The Andrew and Erna Viterbi Faculty of ELECTRICAL & COMPUTER ENGINEERING

Technion Israel Institute of Technology

מעבדה ב- VLSI ספרתי - 0450111

סינתזה ותכנון Layout רקע - (BackEnd)

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

הערות נא לשלוח ל-goel@ee

כל הערה תתקבל בברכה!

עדכון אחרון - 11:18 19/09/2024

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

תוכן עניינים

מבוא.....3
מושגים בסינתזה ובתזמון.....3
ספריית השערים.....3
תזמון ואילוצי תזמון (Timing Constraints).....3
הוספת אמצעי בדיקתיות למעגל : Design For Testability (DFT).....3
כלי סינתזה – Design Vision.....3
כלי ה- Innovus – Layout.....3

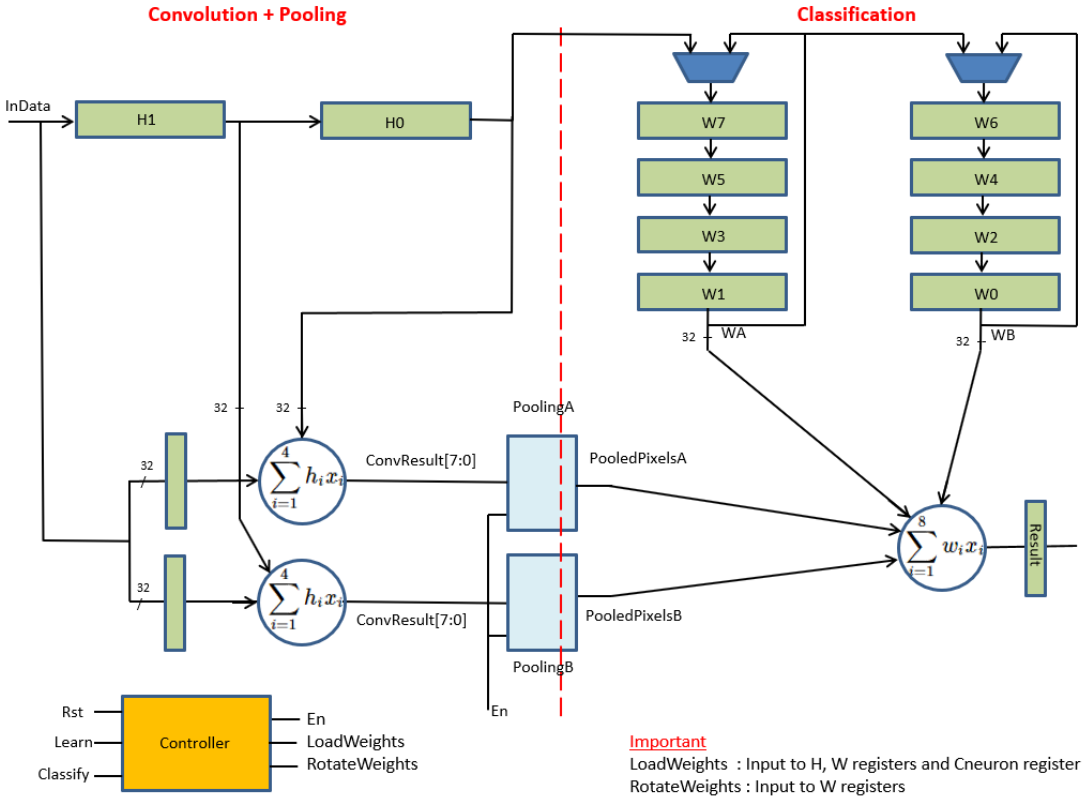
מבוא

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. בנית ה- Layout

מטרת הניסוי היא לבצע את השלבים של הסינתזה וה- layout (כלומר תהליך ה- Backend Design) על תכנון קיים שממומש מאיץ למערכת לומדת. הניסוי יכלול הכרה והפעלה של כלי סינתזה ו- Layout מתקדמים.
העבודה תתבצע על מימוש מאיץ של מערכת לומדת שממומשת בתחילת הניסוי.

להלן סכמת המלבנים של המערכת :



איור מס׳ 1 – הארכיטקטורה של מאיץ של מערכת לומדת

כפי שניתן לראות, המערכת מורכבת יחידת רגיסטרים, שני Convolution Neurons, שתי יחידות Pooling, Fully Connected Neuron אחד והבקר. חשוב רק להכיר את המבנה הלוגי שמתאורת לעיל. לפני שנמשיך נתחיל במבוא קצר של שפת SystemVerilog.

מושגים בסינתזה ובתזמון

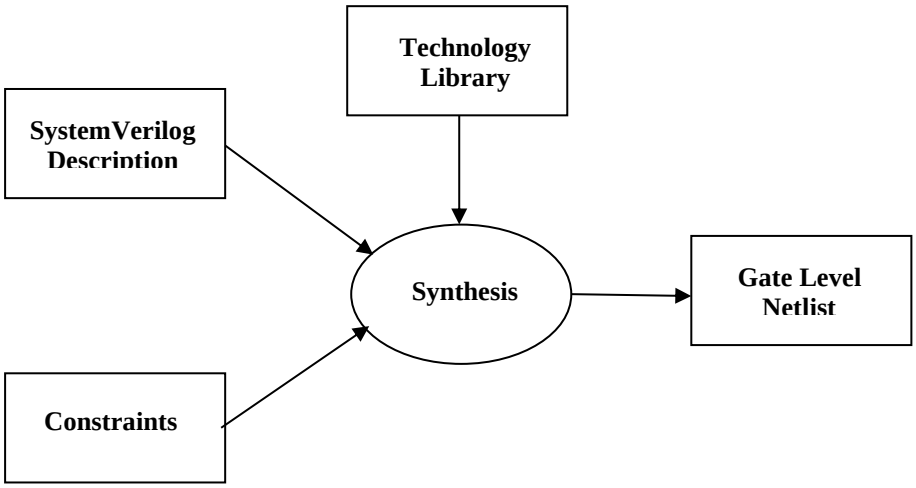
נכיר מספר מושגים מעולם הסינתזה ואנליזת זמנים.

זהו התהליך שבו מתורגם מעגל המתואר בשפת HDL (hardware description language) לדוגמה SystemVerilog למימושו באמצעות שערים לוגיים. כלי סינתזה 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' 2 : 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 מטרנ' במידות W מינימלי, MEDIUM, LARGE 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 הייצרן מה שמבטיח את הדיוק של המודל שלהם.

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

התוצר של הסינתזה הוא הסכימה של המעגל. בשפה המקצועית היא נקראת NETLIST. זהו קובץ שמכיל את כל הפינים של הכניסות והיציאות, את כל קווי הסיגנלים הפנימיים, ואת תאי הספרייה שמרכיבים את המעגל, והחיבוריות שלהם אל כל הסיגנלים.
הסינתזה מממשת את הפונקציות הלוגיות ועושה להם מיטוב (אופטימיזציה וצמצומים לוגיים) ובהמשך גם מנסה לשפר את ההשהיות כך שהמעגל יעבוד בזמן המחזור הרצוי.

על מנת לאפשר ביצוע תהליך הסינתזה, 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_TOWR_
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 ביניריים 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 הסינתזה. 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");
      }
    }
  }
}
```

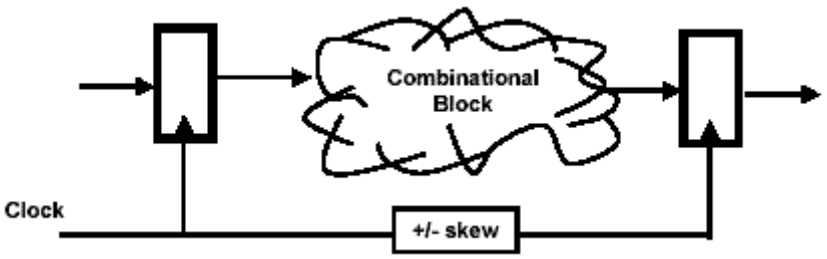
לא נעמיק בהסברים מפורטים על קובץ זה.

תזמון ואילוצי 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׳ 3

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 $\mathbf{T_c}$ קצר ממחזור השעון. אם הוא קצר יותר אז אומרים שלמסלול הקריטי, slack חיובי אחרת ה- **slack** שלילי!!.

$$Slack = T - T_c$$

ברור שתמיד צריך להגיע למצב שבו ה- **slack** יהיה חיובי. כמובן שתיאור פשטני זה מזניח נתונים חשובים כגון **setuptime** ו- **holdtime** (ראה המשך).

Max delay requirement

$$Longest_path_delay + Tclock_to_q < Tcycle - Tsetup + Tskew$$

קיום דרישה זאת מבטיח שהמידע מגיע לפני שהשעון עלה.

Min delay requirement

$$Shortest_path_delay + Tclock_to_q > Thold + Tskew$$

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

הגדרת אילוצים בקובץ sdc

הגדרת שעון :

לפני ביצוע הסנתזה יש להגדיר את אילוצי התכנון. האילוץ הבסיסי שעלינו להגדיר הוא מחזור השעון.

פקודה להגדרת שעון **clock_create** , דוגמא:

create_clock I39/CIN -name clk -period 10 -waveform {**5 0**}

מגדירה אות שעון על ההדק **clk** בעל מחזור **10ns** המתחיל ב - 0 ועולה ל- 1 בזמן **5ns**.

set_input_delay

מגדיר השהיית קלט בפינים או יציאות כניסה ביחס לאות שעון.

הדוגמא הבאה מגדירה השהיית קלט של **1ns** ביחס עלית השעון **clk** עבור כל הקלט:

set_input_delay 1 -clock clk [all_inputs]

set_output_delay

משמש כדי להגביל את ההשהיה עלית שעון לפלט של רגיסטר או אלמנט לוגי אחר.

הדוגמא הבאה מגדירה השהיית פלט מקסימלית ומינימלית עבור יציאה **OUT1** ביחס לקצה ירידת השעון **CLK**:

set_output_delay 1.0 -clock_fall -clock CLK –min {OUT1}
set_output_delay 1.4 -clock_fall -clock CLK –max {OUT1}

הוספת אמצעי בדיקתיות למעגל : 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.

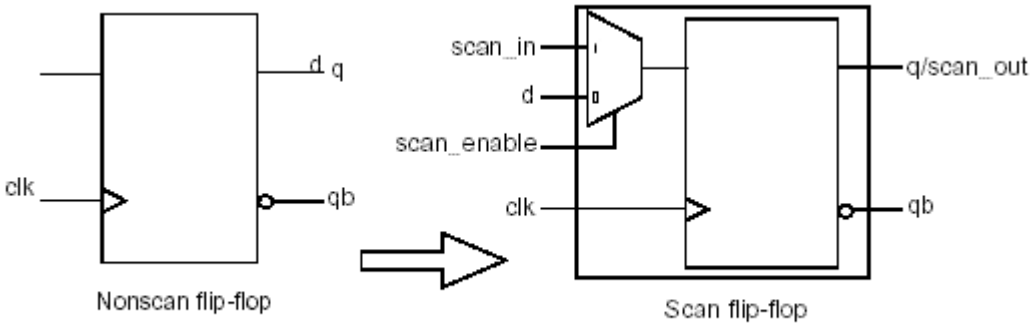
קיימים גם מנגנוני כשל נוספים מורכבים יותר, אך לא נתעסק בהם הפעם.
לאחר היצור נדרש לבדוק את השבב בצורה מקיפה כדי לוודא שהוא פועל בצורה מושלמת והוא נקי מפגמים. אחת הבדיקות החשובות היא בדיקה פונקציונלית. צריכים לוודא שאף צומת אינו מתנהג באחת משלוש הצורות הנ"ל. . על מנת לבצע בדיקה זאת, יש לספק לשבב כניסות 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 הבדיקתיות 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-port :scan_in.

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

ששינוים אלה מגדילים את שטח המעגל ויכולים לפגוע בביצועים, אבל הם חיוניים לשיפור הבדיקתיות.
ניתן לבצע הוספת ה- scan chain בשלב הסינתזה בתצורות שונות באופן אוטומטי.



איור מס' 4: 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.

כלי סינתזה – Design Vision

ה- **design compiler (dc)** (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- **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.

<u>אילוצים שניתן להגדיר לכלי:</u>
- **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
attributes->specify clock. למציאת הדק השעון יש ללחוץ על הרכיב הרצוי ולשנות את הבחירה
cells (hierarchal) ל pins/ports.

<u>קבלת נתונים על הביצועים של המעגל</u>:

- שטח: ע״י **design->report area**
- צריכת הספק: ע״י **design->report power**
- תזמון: ע״י **timing->report timing**

כלי ה- Innovus – Layout

ה- **layout** הוא השלב הבא בעבודת הפיתוח שבו אנו עוברים ממצב של סכמה למימוש הפיזי של המעגלים. בשלב זה אנו ממקמים את המימוש הפיזי של כל שער בשטח המוקצה לנו בצ׳יפ על הסיליקון, ומחברים את כל קווי האותות והספקים. בסיום התהליך יווצר 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 שבנית **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 **Import** <-**Design File**. 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**.

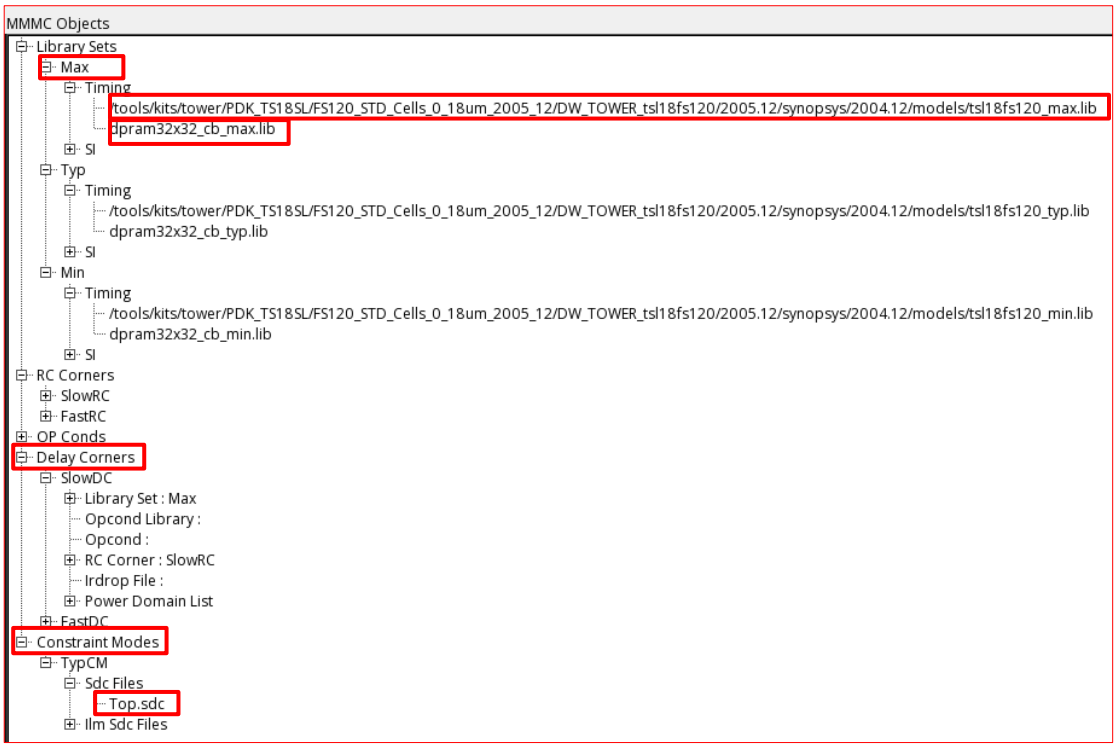
<u>חשוב</u> : הכלי מאפשר הצגת התכנון בשלשה ייצוגים שונים :
א. **Floorplan View** : מציג מיקום הבלוקים
ב. **Amoeba (placement) View** : מציג מיקום התאים הבסיסיים
ג. **Physical View** : מציג את כל הפרטים של המימוש
ניתן לעבור בין הייצוגים ע"י לחיצה על שלשת הכפתורים בצד ימין של המסך.

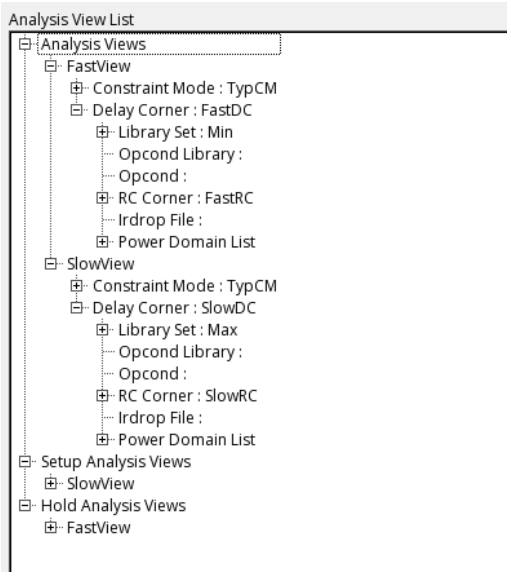
קובץ ה- mmmc.view
בקובץ זה ניתן להגדיר סטים של תנאים בכל מיני צירופים שישמשו את הסימולטור כדי לחשב תיזמונים, והספקים ועוד.
ישנם מספר גורמים שמשפיעים על מהירות התפשטות האותות במעגל וכן על צריכת ההספק עבור סכמה נתונה. ז"א, לאחר שנקבעו השערים, החיבורים, השעונים, וכל הסכימה, יש צורך לחשב את התיזמון וההספקים בתנאים שונים.
הגורמים העיקריים הבלתי תלויים שמשפיעים על התזמונים 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** בטרנ'. 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 – **TYPICAL, MAX** , ו **MIN**. רוב הייצור ייצא במצב **TYP**, אבל הפיזור של אורכי התעלה ופרמטרים נוספים, יהיה בד"כ גאוסייני. כך שבנק' ה **MAX** יאופיין החומר האיטי יותר ברמה של 3 סיגמה, ובנק' 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'. כמובן 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**) , ושאין 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׳ 5 : מבנה קובץ ה- mmmc.view

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.

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 יווצר 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 בקוו צר, יתכן מצב של "פיוז" והקו יישרף.

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 ניתן. 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 הפליפ' פלופים באותו זמן פחות או יותר. להלן דוגמא של רצף הפקודות שממממשות 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**.

<u>מילוי הרווחים</u>

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

<u>חיווט קווי האספקה</u>

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

<u>חיווט התכנון</u>

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

<u>בדיקת התזמון</u>

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