk_leaf -net_type trunk
set_ccopt_mode -integration native
ccopt_design -cts
```

ניתן לראות את העץ בעזרת : **Clock->CCopt Clock Tree Debugger**.

#### מילוי הרווחים

נהוג למלא אזורים באמצעות תאים ריקים או תאי **decap**. גם כאן קיים **script** שעושה זאת.

#### חיווט קווי האספקה

זה השלב שמחברים את רשתות האספקה לכל התאים שבתכנון. זה מתבצע ע"י הפקודה **sroute**.

#### חיווט התכנון

ניתן לבצע את החיווט הסופי בעזרת **Route->Nanoroute->Route**.

#### בדיקת התזמון

בדיקת העמידה ב- **setup time** וב- **hold time** יכול להתבצע בכל שלב של התכנון. הרצת אופטימיזציה יכולה לשפר באופן משמעותי את הביצועים מבחינת הזמנים.

במהלך הניסוי נבצע את כל השלבים שתוארו ויתקבל הסבר מפורט על הדרך לבצע כל שלב.

## 2.4 רקע בסיסי למערכת ההפעלה Linux

[סטודנטים בעלי ניסיון במערכת ההפעלה Linux יכולים לפסוח על חלק זה.]

### ממשק המשתמש של Linux

המעבדה מצויידת במחשבי PC עליהם מותקנת מערכת ההפעלה RedHat Linux. זו מערכת בעלת ממשק גרפי (GUI) המאפשר ביצוע של מרבית הפעולות הנדרשות הן מתוך שורת פקודה כמקובל ב-Linux והן דרך תפריטים וצלמיות (Icons) כמקובל במערכות מסוג זה. עם הכניסה למערכת, מופיע על המסך שולחן עבודה (Desktop) ובתחתיתו לוח הפעלה. לוח ההפעלה מאפשר הפעלת תוכניות נפוצות רבות, מעבר בין 4 שולחנות עבודה שונים, קבלת מידע בסיסי על מצב המערכת ועוד. אולם, במעבדה זו כמעט אין צורך בו. זאת, כיוון שאת כל הפעולות הבסיסיות הנדרשות לנו, ניתן להפעיל מתפריט **מהיר** (Workspace Menu), הנפתח בלחיצת עכבר ימני על שטח ריק בשולחן העבודה. כדי להשלים ההכרות עם שולחן העבודה, נוסיף כי על משטח זה נפתחים חלונות כבכל מערכת חלונאית, ומיזעורם (Minimize) מביא להצגתם כצלמיות בעמודה המתחילה בפינה השמאלית העליונה של המסך.

## מערכת הקבצים

להלן מספר פקודות בסיסיות לניהול מערכת הקבצים ב-LINUX, דרך שורת פקודה (פעולות דומות ניתן לבצע גם ממנהל הקבצים - File Manager):

פירוש	פקודה
שם הספרייה הנוכחית	pwd
שינוי הספרייה הנוכחית	cd [שם ספרייה]
מעבר לספריית האב של ההנוכחית	cd ..
העתקת קובץ	cp [קובץ יעד] [קובץ מקור]
העתקת קובץ לספריית בת של הנוכחית	cp [שם ספרייה] [שם קובץ]
העתקת קובץ לספריית האב	cp [שם קובץ] ..
העברת קובץ לספריית בת של הנוכחית	mv [שם ספרייה] [שם קובץ]
שינוי שם קובץ	mv [שם ישן] [שם חדש]
העברת קובץ לספריית האב	mv [שם קובץ] ..
מחיקת קובץ	rm [שם קובץ]
יצירת ספריית בת לספרייה הנוכחית	mkdir [שם ספרייה]
מחיקת ספריית בת של הנוכחית	rmdir [שם ספרייה]
רשימת שמות הקבצים הגלויים בספרייה הנוכחית	ls
רשימת פרטי כל הקבצים בספרייה הנוכחית	ls -la
כפי שניתן לראות, שם ספריית האב של הנוכחית הוא תמיד ".." (נקודה-נקודה).	
כשברצוננו להתייחס לספרייה אחת (הנמצאת תחת אותו אב), ניתן לרשום: [שם ספרייה]/..	

## עורכי טקסט

במהלך הניסויים נדרש להשתמש בעורך טקסט (Text Editor), כדי ליצור הקבצים. קיימים מספר עורכים במערכת כגון: nedit, gedit ועוד. ניתן להשתמש בכל עורך לפי שיקול המשתתפים בניסוי. מומלץ להשתמש בעורכים nedit, gedit אשר "מכירים" שפת SystemVerilog כלומר מבצעים highlight למילות מפתח.

הנחיות כלליות : עליך לענות על כל השאלות ולהכין את הקבצים בהתאם להנחיות.

הכנה ניסוי מספר 1

1. הכנת סביבת הסינתזה

**שאלה 1 :** הסבר במשפט או שניים מה הם קבצי **.lib** וקבצי **.db**.

פתח את כל בקבצים עם סיומת **sv** שהועלו ב- **labadmin**. שים לב שכמעט בכל ביחידות יש חלוקה של הלוגיקה ללוגיקה קומבינטורית (**always\_comb**) ולוגיקה סינכרונית (**always\_ff**).

**שילוב זיכרונות SRAM במימוש :**

ניתן לאכסן מידע בפליפ-פלופים או בזיכרונות **SRAM**. כאשר יש צורך באכסון של כמות גדולה של מידע חייבים להשתמש ב- **SRAM** כי צפיפות הסיביות ב- **SRAM** גבוה בהרבה מצפיפות הסיביות ברגיסטרים (פליפ-פלופים). לא ניתן לסנתז **SRAM** כפי שמסנתזים לוגיקה רגילה. מקבלים את התיאור של היחידה לסימולציות, קובץ שמאפיין את התא, ו"בלוק שחור" שניתן לשלב ישר ב- **layout**.

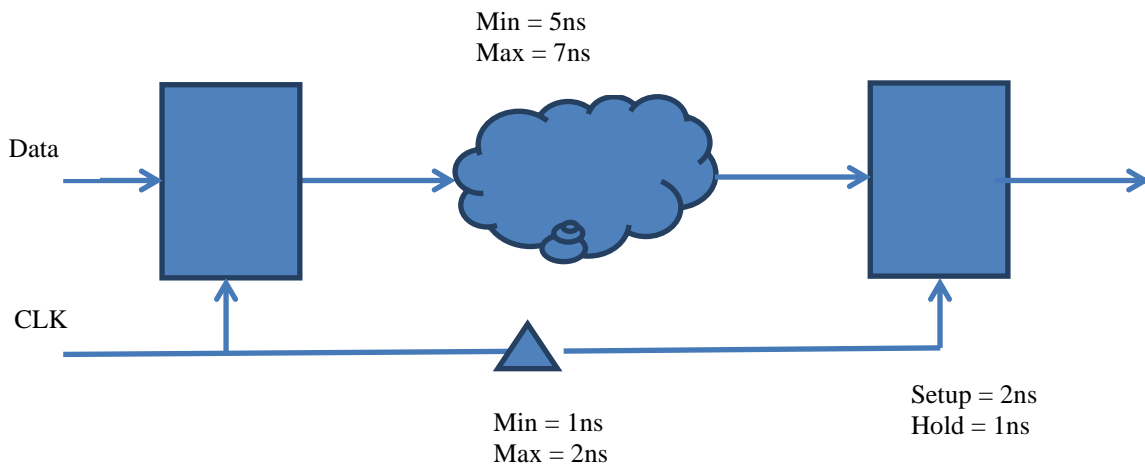
**שאלה 2 :** מה שם המודול של הרמה העליונה של היחידה שלנו? רשום את שמות המודולים שהוא מכיל?

**שאלה 3 :** מדוע לא מופיע מימוש של הזיכרונות ?

**חשוב :** שים לב שביחידת ה- **pooling** קיים **always\_ff** שמופעל עם עלית השעון ואחד שמופעל עם ירידת השעון.

2. אנליזה של **setup time** ו- **hold time**

סעיף זה יעסוק בהעמקת הבנה בנושא **setup time** ו- **hold time**. חשוב להבין שזמני השהיה של שערים נעים בין ערך מינימלי לערך ומקסימלי. חיוני לבצע את האנליזה עם ההשהיות הנכונות (ולא רק עם זמני ה- **typical**). נתון המעגל הבא :



**איור מס' 8 : מעגל סינכרוני טיפוסי**

המעגל מכיל שני **FlipFlops** ולוגיקה קומבינטורית ביניהם. עבור סעיף זה, נזניח את ההשהיות הפנימיות של ה- **FlipFlops** וההשהיות של החוטים.

**שאלה 4 :**

- א. הסבר מה זה setup time ומה זה hold time ?
- ב. מה זה "המסלול הקריטי" ?
- ג. מהו מחזור השעון המינימלי שעדיין מאפשר למעגל לעבוד בצורה תקינה ? הסבר.
- ד. מהו הפתרון פשוט למעגל שאינו מקיים את דרישת ה- setup time ?
- ה. האם המעגל באיור 8 מקיים את דרישת ה- "hold" של ה- FlipFlop ? הסבר.
- ו. מהו הפתרון למסלול שאינו מקיים את דרישת ה- hold time ?

**שאלה 5 :** הכן קובץ מכיל **Top.sdc** ש :

- א. מכיל הגדרת השעון בעל מחזור של 8ns
- ב. מגדיר שהכניסות מגיעות בהשהיה של 1ns אחרי עלית השעון
- ג. מגדיר שהיציאות יוצאות בהשהיה של 1ns אחרי עלית השעון

### 3. שיפור הבדיקות (DFT) : Design For Testability

המטרה בסעיף זה היא להכיר את היכולת של הכלי ה- **DFT** לשפר באופן אוטומטי את הבדיקות של התכנון. מחוסר זמן, התרגיל של ה- **DFT** יתבצע רק על ידידת ה- **pooling**.

**שאלה 6 :**

- הסבר מה זה **scan flipflop**. למה מוסיפים **scan flipflops** וכיצד משתמשים בהם ?
- צייר תרשים של **scan chain** בעל 4 **scan flipflops**.

### 4. Logical Equivalence Checking - LEC

כאמור, תוכנה ה- LEC משוואה בין מימוש ה- RTL ומימוש המסונתז. על מנת שניתן יהיה להשוואת את ה- RTL למימוש מסונתז, הכלי ראשית מבצע סינתזה משלו לקוד ה- RTL לפני ההשוואה.

לכלי, יש להזין את המימושים, פקודות לביצוע ההשוואה ועוד מידע. נבצע זאת עם `command` file בשם `dofile`. להלן מתואר מבנה קובץ ה- `dofile` :

```
set log file enter log file name here -replace
set compare effort low
add notranslate module dpram32x32_cb -both -library

read library -Both -Replace -sensitive -Statetable -Liberty enter all lib files here

read design enter all SV files here -SYS -Golden -continuousassignment
Bidirectional -nokeep_unreach -norangeconstraint -NOE

elaborate design

read design enter synthesized file name here -Verilog -Revised -sensitive
-continuousassignment Bidirectional -nokeep_unreach -nosupply

//Disregard gated clocks as a reason for non-equivalence
set flatten model -gated_clock
```

```
//Make each module unique (i.e. hierA/ModuleA != hierB/ModuleA). A must for hierarchical //compare  
uniquify -all
```

```
//Write Hierarchical dofile with wordlevel (reverse engineering algorithm that takes longer)  
write hier_compare dofile hier.do -replace -prepend_string "analyze datapath -module  
-threads 4 ; analyze datapath -wordlevel -verbose" -usage
```

```
set compare effort low
```

```
//run hier_compare hier.do  
run hier_compare hier.do
```

```
//close the logfile  
set log file
```

#### שאלה 7 :

רשום את הפקודות הנ"ל בקובץ בשם **dofile**. יש לשנות את כל הטקסט שבאדום בשמות של מימוש שבניסוי. בחר שם כלשהו עבור המימוש המסוננת. הקפד להשתמש בשם זה כאשר תשמור את המימוש המסוננת במהלך הניסוי.

#### שאלה 8 :

הסבר מה לפי דעתך התפקיד של הפקודות הבאות (אין צורך להסביר את הארגומנטים של הפקודה). אין צורך להסביר את המשמעות של כל הפרמטרים שבפקודה.

```
set log file enter log file name here -replace
```

```
read library -Both -Replace -sensitive -Statetable -Liberty enter all lib files here
```

```
read design enter all SV files here -SYS -Golden -continuousassignment  
Bidirectional -nokeep_unreach -norangeconstraint -NOE
```

```
read design enter synthesized file name here -Verilog -Revised -sensitive  
-continuousassignment Bidirectional -nokeep_unreach -nosupply
```



בתחילת החלק השני של הניסוי, נסנתז את התכנון מחדש מספר פעמים על מנת לצור תכנון מהיר. נשמור את תוצאת התזמון אחרי כל קומפילציה. להלן מסגרת של קובץ הסינתזה.

### שאלה 1 :

רשום את הפקודות הבאות בקובץ בשם `compile.tcl`. עליך להשלים (מסומן באדום):

1. שמות קבצי התכנון
2. שמות הקבצים לשמירת המעגל המסונתז
3. שמות הקבצים לשמירת תוצאות התזמון

```
set TopModule Top
```

```
set link_library " dw_foundation.sldb \
/tools/kits/tower/PDK_TS18SL/FS120_STD_Cells_0_18um_2005_12/DW_TOWER_ts18fs120/2005.12/synopsys/2004.12/models/ts18fs120_typ.db dpram32x32_cb.db"
```

```
set target_library \
"/tools/kits/tower/PDK_TS18SL/FS120_STD_Cells_0_18um_2005_12/DW_TOWER_ts18fs120/2005.12/synopsys/2004.12/models/ts18fs120_typ.db dpram32x32_cb.db "
```

**חשוב :** סימן ה"\" בשורות `set link_library` ו-`set target_library` **מציין שהמשך הפקודה מופיע בשורה הבא. על השורות בכחול להופיע בשורה אחת ארוכה.**

```
sh mkdir -p WORK
```

```
define_design_lib work -path ./WORK
```

```
read_file -format sverilog {cneuron file name}
read_file -format sverilog {fcneuron.sv file name}
read_file -format sverilog {controller file name}
read_file -format sverilog {pooling file name }
read_file -format sverilog {Top level file name }
```

```
current_design ${TopModule}
```

```
# Read SDC
```

```
source ./${TopModule}.sdc
```

```
current_design ${TopModule}
```

```
set compile_delete_unloaded_sequential_cells true
```

```
set case_analysis_with_logic_constants true
```

```
set case_analysis_with_logic_constants true
```

```
set template_separator_style "_"
```

```
set_register_merging [ get_designs ${TopModule} ] false
```

```
set compile_seqmap_propagate_constants false
```

```
set compile_seqmap_propagate_high_effort false
```

```
# First compile
```

```
compile
```

```
# synt with clock gating - optional
```

```

report_timing > ./reports/file name for first timing results
write -format verilog -hierarchy -output file name for first synthesized file ${TopModule}

#Second compile
compile -incremental
# synt with clock gating - optional
report_timing > ./reports/file name for second timing results
write -format verilog -hierarchy -output file name for second synthesized file ${TopModule}

# Third incremental compile
compile -incremental
report_timing > ./reports/file name for third timing results
write -format verilog -hierarchy -output file name for third synthesized file ${TopModule}

```

עבור שמות הקבצים המסונתזים, עליך להשתמש בשמות top\_syn\_second.v ,top\_syn\_first.v ו- top\_syn\_third.v

## 2. מבנה קובץ ה- LEF

כאמור קבצי ה- **lef** מכילים (בין היתר) את התיאור גיאומטרי (כולל מיקום מדויק של ה- **pins**) של התאים הסטנדרטיים. להלן הפורמט של חלק מהנתונים שמופיעים בקובץ **lef** :

```

01. LAYER layerName
02.     TYPE ROUTING ;
03.     DIRECTION {HORIZONTAL | VERTICAL} ;
04.     PITCH {distance | xDistance yDistance} ;
05.     WIDTH defaultWidth ;
06.     OFFSET {distance | xDistance yDistance} ;
07. END layerName

01. MACRO macroName
02.     CLASS {PAD | CORE} ;
03.     ORIGIN point ;
04.     SIZE width BY height ;
05.     SITE siteName ;
06.     PIN pinName DIRECTION {INPUT | OUTPUT}
07.     PORT
08.     LAYER layerName ;
09.     RECT point point ;
10.     END
11.     END pinName
12. END macroName

```

**שאלה 2** : רשום לפי דעתך :

- א. מה מתארים המבנים **LAYER** ו- **MACRO** שמופיעים לעיל ?
- ב. כמה מבנים מסוג **LAYER** ו- **MACRO** יופיעו בקובץ המלא ?
- ג. כמה משפטי **PIN** יופיעו בכל מבנה **MACRO** ?

## 3. קובץ mmmc.view וקובץ אילוצי תזמון

**שאלה 3** :

לפי איור 7 :

- א. אלו קבצים דרושים להגדרת ה- **library set** מסוג **max** ו- **min** ?
- ב. איזה **library set** דרוש להגדרת **delay corner** מסוג **SlowDC** ?

- ג. מה זה קובץ ה-**Top.sdc**. איזה **MMMC Object** משתמש בו ?  
ד. כמה **Analysis Views** קיימים בהגדרת ה-**mmmc.view** שלנו ?

#### 4. שלבי מימוש ה- layout

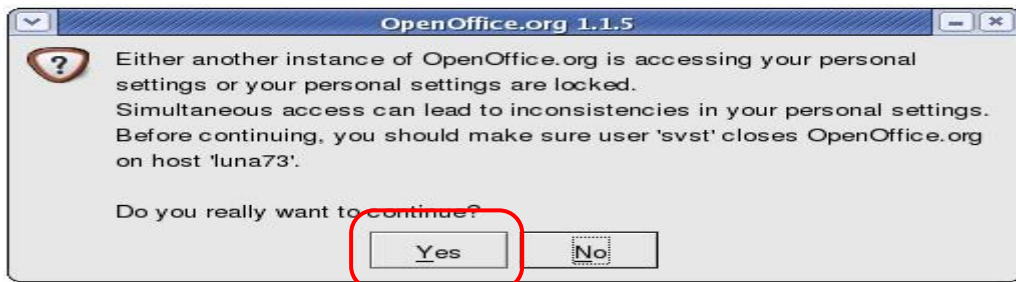
##### שאלה 4 :

- א. אלו קבצי הטכנולוגיה הדרושים למימוש ה- **layout** ?  
ב. אלו פרמטרים עליך להגדיר על מנת לממש עץ שעון ?  
ג. רשום את כל השלבים העיקריים במימוש ה- **layout**. תאר כל שלב במשפט אחד.

הערות :

כתיבת דוח הניסוי תבצע במהלך הניסוי באופן ממוחשב בעזרת תבנית שברשותכם. כל הגרפים והסכמות ישולבו בדו"ח זה ללא הצורך בהדפסות. לעריכת הדו"ח רשום :

oowrite name1\_name2.odt  
 כאשר name1 ו-name2 הם שמות הסטודנטים. שים לב ש- name1\_name2.odt אינו קובץ ריק אלא תבנית עליה מתבסס הדו"ח.  
 עריכת הדוח הממוחשב תעשה ע"י מעבד תמלילים של Open Office. יתכן ויפתח חלון ההערה הבא :



לחץ על Yes בכדי להמשיך. בדוק שנפתח קובץ לפי שמות הסטודנטים שלכם.

**צירוף תמונות/סכמות לדו"ח :**

- ראשית יש צורך לצור קובץ .jpeg
- הפעל את תוכנת ksnapshot בחלון terminal. לחץ על New Snapshot.
- בחלון שנפתח בחר ב- Region עבור ה- Capture mode
- באמצעות הכפתור השמאלי בחר באיזור של המסך שברצונך להוסיף לדו"ח.



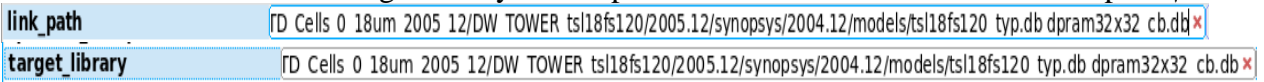
- לחץ על Copy to Clipboard וצרף למסמך עם CNTR V.
- ניתן גם לשמור את התמונה באמצעות Save As בפורמט jpg ולהכניס את הסכמה לדו"ח ב- openoffice עם Insert->Picture->From File.
- רצוי להקטין את חלון ה- waveform כדי שהוא יכנס בצורה יפה יותר לדו"ח.
- **שינוי שפה** יעשה ע"י Alt+Shift ושינוי כיוון כתיבה ע"י Ctrl+Shift.
- **שינוי כיוון הטקסט** יעשה ע"י : Ctrl+Shift+A ו- Ctrl+Shift+D.
- **הערה** : סעיף לביצוע מופיע כ- "–", סעיף לשמירת גרף מופיע כ- 'G111' וסעיף תשובה לשאלה מופיע כ-'Q111'. מומלץ לשמור את התמונות בהתאם למספר הסעיף, למשל G111

1. הכנת סביבת הסינתזה

מטרת סעיף זה היא ללמוד כלי הסינתזה והנושא של Logical Equivalence Checking. במהלך הניסוי, נכיר לא רק את הכלים אלא גם קבצי הטכנולוגיה שכל כלי דורש על מנת שיוכל לעבוד. פתח את הקובץ compile1.tcl עם העורך nedit:

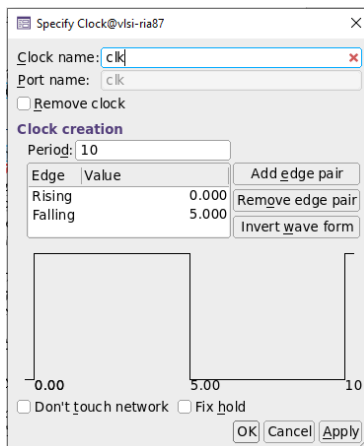
nedit compile1.tcl

- Q11 : הסבר בקיצור את תוכן הקובץ.
- הפעל את ה- design vision בעזרת הפקודה .dv
- בעזרת הפקודה File->Execute קרא את הקובץ compile1.tcl ולחץ על Open בחלון שנפתח.
- Q12 : רשום את ה- Warnings שמתקבלים.
- בחר ב- Top בחלון Logical Hierarchy בצד שמאל. בצע Design->Compile Design.
- Q13 : רשום את ה- warnings/errors ואת הודעת השגיאה שמתקבלים.
- Q14 : מה חסר ? בצע File->Setup ורשום בדו"ח משמופיע בשדות : link path ו- target library.
- נבצע את התיקון הנדרש. על מנת לבצע את הסינתזה חובה להגדיר את הטכנולוגיה שבה אנו משתמשים.
- בצע File->Exit מה- design\_vision.
- פתח את הקובץ libs.tcl. רשום את התוכן שלו בדו"ח.
- הפעל את ה- design vision בעזרת הפקודה .dv
- בעזרת הפקודה File->Execute קרא את הקובץ libs.tcl ולחץ על Open בחלון שנפתח.
- לחץ על File->Setup ושים לב לשינויים בשדות link path ו- target library.



- בעזרת הפקודה File->Execute קרא את הקובץ compile1.tcl ולחץ על Open בחלון שנפתח.
- בצע Design->Compile Design ולחץ על O.K.
- Q15 : האם הפקודה מצליחה לרוץ הפעם? הסבר מדוע הודעות השגיאה מהסעיף הקודם נעלמו? מדוע יש צורך בקבצים שהוספנו בסעיף הקודם ?
- ננסה למצוא את המסלול הקריטי. בצע Timing->Report Timing Path.
- Q16 : רשום את נקודת ההתחלה, הסיום וההשהיה של המסלול הקריטי שמתקבל. האם זה הגיוני? רשום את ההודעה שמתקבלת אחרי המסלול. מה הפרוש ? עליך לקרוא למנחה להסביר לו את התשובות.

- נוסף הגדרת שעון. כעת בחלון ה- Logical Hierarchy בחר ב- Top ולחץ על ה- icon
- בחר את pin השעון ולחץ על Attributes->Specify Clock.
- בחלון שנפתח נגדיר את השעון באופן הבא :



- בצע Timing->Report Timing Path.
- Q17 : רשום את נקודת ההתחלה, הסיום וההשהיה של המסלול הקריטי שמתקבל. האם התכנון עומד בזמנים ?
- בצע Design->Compile Design ולחץ על O.K.
- בצע Timing-Report Timing Path.
- Q18 : רשום את נקודת ההתחלה, הסיום וההשהיה של המסלול הקריטי שמתקבל. האם התכנון עומד בזמנים ? הסבר מדוע (גם למנחה).
- Q19 : מדוע הפעם התוצאה שונה מהפעם הקודמת?
- סגור את ה- design\_vision עם File->Exit ו- OK.

**הערה :** בסעיף זה נדגים כיצד שימוש של האופציות השונות מאפשרות ל- design vision לסנתז מימושים בעלי ביצועים משופרים.

- נתחיל מחדש את dv עם כל קבצי ה- lib ואילוצי התזמון מוגדרים מראש ב- compile2.tcl
- פתח את הקובץ compile2.tcl עם העורך nedit :
- nedit compile2.tcl

Q21 : הסבר בקיצור את תוכן הקובץ.

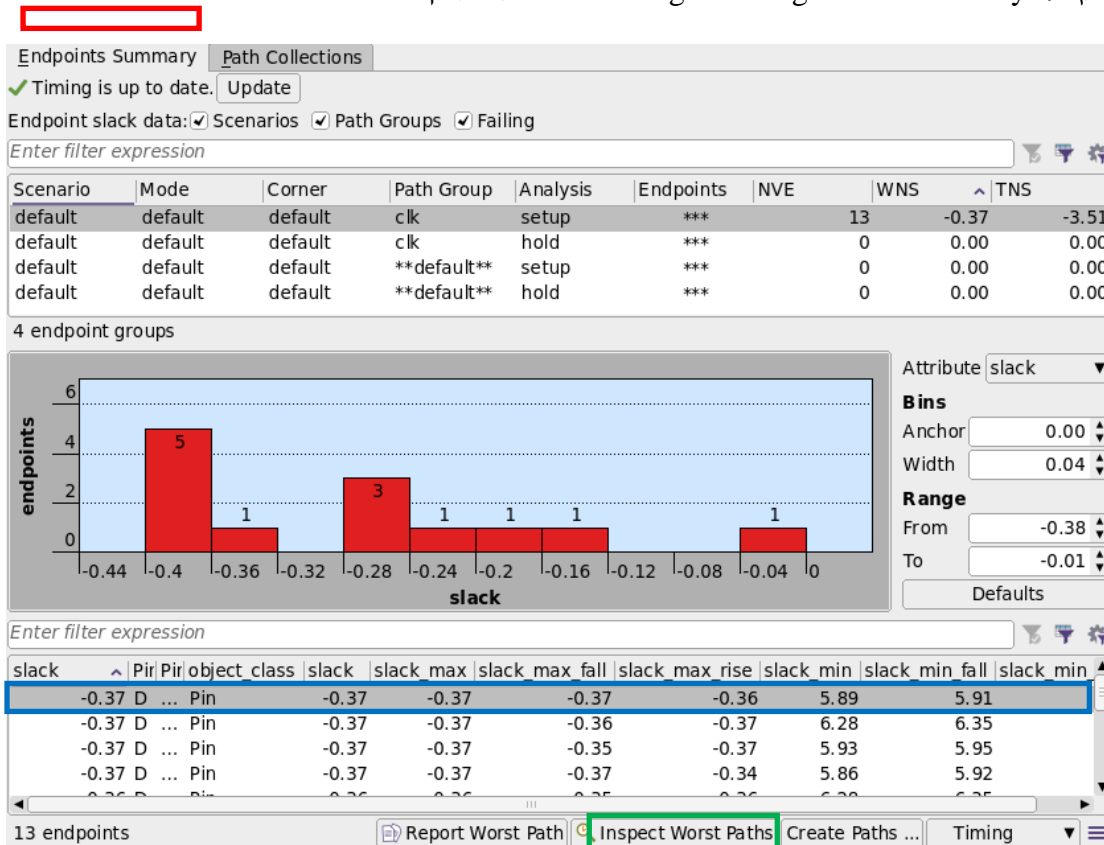
- פתח את ה- .dv בעזרת הפקודה File->Execute קרא את הקובץ compile2.tcl ולחץ על Open בחלון שנפתח.

- בצע Design->Compile Design.

Q22 : בצע Timing->Report Timing. רשום את ההשהיה של המסלול הקריטי בטבלה שמופיעה בעמוד הבא.

Q23 : בדוק את שטח וצריכת ההספק (Design->Report Area ו- Design->Report Power). כמה הספק צורך הזיכרון ? רשום את התוצאות של השטח ושל צריכת ההספק בטבלה (עבור ההספק רשום את ההספק ללא צריכת ההספק של הזיכרון).

- לחץ על Timing->Timing Status Summary . יפתח החלון הבא :



- יש לוודא שתוית ה- Endpoints Summary לחוץ. בחר במסלול הראשון (מלבן כחול) ולחץ Inspect Worst Paths (מלבן ירוק).

Q24 : הוסף את הסכמה לדו"ח. בחלון שנפתח, בחר ב- על Path Schematic. פעולה זאת מציגה הסכמה של המסלול הקריטי.

Q25 : רשום בדו"ח את כל השערים שנמצאים על המסלול הקריטי. ניתן לראות אותם בחלון ה- Report Timing.

א. **הסבר למנחה** ורשום בדו"ח מדוע הכלי מציין שנקודת הזמן של תחילת המסלול הוא 4ns (ולא 0ns כמצופה)?

ב. מה ההשהיה של המסלול הקריטי (כפי שמדו"ח עלי ידי הכלי) אם מקזזים ממנו 4ns?

ג. האם יתכנו מסלולים במעגל שההשהיה שלהם גדולה מהזמן שקיבלת בסעיף ב? הסבר את תשובתך למנחה. רשום את ההסבר בדו"ח.

- בצע שוב Design->Compile Design **הפעם עם ungroup דלוק**, ולחץ על O.K. אופצית ה- ungroup "משטח" את התכנון ומאפשר אופטימיזציות בין הבלוקים שבתכנון.  
Q26 : בדוק את התזמון כפי שהוסבר בסעיף הקודם. רשום את ההשהיה של המסלול הקריטי הטבלה. בדוק את שטח והספק (Design->Report Area ו- Design->Report Power) ורשום את התוצאות של השטח ושל צריכת ההספק. האם התכנון עומד בזמנים ?

סגור את ה- design\_vision. פתוח אותו מחדש ובעזרת הפקודה File->Execute קרא את הקובץ compile2.tcl ולחץ על Open בחלון שנפתח.

- בצע Design->Compile Ultra, ולחץ על O.K.  
Q27 : בצע Timing->Report Timing. רשום בטבלה את ההשהיה של המסלול הקריטי שמתקבל. האם התכנון עומד בזמנים ? הסבר. בדוק את שטח והספק (Design->Report Area ו- Design->Report Power) ורשום את התוצאות בטבלה. **שים לב שהפעם הכלי מדווח על ההספק ללא צריכת ההספק של הזיכרון.**

- בתוך הכלי בשורת הפקודות למטה הכנס את הפקודה הבאה :  
- report\_net\_fanout -threshold 10  
- פקודה זו מציגה את כל הרשתות בעלות fanout גדול מ-10.  
- רשום

- set\_max\_fanout 10 Top  
- פקודה זו מאלצת את כלי הסינתזה להגביל את ה- fanout ל-10.  
- בצע Design->Compile Ultra.

Q28 : בצע Timing->Report Timing. רשום בטבלה את ההשהיה של המסלול הקריטי שמתקבל. האם התכנון עומד בזמנים ? הסבר. בדוק את שטח והספק (Design->Report Area ו- Design->Report Power) ורשום את התוצאות בטבלה.

Q29 : מה זה fanout של שער ? האם התכנון עומד בדרישות ה- fanout ? האם הכלי תמיד מסוגל לעמוד בכל דרישות המשתמש ? הסבר.

- השעון שמוגדר ב- compile2.tcl הוא בעל מחזור של 8ns. הגדר שעון חדש בעל מחזור של 6ns.  
- וגם בצע :

- set\_max\_fanout 20 Top  
- בצע Compile-Ultra

Q210 : בצע Timing->Report Timing. רשום בטבלה את ההשהיה של המסלול הקריטי שמתקבל. האם התכנון עומד בזמנים ? הסבר. בדוק את שטח והספק (Design->Report Area ו- Design->Report Power) ורשום את התוצאות בטבלה.

Q211 : האם התכנון עומד בזמנים ? האם התכנון עומד בדרישות ה- fanout ? הסבר.

	Critical Path	Area	Power
compile			
With ungroup			
Compile ultra			
10 max_fanout			
6ns clk + 20 fanout			

- צא מהכלי.

Q212 : סכם את כל התוצאות שהתקבלו בטבלה.  
 פתוח את ה- design\_vision מחדש ובעזרת הפקודה File->Execute קרא את הקובץ  
 compile3.tcl ולחץ על Open בחלון שנפתח.  
**חשוב : הסקריפט יוצר את המעגל המסוננת לחלק שני של הניסוי בקובץ בשם top\_syn.v.**


**3. שיפור הבדיקתיות : Design For Testability (DFT)**

המטרה בסעיף זה היא להכיר את היכולת של ה- **design\_compiler** לשפר באופן אוטומטי את הבדיקתיות של התכנון. מקובל לבצע את התהליך הזה על המימוש המלא. מחוסר זמן נבצע את התרגיל על תת הבלוק בשם **pooling**.

- המימוש מופיע בקובץ **pooling.sv**.
- הפעל את ה- design vision בעזרת הפקודה .dv
- בעזרת הפקודה File->Execute קרא את הקובץ compilePool.tcl ולחץ על Open בחלון שנפתח. בצע את הפקודות :

- **create\_clock -name "clk" -period 6 -waveform {0 3} {clk}**
- **set\_output\_delay -max 0 pooledPixels**
- בצע סינתזה של המעגל ע"י בחירת Design->Compile Design.

Q31 : בצע Timing-Report Timing. רשום בטבלה את ההשהיה של המסלול הקריטי שמתקבל. האם התכנון עומד בזמנים ? רשום את התוצאות בטבלה. בדוק את שטח והספק (Design->Report Area ו- Design->Report Power) ורשום את התוצאות בטבלה.


Q32 : בחר ביחידה **pooling** בצד שמאל. לחץ על . כדי לראות את הסכמה לחץ שתי לחיצות מהירות על היחידת ה- pooling שמופיעה בחלון. בצע zoom לפינה הימנית העליונה של הסכמה.

- א. הוסף לדו"ח צילום מסך של הסכמה שמכיל כ- 4 רגיסטרים.
- ב. האם קיים קשר בין היציאה של רגיסטר אחד לכניסה של רגיסטר אחר ? הסבר.
- חזור לרמה העליונה של הסכמה.
- בחלון הפקודות רשום :

- **source dft.src**
- ב- **dft.src** רשומות שתי פקודות עם הגדרות עבור ה- **scan insertion**.
- בחלון הפקודות רשום :
- **insert\_dft**

Q33 : בצע שוב Timing-Report Timing. רשום בטבלה את ההשהיה של המסלול הקריטי שמתקבל. האם התכנון עומד בזמנים ? רשום את התוצאות בטבלה. בדוק את שטח והספק (Design->Report Area ו- Design->Report Power) ורשום את התוצאות בטבלה.

	Area	Critical Path	Power
Before DFT			
After DFT			

Q34 : רשום כיצד **insert\_dft** השפיע על ערכים שבטבלה.  
 Q35 : בחר ביחידה **pooling** בצד שמאל. לחץ על . כדי לראות את הסכמה לחץ שתי לחיצות מהירות על היחידת ה- pooling שמופיעה בחלון. בצע zoom לחלק השמאלי של הסכמה.

- א. הוסף לדו"ח את החלק של הסכמה שמכיל כ- 4 רגיסטרים.
- ב. האם קיים קשר בין היציאה של רגיסטר אחד לכניסה של רגיסטר אחר ? הסבר.
- ג. רשום את כל השינויים שהוכנסו למעגל כתוצאה של הפקודה **insert\_dft**. התייחס גם לכניסות ויציאות החדשות.

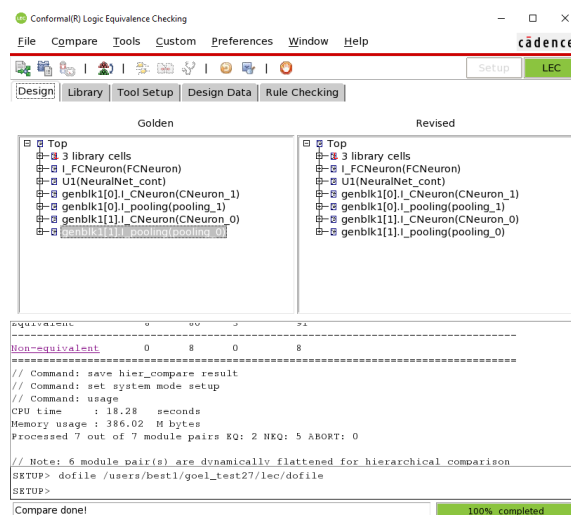


- בצע את הפקודה :
- `write_scan_def -output scandef`
- פקודה זאת יוצרת קובץ בשם `scandef` המכיל את כל הנתונים של ה-`scan chains`. פתח את הקובץ ובחן את תוכנו ורשום בדו"ח את מספר השרשרות שנבנו, נקודת ההתחלה שלהם ונקודת הסיום.
- Q36 : כמה `scan chains` נוספו למעגל ? מדוע ? איך אתה יודע ? הסבר את התפקיד של כל הכניסות והיציאות החדשות במעגל.
- אם ברצוננו לשנות את שיטת הפעולה של הפקודה `insert_dft` יש להשתמש בפקודה `.set_scan_configuration`.
- ניתן לקבל הסבר מפורט על `configuration_scan_set` על ידי רישום :  
- `man set_scan_configuration`
- בחלון הפקודות.
- Q37 : בעזרת ההסבר רשום בדו"ח פקודה שיגרום ל-`dft_insert` להכניס 3 `chains_scan`
- סגור את ה-`dv` עם `File->Close`.

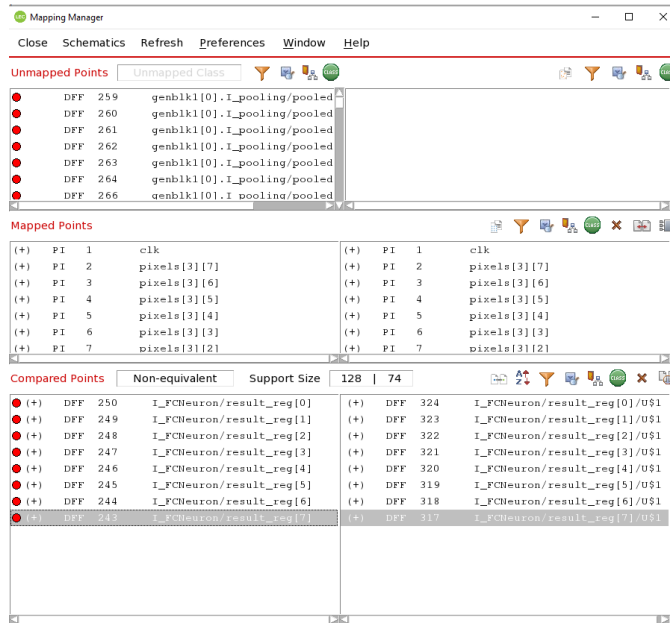
## LEC .4

לאחר ביצוע סינתזה חובה לוודא שהמעגל המסונתז זהה מבחינה לוגית לתיאור ה-RTL. זאת המטרה של סעיף זה.

- חזור לחלון הפקודות של ה-LINUX
- עבור לתיקיה `lec` עם :  
- `cd lec`
- ראשית נבצע שוב את הסינתזה. בצע סינתזה של המעגל עם `compileLec.tcl`. הסינתזה יוצרת את המעגל מסונתז : `top_syn_first.v`
- חשוב : שם לב שבפלט את ה-`design_vision` יש הודעות מסוג :  
the register 'genblk1[1].I\_pooling/pooledReg\_reg[0][7]' is removed because it is merged to 'genblk1[1].I\_pooling/pooledReg\_reg[0][1]
- זה חשוב להמשך !
- **בחלון נפרד** הפעל את הכלי עם : `lec -CCDXL`
- בצע `File->Do Dofile` ובחר בקובץ בשם `dofile`. לחץ על `Select`.
- Q41 : האם שני התכנונים שקולים ?



- בחלון Conformal – Logical Equivalence Checking דפדף עד שמופיע :
- Non-equivalent
- ולחץ עליו. מופיע החלון הבא :



- ניתן לראות שקיימים flipflops בתכנון המקורי שאינם קיימים במעגל המסונתז.
- Q42 : מדוע לפי דעתך קיימת אי התאמה כזאת? זכור את ההערה שהפיק ה- design\_vision. רשום את התשובה בדו"ח והסבר את התשובה למדריך.
- פתח את הקובץ compileLecNoMerge.tcl. שם לב, שלפני פקודת הסנתזה compile\_ultra מופיעה השורה :  
set\_register\_merging \* false
- Q43 : רשום את המשמעות של הפקודה בדו"ח והסבר את התשובה למדריך.  
- סגור את חלון ה- dv . פתח את ה- dv מחדש.  
- נסנתז מחדש עם : compileLecNoMerge.tcl  
- כעת נבצע את ההשוואה מחדש. בחלון של ה- lec בצע File->Reset Design.  
- בצע שוב File->Do Dofile ובחר בקובץ בשם dofile. לחץ על OK.
- Q44 : האם שני התכנונים שקולים ? האם לפי דעתך זה פתרון טוב לבעיה ?  
- פתח את קובץ ה- pooling.sv. ניתן לראות את השורה הבאה :  
- pooledReg[i] <= (\$signed(convolution[i]) > 2) ? 8'b1 : 8'hff;  
- כלומר pooledReg[i] שומר רק +1 או -1 במשתנה של 8 סיביות. קצת בזבזני. כלי הסנתזה מזהה זאת ומבצע אופטימיזציה.
- המימוש poolingOpt.sv מהווה מימוש משופר שבו pooledReg[i] הוא משתנה של סיבית אחת בלבד. נבצע כעת את כל התהליך מחדש עם המימוש המשופר.  
- סגור את חלון ה- dv . פתח את ה- dv מחדש.  
- בצע סינתזה של המעגל עם compileLecOpt.tcl. הסינתזה יוצרת את המעגל מסונתז : top\_syn\_second.v
- חשוב : שם לב שבפלט את ה- design\_vision כבר אין הודעות מסוג :  
the register 'genblk1[1].I\_pooling/pooledReg\_reg[0][7]' is removed because it is merged to 'genblk1[1].I\_pooling/pooledReg\_reg[0][1]'  
- כעת נבצע את ההשוואה מחדש. בחלון של ה- lec בצע File->Reset Design.  
- בצע שוב File->Do Dofile ובחר בקובץ בשם dofileOpt. לחץ על OK.
- Q45 : האם שני התכנונים שקולים ? הסבר בדו"ח וגם למנחה מדוע פתרון זה טוב בהרבה מהקודם.
- סגור את ה- design\_vision וה- lec.

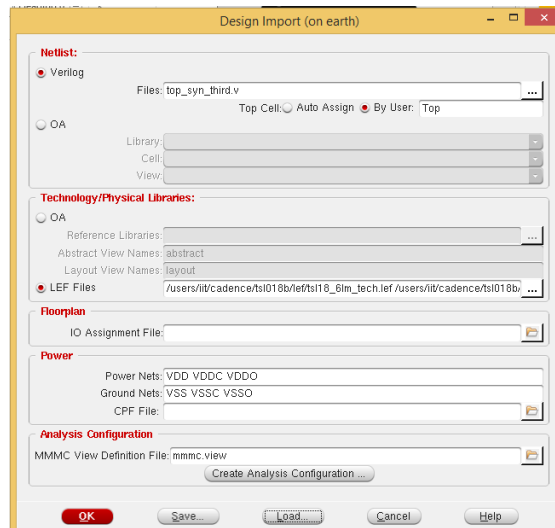
**סיום חלק ראשון !**


בפגישה זאת אנו נבנה את ה- **layout** של התכנון בעזרת כלי אוטומטי של חברת **Cadence** בשם **Innovus**. כזכור, התכנון שלנו מורכב מתאים סטנדרטיים ומשלש יחידות **SRAM**.

כזכור, בחלק הראשון יצרנו קובץ מסונתז בשם **top\_syn.v**.

## 1. תכנון ה- Floorplan

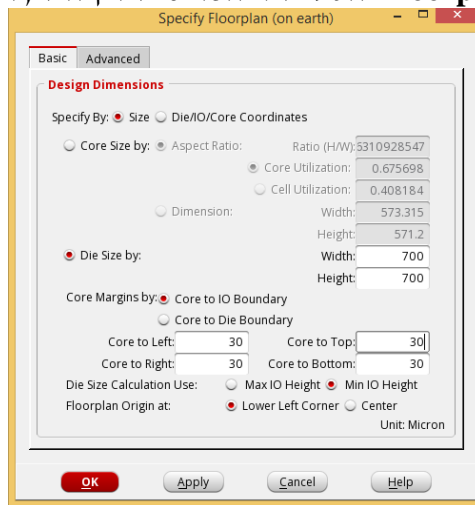
- בשלב ראשון, עלינו לקרוא את נתוני הטכנולוגיה והתכנון (קובץ ה- **Verilog** שהתקבל כפלט מהסינתזה. על מנת להקל על התהליך נבצע זאת באופן הבא :
- קווי החיווט ב- **layout** מוסיפות קבלים ונגדים פרוזיטיים ולכן יהיה בלתי אפשרי לעמוד במחזור שעון של **8ns** כמו בסעיף הקודם.
  - שנה את מחזור השעון ב- **Top.sdc** ל- **12ns**.
  - הפעל את כלי ה- **layout** בעזרת הפקודה **innovus**.
  - בצע **File->Import Design**. החלון הבא ייפתח :



- בשלב ראשון, כל השדות ריקים. לחץ על **Load** בחר בקובץ **Top1.globals**.
- בדוק את שם קובץ ה- **Verilog** (בשדה **Files**) ואת שם המודול העליון. תקן את השמות אם יש בזה צורך. לחץ על **OK**.
- כאמור קובץ ה- **mmmc.view** קורא לקובץ **Top.sdc** (אילוץ התזמון)
- קבצי ה- **lef** מכילים את התיאור גיאומטרי (כולל מיקום מדויק של ה- **pins**) של התאים הסטנדרטיים.
- לחץ על **OK**. אם אין שגיאות הכלי יעלה את התכנון ואת כל קבצי הטכנולוגיה הדרושים.
- לחץ על כל אחת מהאפשרויות מבאות (בצד ימין) :  על מנת לראות ייצוגים שונים של המעגל.
- במרכז ניתן לראות את האזור שיכיל את התאים הסטנדרטיים ובצד שמאל את הבלוקים השונים.
- Q11 : מה חסר ? עבור לחלון ה- **console**. חפש הערה בנושא ה- **dpram32x32\_cb** ? מה היא ?
- Q12 : מדוע לפי דעתך הזיכרונות אינם מופיעים ? כזכור, על מנת לשלב מודול ל- **layout** יש לספק עבורו את קבצי ה- **lef** ו- **lib**. האם מידע זה עבור ה- **dpram32x32\_cb** סופק לכלי ? אם כן, איזה קובץ סופק ואיפה בדיוק ?
- קרא למדריך על מנת לקבל הסבר על מה שחסר.
- צא מהכלי עם **File->Exit**.
- הפעל את הכלי מחדש. קרא את התכנון, אבל הפעם יש להשתמש ב- **Top2.globals**.

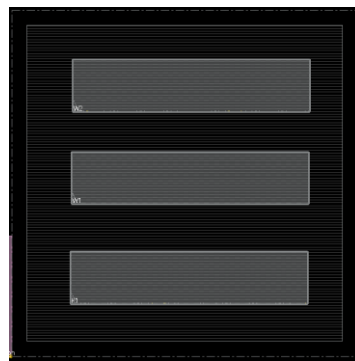
Q13 : מה שונה לעומת **Top1.globals** ?  
 השלב הראשון של התהליך הוא לבנות את ה- **Floorplan** ולמקם את הבלוקים המוכנים (הזיכרונות במקרה שלנו).

- בצע **Floorplan->Specify**. השלם את הטופס באופן הבא, ולחץ על **OK** ?



Q14 : הסבר את המשמעות את השינויים שעשית בטופס.

- לחץ על  בצד שמאל ומקם את יחידות ה- **RAM** כפי שמופיע באיור הבא :



- ליחידות הזיכרון מתחברים חוטים רבים. על מנת לאפשר לכלי החיווט גישה נוחה, נגדיר שטח מסבב לזיכרונות שבו לא ניתן למקם תאים סטנדרטיים.
  - פעולה זאת מתבצעת עם **Floorplan->Edit Floorplan->Edit Halo**.
  - רשום 20 עבור **Top, Bottom, Left, Right** ולחץ על הכפתור **All Macros**.
  - לחץ על  כדי לראות את התוצאה.
- Q15 : הוסף את הסכמה לדו"ח.

## 2. מיקום הפינים והגדרת רשתות האספקה

**קביעת מיקום ה- pins :**

בד"כ נרצה למקם את הפינים בצלע שהכי קרובה לבלוק הסמוך שאו - שולח את המידע או - מקבל אותו. וזאת כדי לשמור על קווי חיבור קצרים ככל שניתן. נוכל להשתמש בכל אחת משכבות המתכת הקיימות. למשל כאן יש לנו 6 שכבות. אבל בניסוי היום, נפזר את הפינים בכל הצלעות בלי להתייחס לאן הם מתחברים.

- כעת יש לחלק את ה- **pins** בצורה שווה פחות או יותר בין הצלעות. יש לשים לב שמדובר ב- **pins של הבלוק פנימי ולא ב- pins של שבב שתפקידם לחבר את השבב לעולם החיצוני**. התכנון שלנו כולל 7 **buses** של 32 סיביות, **bus** אחד של 8 סיביות ועוד כמה

פינים. הרעיון כאן הוא לחלק את ה- pins באופן שווה בין הצלעות ולהרחיק אותם אחד מהשני בצורה אחידה.  
 לשם כך נשתמש בכלי שנקרא Pin Editor, שמאפשר לבצע את כל הפעולות ולהגדיר את כל הפרטים.

- לחץ על **Edit->Pin Editor**
  - בעזרת **ctrl** בחר ב- **classify, clk, learn, pixels, result, rst**
  - לחץ על הכפתור שליד **Spread** ובחר ב- **Along Entire Edge**
  - עבור **Side/Edge** בחר ב- **Top**.
  - לחץ על **OK**. ראה את מיקום הפינים על ה- **layout**.
  - חזור על הפעולה עבור יתר הצלעות לפי בסידור הבא :
  - שמאל : **KIDATA1, KIDATA2**
  - ימין : **W1DATA1, W1DATA2**
  - למטה : **W2DATA1, W2DATA2**
  - סגור את החלון של ה- **Pin Editor**
- Q21: כיצד ניתן לבחור באיזה שכבה ימוקמו הפינים? על איזה layer מופיעים ה- pins שמוקמו ?  
 Q22 : הוסף את הסכמה לדו"ח.

**הגדרת רשתות האספקה**

בשלב זה נדרש להשתמש בחוטים עבים כאשר הזרם גבוה ובחוטים דקים יותר כאשר הזרם נמוך. לשם כך נבנה שתי טבעות (עבור **VDD** ו- **VSS**) של חוטים עבים מסביב לבלוק שלנו. בנוסף נוסיף עוד כמה חוטים עבים שחוצים את התכנון באופן אנכי. החיבור של כל התאים לרשתות האלו יתבצע באמצעות חוטים דקים.  
 את הטבעות ואת החוטים האנכיים ניתן לממש בעזרת שתי פקודות :

- א. **Power->Power Planning->Add Ring**
- ב. **Power->Power Planning->Add Stripes**

בעזרת **Power->Power Planning->Add Rings** נוסיף טבעות אספקה סביב לבלוק. בחלון שנפתח, הגדר איזה טבעות רצויות ובחר :

- ב- **Nets** בחר **VDD** ו- **VSS** בלבד.
- ב- **Ring Configuration** :
- הצלעות **M5 : left ,right**
- הצלעות **TOP\_M : top ,bottom**
- רוחב הפס : 6
- מרחק בין הפסים : 1.8
- ליד **offset** לחץ על **Centre of Channel**.
- בסוף לחץ על **OK**. בדוק שנוספו טבעות עבור **VDD** ו- **VSS**.

בעזרת **Power->Power Planning->Add Stripe** ניתן להוסיף רצועות נוספות של קווי האספקה. השלם את הטופס באופן שתואם את הטופס של הטבעות :

- בחר **Nets** בחר ב- **VDD** ו- **VSS** בלבד.
  - הצלעות האנכיות : **M5**
  - בדרך כלל נגדיר רק צלעות אנכיות או צלעות אופקיות
  - רוחב הפס : 6
  - מרחק בין הפסים : 1.8
  - עבור **Set to set distance** רשום 100
  - ב- **First/Last Strip** לחץ על **Absolute**. עבור **Start** רשום 80 ועבור **Stop** רשום 600
- Q23 : הסבר את המשמעות של המספרים בסעיף הקודם.  
 לחץ על **OK**. בדוק שנוספו קוים אנכיים עבור **VDD** ו- **VSS**.
- Q24 : הוסף את הסכמה לדו"ח.

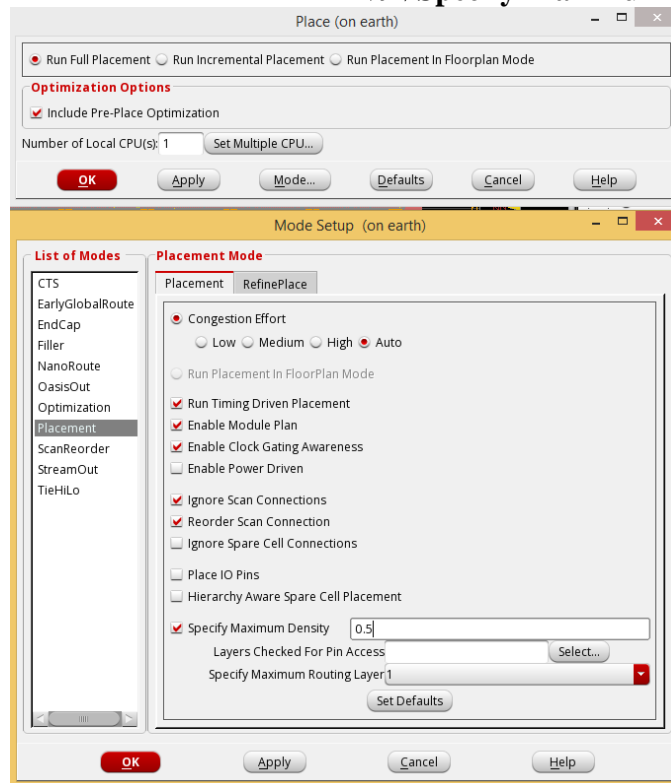
כלי ה- innovus יוצר קובץ ב- **directory** שבו הינכם עובדים בשם **innovus.cmdN** כאשר N הוא מספר. פתח את הקובץ עם ה- N הגבוה ביותר. קובץ זה מכיל את רשימת כל הפקודות שבוצעו עד כה באמצעות הממשק הגרפי. עבור לסוף הקובץ.

Q25 : העתק והוסף לדו"ח את הפקודה שהוסיפה את הטבעות של רשתות האספקה.  
 סגור את הקובץ. שים לב שבעזרת קובץ זה ניתן לבנות script files בקלות רבה!


### 3. מיקום התאים הסטנדרטיים ובנית עץ השעון (Clock Tree Synthesis CTS)

שים לב שעדיין לא מיקמנו את התאים הסטנדרטיים. בשלב זה, נמקם את כל התאים הסטנדרטיים שמממשים את המעגל. על מנת למנוע מיקום של תאים מתחת לקווי VDD ו-VSS לחץ על **Place->Specify->Placement Blockage**. בחר ב-M3 עד M6.

כעת לחץ על **Place ->Place Standard Cell**. בחלון שנפתח, לחץ על **Mode**. נרצה להגדיר את הצפיפות של התאים. נבקש צפיפות שלא עולה על 50% על מנת להשאיר מספיק מקום לחיווט. בשדה של **Specify Maximum Density** רשום 0.5 .



לחץ על **OK** ושוב על **OK**.

Q31 : לחץ על  שבצד ימין אל מנת לראות את תוצאת המיקום. בחר בתא (לא ה-SRAM), לחץ על הכפתור הימני ובחר ב- **Attribute Editor**. מה ה- **status** ? מה המשמעות של **status** זה ?

Q32 : חזור על הפעולה עבור חוט. מה ה- **wire status** ? מה המשמעות של **status** זה ?  
עליך להסביר למנחה את התשובה שלך!

מכאן ניתן להסיק שהחיווט שמוצג אינו חיווט אמיתי/סופי אלא רק חיווט אפשרי. הכלי INNOVUS מאפשר לנו לבחור איזה שכבות נרצה להציג על המסך כדי שנוכל להתרכז רק בשכבות מסוימות. בצד ימין למטה ישנה רשימה של כל השכבות ומשבצת בחירה. נכבה את הצגת החוטים לחיצה על 1,2,3,4,5,6. בצע **zoom** והכנס לדו"ח תמונה של חלק קטן של ה- **layout**. שים לב שהיחס בין השטח הריק והמלא בתאים הוא בערך 1:1. בהתאם למה שביקשנו בשלב המיקום. חזור את כל חוטים שוב עי"י לחיצה על 1,2,3,4,5,6.  
 Q33 : הוסף את הסכמה לדו"ח ללא המתכות.

בצע **(Pre-CTS) Timing->Report Timing**. מה ההשחיה של המסלול הקריטי ?  
 בצע את הבדיקה פעם עבור **Setup** ופעם נוספת עבור **Hold**.

רשום את כל תוצאות ה- **timing** לסעיף זה וליתר הסעיפים בטבלה.

בצע :

optDesign -preCTS

יש לרשום את הפקודה בחלון ה- **console** כלומר החלון שממנו הופעל **Innovus**.  
**הערה : חשוב ביותר להפעיל את פקודות האופטימיזציה ובדיקות התזמון עם האופציות הנכונות אחרת תתקבלנה תוצאות לא נכונות!**

פקודה זאת מבצעת אופטימיזציה של התכנון. שים לב שזה לוקח קצת זמן ☹️.  
מה ההשגיה של המסלול הקריטי עכשיו? האם יש שיפור ?  
מכאן החשיבות להריץ את פקודת אופטימיזציה ! רשום את התוצאות בטבלה :

Clk = 12n	Setup WNS	Setup TNS	Hold WNS	Hold TNS
Pre-CTS before OptDesign				
Pre-CTS after OptDesign				
Post-CTS before OptDesign				
Post-CTS after OptDesign				
Post-Route before OptDesign				
Post-Route (hold) after OptDesign				

**הערה :** כעת ניתן למלא רק שתי שורות. את יתר השורות נמלא בהמשך.

### הגדרת עץ השעון

תפקיד עץ השעון הוא להבטיח שהשעון מגיע לכל הפליפי פלופים באותו זמן פחות או יותר.  
עליך להריץ את הפקודות הבאות ב- **terminal** שבו הפעלת את **innovus**.  
ראשית נגדיר את סיגנל השעון :

```
create_ccopt_clock_tree -name top -source clk
```

כעת נגדיר אלו תאים יכולים להשתתף בבניית עץ השעון :

```
set_ccopt_mode -cts_inverter_cells {invbd2 invbd4 invbd7 invbda invbdf invbdk}
```

```
set_ccopt_mode -cts_buffer_cells {bufbd1 bufbd2 bufbd3 bufbd4 bufbd7}
```

בשלב זה מגדירים את זמני עליה וירידה מכסימליים :

```
set_ccopt_property target_max_trans 220ps
```

וגם את ה- **skew** המכסימלי

```
set_ccopt_property target_skew 0.2
```

כעת בנה את עץ השעון עם :

```
source IITccopt.src
```

- בדוק את מבנה השעון עם **Clock->CCOpt Clock Tree Debugger**  
Q34 : מה ההשגיה, כלומר הזמן בין שורש השעון ל- **FF** הראשון (בערך) ? מה ה- **skew** כלומר הפרש הזמן המכסימלי בין זמן ההגעת השעון ל- **FF** אחד כלשהו לאחרים (בערך)?

Q35 : הוסף את הציור לדו"ח

- בצע **Timing->Report Timing**. לחץ על האופציה **Post-CTS** בטופס. בצע את הבדיקה גם עבור **Setup** וגם עבור **Hold**.

Q36 : רשום את התוצאות בטבלה. האם יש שיפור (ביחס ל- **Pre-CTS after OptDesign**)?

- בצע
- optDesign -postCTS
- בצע **(Post-CTS) Timing->Report Timing**.
- Q37 : רשום את התוצאות בטבלה. האם יש שיפור ?
- Q38 : הסבר את המושגים **WNS (Worst Negative Slack)** ו- **TNS (Total Negative Slack)**.

#### 4. חיווט: קווי האספקה והתכנון כולו

מדוע חשוב לדעת את שני מספרים ? קרא למנחה על מנת להסביר לו את התשובה!

- כעת נבצע חיווט של רשתות האספקה. בצע **Route->Special Route**. בשדה **nets** רשום **VDD VSS**. לחץ על **OK**. שים לב איך כל התאים מתחברים לרשתות האספקה.
- Q41 : הסבר במילים שלך איזו פעולה מבצעת הפקודה **Route->Special Route**. כלומר, מה המשמעות של " חיווט של רשתות האספקה " ? מה הם ה- **X**-ים הלבנים שמופעים על ה- **layout** ?
- Q42 : הוסף את ה- **layout** לדו"ח.
- שים לב שהחיווט ניסיוני נעלם.
- על מנת להשלים את כל החיבורים בצע **Route->Nanoroute->Route**
- סבלנות (☺)... פעולה זאת לוקחת קצת זמן.
- Q43 : הסבר במילים שלך איזו פעולה מבצעת הפקודה **Route->Nanoroute->Route**.
- Q44 : בחר חוט כלשהו ובדוק עבורו ה- **Attribute Editor**. מה הסטטוס שלו ? מדוע הפעם ה- **status** שונה ?
- אנו שמים לב שמופעים **X**-ים לבנים שמצביעים על שגיאות. ננסה להבין אלו שגיאות קיימות בתכנון. פתח את החלון **Tools->Violation Browser**.
- Q45 : קרא למנחה והסבר לו מהן כל סוגי השגיאות בתכנון ? רשום את התשובה בדו"ח.
- Q46 : לאחר התייעצות עם המנחה, רשום בדו"ח כיצד מתקנים את השגיאות.
- Q47 : הוסף את ה- **layout** לדו"ח.

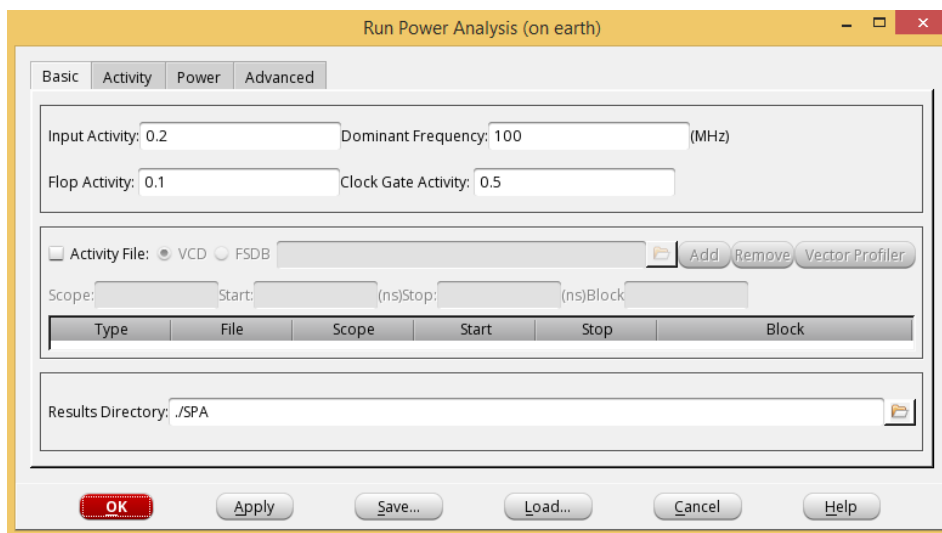
#### 5. אנליזת תיזמון PostRoute והספק

כל האנליזות של התיזמונים שעשינו עד כה, היו מבוססות על אלגוריתם שמעריך את הקיבולים וההתנגדויות של קווי המתכת. לאחר שסיימנו את כל מיקום התאים וקווי הסיגנלים, הכלי יכול לעשות חישוב מדויק של הפרמטרים האלה, ולחשב את התיזמונים בצורה הרבה יותר מדויקת. [עדיין יש לזכור, שזו סימולציה, וביצועי המעגלים בפועל עלולים להיות קצת שונים, אבל זה הדיוק הטוב ביותר שהמתכנן יכול להשיג בשלב הזה]

- ראשית בצע :
- **source preSO.src**
- קובץ ה- **preSO.src** מכיל פקודה שגורמת לכלי לזהות את כל המתכות בצורה נכונה.
- בצע את הפקודה :
- **setAnalysisMode -analysisType onChipVariation**
- שדרוש להרצת האנליזה.
- בצע **Post Route Timing** ל- **setup** ול- **hold**. לחץ על האופציה **Post-Route** בטופס. הפעם זה לוקח יותר זמן כי הכלי מחלץ את הערכים של הקבלים והנגדים הפרזיטיים של החוטים שכרגע הוספנו.
- Q51 : רשום את התוצאות בטבלה.
- לעתים, קיימים חריגות **timing** גם עבור **hold** וגם עבור **setup**.
- ראשית נבקש מהכלי לנסות לפתור את בעיות ה- **hold** עם :
- **setDelayCalMode -reset -siMode**



- אם קיימת בעיית **setup** בצע :  
**optDesign -setup -postRoute**
- אם קיימת בעיית **hold** בצע :  
**optDesign -hold -postRoute**
- **Q52** : רשום את תוצאות ה- **timing** שהתקבלו בטבלה.
- **Q53** : האם עדיין קיימת בעיית **Hold**? האם עדיין קיימת בעיית **Setup** ?
- שים לב שהחריגות קטנו מאד. אנו נסתפק בתוצאות אלה עבור הניסוי.  
**אנליזה הספק :**
- ראשית נגדיר את סוג האנליזה.
- בצע **Power->Power Analysis->Setup**
- עבור **Analysis Method** נבחר ב- **Static**, כלומר, כל טרנזיסטור מיוצג צרכן זרם קבוע.
- עבור **Corner** בחר ב- **max**. לחץ על **OK**.
- פתח את החלון : **Power-> Power Analysis->Run**. הכנס ערכים לכל השדות כפי שמופיע באיור הבא :



- תוצאות האנליזה מופיעות בקובץ **SPA/Top.rpt**. היחידות הן **mW**. פתח את הקובץ ורשום בטבלה את הערכים שהתקבלו עבור :

**Dpram Total Power, Total Internal Power, Total Switching Power, Total Leakage Power, Total Power**

ה- **Dpram Total Power** מופיע בתחילת הקובץ. חפש את השורות הבאות :

cell	Internal Power	Switching Power	Total Power	Leakage Power	Cell Name
w1	-----	0.04741	-----	0.001494	dpram32x32_cb
w2	-----	0.03453	-----	0.001494	dpram32x32_cb
K1	-----	0.09819	-----	0.001395	dpram32x32_cb
CTS_ccl_a_buf_00029	0.03868	0.0615	0.1002	8.163e-08	bufbda
CTS_ccl_a_buf_00025	0.03897	0.05919	0.09816	8.163e-08	bufbda
CTS_ccl_a_buf_00023	0.03898	0.05889	0.09786	8.163e-08	bufbda

כל יתר הנתונים מופיעים בסוף הקובץ.

- **Q64** : רשום את התוצאות בטבלה.
- **Q65** : חזור על הבדיקה כאשר ה- **Corner** הוא **min**. רשום את התוצאות בטבלה.

Corner/Freq/FA	Dpram Power (SRAM)	Total Internal Power	Total Switching Power	Total Leakage Power	Total Power
Max/100Mhz/0.1					
Min/100Mhz/0.1					
Max/200Mhz/0.1					
Min/200Mhz/0.1					
Max/100Mhz/0.2					
Min/100Mhz/0.2					

חוזר על שתי הבדיקות.

- פעם עבור Dominant Frequency = 200Mhz

- פעם עבור Flop Activity = 0.2

Q56 : הסבר את סיבה להבדל בין שתי הפינות (max,min).

Q67 : הסבר את סיבה להבדל עבור התדרים השונים.

Q58 : הסבר את סיבה להבדל עבור ה- Flop Activities השונים.

Q59 : חוץ מהזיכרונות אלו תאים הם צרכני הספק יחסית גדולים ?

**סיום חלק שני !**