f

2022

Ψηφιακά Ολοκληρωμένα Κυκλώματα VLSI-ASIC Μεγάλης Κλίμακας

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

ΜΑΡΙΟΣ ΠΑΚΑΣ – ΔΙΠΛ. ΗΛ. ΜΗΧ ΚΑΙ ΜΗΧ. ΥΠΟΛ.
ΑΡΙΣΤΟΤΕΛΗΣ ΤΣΕΚΟΥΡΑΣ – ΔΙΠΛ. ΗΛ. ΜΗΧ ΚΑΙ ΜΗΧ. ΥΠΟΛ.
ΒΑΣΙΛΗΣ ΠΑΥΛΙΔΗΣ – ΑΝ. ΚΑΘΗΓΗΤΗΣ, ΑΠΘ

ΑΠΘ | Εργαστήριο Ηλεκτρονικής

Περιεχόμενα

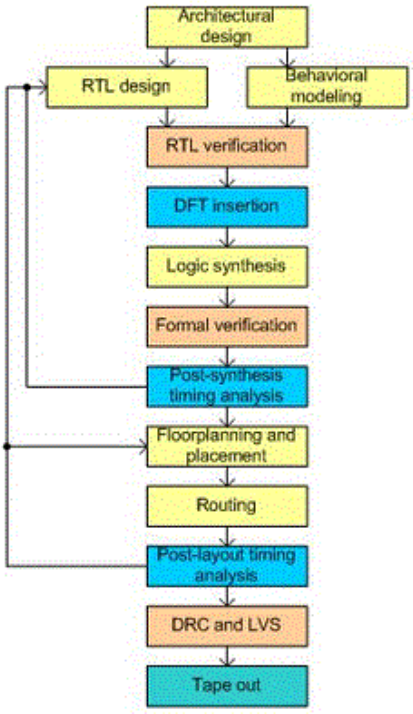
1. Εισαγωγή

Το εγχειρίδιο αυτό αποτελεί ένα βήμα προς βήμα οδηγό για το εργαστήριο του μαθήματος «Ψηφιακά Ολοκληρωμένα Κυκλώματα VLSI-ASIC Μεγάλης Κλίμακας» που διδάχθηκε για πρώτη φορά το ακαδημαϊκό έτος 2020-2021. Η παρούσα έκδοση του είναι η έκδοση v2022 και καταβάλλεται κάθε προσπάθεια να ενημερώνεται ανάλογα με την εξέλιξη των βιομηχανικών εργαλείων που χρησιμοποιούνται.

Τα εργαλεία που χρησιμοποιούνται στο εργαστήριο αυτό αποτελούνται αποκλειστικά από εργαλεία της εταιρείας Cadence (https://www.cadence.com/en_US/home.html) και αποτελούν ένα υποσύνολο των εργαλείων της Cadence τα οποία σχετίζονται με τη σχεδίαση ολοκληρωμένων κυκλωμάτων (IC Package). Τα εργαλεία αυτά καθώς και πολλά άλλα που σχετίζονται με τη σχεδίαση και κατασκευή ολοκληρωμένων κυκλωμάτων παρέχονται στα Ευρωπαϊκά πανεπιστήμια από τον οργανισμό Europractice (http://www.europractice.stfc.ac.uk/welcome.html).

Συνήθως μια βιομηχανική ροή σχεδίασης θα περιέχει εργαλεία σχεδίασης από διαφορετικές εταιρείες, π.χ. Cadence, Synopsys, Mentor/Siemens, καθώς κάθε μία από αυτές τις εταιρείες έχει παράγει εργαλεία για τα διάφορα βήματα της ροής σχεδίασης που έχουν πλέον καθιερωθεί στη βιομηχανία ως στάνταρντ (μέχρι φυσικά να προκύψει και να υιοθετηθεί ένα νέο καλύτερο εργαλείο).

Μία τυπική ροή σχεδίασης ψηφιακών κυκλωμάτων περιλαμβάνει τα βήματα που εμφανίζονται στο Σχήμα 1. Όπως φαίνεται από το Σχήμα 1, μία τέτοια ροή αποτελείται από πολλά βήματα που με τη σειρά τους απαιτούν τη χρήση διαφορετικών εργαλείων. Επομένως, στόχος του εργαστηρίου του μαθήματος είναι να καλυφθούν κάποια βασικά βήματα μία τέτοιας ροής τα οποία καλύπτονται διεξοδικά και στις διαλέξεις του μαθήματος. Τα βήματα αυτά περιλαμβάνουν τη σύνθεση (logic synthesis), τη χωροθέτηση (floorplanning), και την τοποθέτηση και δρομολόγηση (placement & routing) τα οποία πραγματώνονται στις αντίστοιχες ενότητες των εγχειριδίου αυτού. Επίσης, μία ξεχωριστή ενότητα αφιερώνεται στη στατική ανάλυση χρονισμού (STA).



Σχήμα 1. Τυπική ροή σχεδίασης ενός ψηφιακού ολοκληρωμένου κυκλώματος VLSI – ASIC.

Όλα (ή σχεδόν όλα) τα εργαλεία σχεδίασης ολοκληρωμένων κυκλωμάτων ανεξάρτητα από την εταιρεία παραγωγής τους χρησιμοποιούν κατά κόρον τη γλώσσα προγραμματισμού Tcl. Κάποιες βιβλιογραφικές πηγές για τη γλώσσα αυτή είναι οι ακόλουθες:

[1]. TclTutor, a computer aided instruction package for learning the Tcl language: http://www.msen.com/~clif/TclTutor.html (accessed on 7/11/20)

[2]. John K. Ousterhout, *TCL Reference, Tcl and the Tk Toolkit*, ISBN-13 : 978-0201633375, Addison-Wesley Publishing Company, 1994.

[3]. Brent Welch and Ken Jones, *Practical Programming in Tcl and Tk*, 4th Edition, ISBN-13 : 978-0130385604, Pearson, 2003.

[4]. Tcl on-line command reference manual, https://www.tcl.tk/man/tcl8.6/contents.htm (accessed on 7/11/20)

Το υλικό που περιέχεται στο εγχειρίδιο αυτό έχει βασιστεί εν πολλοίς στα εγχειρίδια χρήσης των εργαλείων της Cadence καθώς και σε επιπλέον υλικό που έχει αναπτύξει η εταιρεία για την ταχύτερη υιοθέτηση των εργαλείων της γνωστά και ως Rapid Adoption Kits (RAK).

1.1. Κανόνες Σύνταξης

Ο παρακάτω πίνακας παρέχει τη βασική σύνταξη και σημειολογία που χρησιμοποιούνται για την περιγραφή των εντολών και άλλων παραμέτρων σχετικά με τη χρήση των εργαλείων. Έχει καταβληθεί κάθε προσπάθεια η σύνταξη αυτή να ακολουθείται σε όλο το εγχειρίδιο.

Πίνακας 1. Βασικοί κανόνες σύνταξης για το εγχειρίδιο.

Σύμβολο/Σύνταξη	**Περιγραφή**
literal	Κανονικοί χαρακτηρές (μη πλάγια γραφή (italics)) αντιστοιχούν σε λέξεις κλειδιά που εισάγονται ως είναι. Αυτές αντιστοιχούν σε εντολές, επιλογές, και χαρακτηριστικά εντολών.
arguments and options	Λέξεις με πλάγια γραφή παραπέμπουν σε χαρακτηριστικά ή πληροφορία ορισμένα από το χρήστη, όπου πρέπει να δωθεί μία αλφαριθμητική ή αριθμητική τιμή.
emphasis	Λέξεις με έντονα γράμματα αποσκοπούν απλά να δώσουν έμφαση στη συγκεκριμένη λέξη ανεξάρτητα από τη σημασία ή/και ρόλο της.
|	Η κάθετη μπάρα αντιστοιχεί στη λογική λειτουργία OR, δείχνοντας ξεχωριστές και δυνατές επιλογές για ένα συγκεκριμένο argument μιας εντολής.
[]	Οι αγκύλες (brackets) δείχνουν προαιρετικά arguments. Όταν συνδυάζονται με μπάρες (|), εσωκλείουν μία λίστα από διάφορες επιλογές για το argument αυτό.
{}	Τα άγκιστρα (braces) δείχνουν ότι μία επιλογή **απαιτείται** από τη λίστα των argument που διαχωρίζονται με μπάρες OR (|). Παραδείγματος χάρη πρέπει να επιλεγεί ένα από την παρακάτω λίστα: { argument1 | argument2 | argument3 }
{}	**Προσοχή:** Τα άγκιστρα σε εντολές Tcl δείχνουν ότι τα άγκιστρα πρέπει να εισαχθούν σαν literal χαρακτήρες καθώς είναι μέρος της εντολής Tcl.
...	Τα αποσιωπητικά δείχνουν ότι μπορείτε να επαναλάβετε το προηγούμενο argument. Αν τα αποσιωπητικά χρησιμοποιούνται εντός αγκυλών (δηλαδή [argument...]), τότε μπορείτε να έχετε κανένα ή περισσότερα argument. Αν τα αποσιωπητικά χρησιμοποιούνται χωρίς αγκύλες (argument...), τότε πρέπει να εισάγετε τουλάχιστον ένα argument.
#	Το σύμβολο pound προηγείται των σχολίων σε γραμμές εντολών.

2. Αρχική Προετοιμασία

Πριν προχωρήσουμε στην εκτέλεση ενός οποιοδήποτε βήματος της ροής σχεδίασης και για οποιοδήποτε εργαλείο, πρέπει να προετοιμάσουμε κατάλληλα το περιβάλλον εργασίας του εκάστοτε εργαλείου. Αν και τα διάφορα εργαλεία της Cadence ακολουθούν πλέον κατά ένα πολύ μεγάλο μέρος τις ίδιες ρυθμίσεις και εντολές λόγω της ομογενοποίησης των εργαλείων που πραγματοποιήθηκε πρόσφατα μέσω του «ενοποιημένου περιβάλλοντος εργασίας χρήστη» (Unified User Interface), θα παρέχουμε τις εντολές προετοιμασίας για το κάθε εργαλείο όσο το δυνατό ξεχωριστά.

Το ενοποιημένο περιβάλλον εργασίας χρήστη υποστηρίζεται (αυτή τη στιγμή) από τα εργαλεία Genus (σύνθεση), Innovus (χωροθέτηση, τοποθέτηση, και δρομολόγηση), και Tempus (ανάλυση χρονισμού) και συνήθως αναφέρεται ως Stylus common UI (User Interface). Όταν καλούνται τα εργαλεία αυτά, τότε εξ ορισμού χρησιμοποιούν το περιβάλλον αυτό. **Σημείωση**: Το εγχειρίδιο αυτό βασίζεται στη χρήση αυτού του περιβάλλοντος εργασίας.

3. Σύνθεση (Synthesis)

Η ενότητα αυτή περιγράφει τα βήματα που απαιτούνται για τη διαδικασία της σύνθεσης ενός ολοκληρωμένου κυκλώματος. Εντολές σχετικές με την εκκίνηση του εργαλείου παρέχονται στην υποενότητα 3.1 Τρόποι για την παροχή βοήθειας ως προς τη χρήση του εργαλείο παρέχεται στην υποενότητα 3.2. Μία βήμα προς βήμα περιγραφή της διαδικασίας σύνθεσης παρουσιάζεται στην υποενότητα 3.3 και οι απαραίτητοι περιορισμοί περιγράφονται στην υποενότητα 3.4, αντίστοιχα. Τέλος, τα βήματα της διαδικασίας σύνθεσης παρουσιάζονται συγκεντρωτικά στην υποενότητα 3.5.

3.1. Εκκίνηση του εργαλείου

Το εργαλείο σύνθεσης κυκλωμάτων που έχουν περιγραφεί σε γλώσσες HDL, όπως Verilog, SystemVerilog, και VHDL είναι το Genus και καλείται από ένα τερματικό με την εντολή:

```
genus
```

Κατά την εκκίνηση το εργαλείο αναζητεί μεταξύ άλλων και το αρχείο genus.tcl στο home directory, το οποίο περιέχει πληροφορία για τις αρχικές ρυθμίσεις του εργαλείου. Δεν είναι υποχρεωτικό και δεν χρησιμοποιείται στο εργαστήριο αυτό.

Σημείωση: Μη χρησιμοποιήσετε το χαρακτήρα «&» ως είθισται σε περιβάλλον Unix.

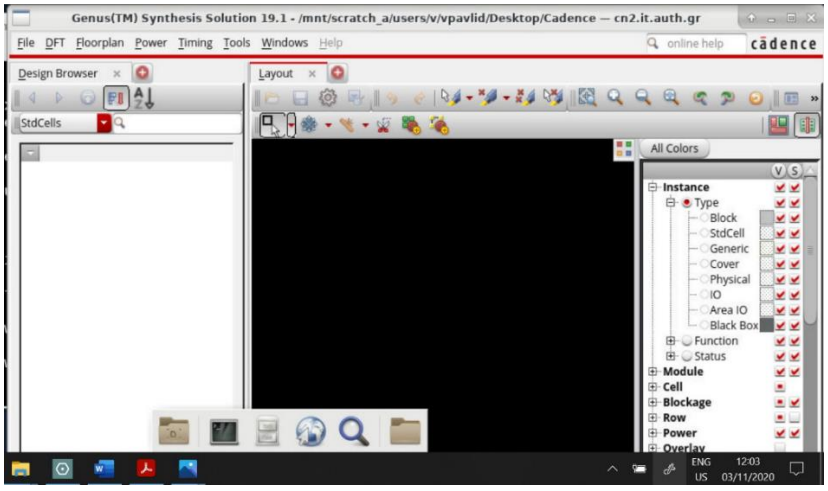
Το εργαλείο τυπώνει κάποια πληροφορία σχετικά π.χ. με τον τύπο άδειας που καλείται και την έκδοσή του κλπ και εν συνεχεία εμφανίζεται η γραμμή εργασίας:

```
@genus:root: 4>
```

Το γραφικό περιβάλλον του εργαλείου που εικονίζεται στο Σχήμα 2 μπορεί να κληθεί με την εντολή

```
gui_show
```

Κάθε φορά που καλείται το εργαλείο δημιουργεί δύο αρχεία εξόδου με καταλήξεις xx.cmd και xx.log, όπου xx είναι ο αύξων αριθμός των αρχείων. Τα αρχεία με κατάληξη .cmd περιέχουν όλες τις εντολές που εκτελέσατε στη γραμμή εντολών για κάθε συνεδρία, ενώ τα αρχεία .log περιέχουν διάφορα μηνύματα και πληροφορίες σχετικά με τη συνεδρία (session).



Σχήμα 2. Γραφικό περιβάλλον του εργαλείου σύνθεσης genus®.

3.2. Βοήθεια για τη Χρήση του Εργαλείου

Στη γραμμή εντολής εμφανίζονται διάφορα μηνύματα ανάλογα με τις εντολές και λειτουργίες που εκτελούνται τα οποία μπορούν να κληθούν με την εντολή:

```
genus:root: 15> report_messages
```

Για συγκεκριμένα μηνύματα (π.χ. TUI-613) μπορεί εναλλακτικά να χρησιμοποιηθεί η εντολή:

```
genus:root: 17> vls -a TUI-613
```

ή εναλλακτικά με τη βοήθεια της εντολής help:

```
genus:root: 18> help TUI-613
```

Υπάρχουν αρκετοί τρόποι για να λάβετε βοήθεια σε διάφορα θέματα όπως η σύνταξη εντολών και argument αυτών. Κάποιοι από αυτούς τους τρόπους περιγράφονται εδώ. Ο πιο απλός τρόπος είναι με τη χρήση της εντολής help *command_name*. Για παράδειγμα για την εντολή path_group

```
genus:root: 24> help path_group
```

επιστρέφει τη σύνταξη της αντίστοιχης εντολής. Παρόμοια για κάποια ιδιότητα εντολής (attribute) έχουμε:

```
genus:root: 28> help -attr attribute_name
```

όπου το παράδειγμα:

```
genus:root: 31> help -attr max_transition
```

επιστρέφει την περιγραφή της ιδιότητας αυτής και σε ποια αντικείμενα (objects) μπορεί να εφαρμοστεί.

Εναλλακτικά, αν δεν είστε σίγουροι για τη σύνταξη μιας εντολής, μπορείτε να την αναζητήσετε χρησιμοποιώντας μόνο μέρος αυτής και να πατήσετε το πλήκτρο <Tab>. Το τύπωμα κάποιων χαρακτήρων και το πλήκτρο <Tab> εμφανίζουν όλες τις εντολές που περιέχουν τους χαρακτήρες αυτούς. Για παράδειγμα:

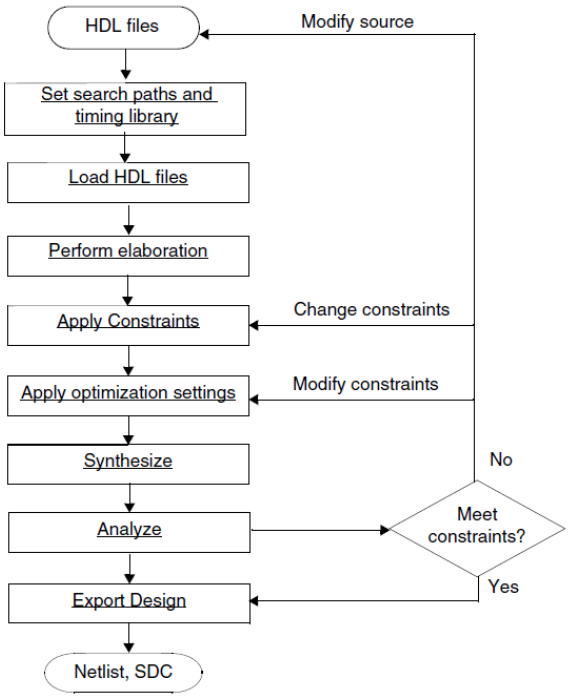
```
genus:root: 41> path_<Tab>
```

Επιστρέφει: `path_group`

3.3. Βήματα της Διαδικασίας Σύνθεσης

Έχοντας εκκινήσει το εργαλείο (με ή χωρίς το γραφικό περιβάλλον) είμαστε έτοιμοι για να προχωρήσουμε στη σύνθεση του κυκλώματος που επιθυμούμε. Για τους σκοπούς του εργαστηρίου αυτού τα κυκλώματα έχουν περιγραφεί σε γλώσσα Verilog και μπορεί να αποτελούνται από ένα ή περισσότερα module. Σε περίπτωση ενός module, τότε αυτό είναι και το top level design και επομένως δε νοείται η έννοια της ιεραρχικής σχεδίασης. Αντιθέτως, μπορεί το κύκλωμά μας να αποτελείται από ένα σύνολο module εμπεριέχοντας αρκετά επίπεδα ιεραρχίας. Ανάμεσα σε αυτά τα module, θα υπάρχει ένα που θα αποτελεί το top level module ή design.

Επίσης, υπάρχουν αρκετές παραλλαγές της διαδικασίας σύνθεσης ανάλογα με το τί επιθυμεί να επιτύχει ο σχεδιαστής. Εδώ καθώς το εργαστήριο αποτελεί μια εισαγωγή στη ροή σχεδίασης, έχει προτιμηθεί η χρήση της βασικής σύνθεσης που περιλαμβάνει τα βήματα που δείχνονται στο Σχήμα 3. Τα περισσότερα από τα βήματα αυτά περιγράφονται στις επόμενες παραγράφους και θα πρέπει να προσαρμοσθούν στα κυκλώματα που σας έχουν δοθεί (ή εναλλακτικά μπορείτε να δοκιμάσετε τις δικές σας σχεδιάσεις Verilog).



Σχήμα 3. Βήματα της βασικής διαδικασίας σύνθεσης με το εργαλείο genus®.

Αρχικά πρέπει να πούμε στο εργαλείο πού βρίσκεται η πληροφορία τεχνολογίας, τα αρχεία περιγραφής του κυκλώματος, καθώς και διάφορα script που μπορούμε ενδεχομένως να χρησιμοποιήσουμε για να καθοδηγήσουμε το genus. Για να το κάνουμε αυτό, πρέπει να θέσουμε τα αντίστοιχα μονοπάτια που θα χρησιμοποιήσει το εργαλείο (εναλλακτικά κάτι τέτοιο θα μπορούσε

να γίνει με τη χρήση ενός αρχείου genus_setup.tcl του εργαλείου, όπως περιγράφεται και παρακάτω):

```
genus:root: 11> set_db init_lib_search_path path
genus:root: 12> set_db script_search_path path
genus:root: 13> set_db init_hdl_search_path path
```

όπου *path* είναι το μονοπάτι όπου βρίσκονται η βιβλιοθήκη, τα αρχεία Verilog, και τυχόν script.

Έχοντας θέσει το μονοπάτι αναζήτησης της τεχνολογίας, μπορούμε να δώσουμε τη συγκεκριμένη βιβλιοθήκη χρονισμού που θέλουμε να χρησιμοποιήσουμε (πολλαπλές βιβλιοθήκες μπορεί να είναι διαθέσιμες).

```
genus:root: 16> set_db library lib_name.lib
```

Το Genus θα χρησιμοποιήσει τη βιβλιοθήκη *lib_name.lib* για το σκοπό της σύνθεσης. Ελέγξτε ότι δείχνετε προς τη βιβλιοθήκη όταν βρίσκεστε στο κορυφαίο directory (δηλαδή root-level ("/")). Εκτελέστε vpwd (όπου ο χαρακτήρας 'v' σημαίνει «virtual») στο genus και αν βρίσκεστε στο κορυφαίο directory, θα επιστρέψει (root:).

Με παρόμοιο τρόπο μπορεί να γίνει ανάγνωση των αρχείων βιβλιοθήκης φυσικών πληροφοριών (*.lef) μέσω της εντολής:

```
genus:root: 17> set_db lef_library lef_name.lef
```

καθώς και του αρχείου παρασιτικών QRC (*.tch) μέσω της:

```
genus:root: 18> read_qrc qrc_name.tch
```

Έχοντας προσδιορίσει τις βιβλιοθήκες που θα χρησιμοποιηθούν, πρέπει επίσης να επιλέξουμε τον τρόπο μοντελοποίησης των διασυνδέσεων. Αυτό καθορίζεται από την ιδιότητα (attribute) interconnect_mode ως εξής:

- `wireload (default)` παραπέμπει το εργαλείο της σύνθεσης να χρησιμοποιήσει wire-load models
- `ple` παραπέμπει το εργαλείο της σύνθεσης να χρησιμοποιήσει Physical Layout Estimators (PLEs). Οι PLE χρησιμοποιούν τα φυσικά χαρακτηριστικά του κυκλώματος (physical information), όπως LEF βιβλιοθήκες, βοηθώντας έτσι τα εργαλεία φυσικής σχεδίασης (γνωστά και ως backend tools) να ικανοποιήσουν τις απαιτήσεις χρονισμού (και άλλες) πιο γρήγορα, με άλλα λόγια να επιτύχουν "timing closure".

Σημείωση: Όταν διάβαζουμε αρχεία LEF σαν είσοδο, η ιδιότητα interconnect_mode (δηλαδή τα μοντέλα διασυνδέσεων) τίθεται αυτόματα στο ple.

Όλες αυτές οι εντολές που στην ουσία αρχικοποιούν τη διαδικασία σύνθεσης μπορούν να συμπεριληφθούν σε ένα αρχείο που θα διαβάζεται όταν εκκινείται το εργαλείο genus. Θυμηθείτε ότι το εργαλείο περιέχει έναν Tcl parser και ερμηνεύει απευθείας εντολές της γλώσσας Tcl. Υπάρχουν δύο τρόποι για να γίνει αυτό. Είτε να δημιουργήσετε ένα αρχείο με το όνομα genus_startup.tcl στο φάκελο από τον οποίο καλείται το εργαλείο genus ή να δημιουργήσετε ένα αρχείο run.tcl και να το δηλώσετε όταν καλείται το εργαλείο:

```
genus -f run.tcl
```

Στο επόμενο βήμα (βλέπε Σχήμα 3), πρέπει να φορτώσουμε τα αρχεία HDL που περιγράφουν το κύκλωμα που θέλουμε να συνθέσουμε. Για το σκοπό αυτό χρησιμοποιούμε την εντολή `read_hdl`. Με την εκτέλεση της εντολής αυτή, το genus διαβάζει τα αρχεία και πραγματοποιεί κάποιους ελέγχους συντακτικής ορθότητας του κώδικα (όχι σωστής λειτουργίας του κυκλώματος! Αυτήν πρέπει να την έχετε επιτύχει προηγουμένως με διεξοδική λειτουργική προσομοίωση). Για να φορτώσουμε ένα ή περισσότερα αρχεία Verilog, μπορούμε είτε να το κάνουμε διαδοχικά:

```
read_hdl file1.v
read_hdl file2.v
read_hdl file3.v
```

Είτε να τα διαβάσουμε όλα μαζί εκτελώντας:

```
read_hdl { file1.v file2.v file3.v }
```

Χρήσιμη Συμβουλή: Στην περίπτωση της ιεραρχικής σχεδίασης (με τον όρο ιεραρχική σχεδίαση εννοούμε το διαχωρισμό του κυκλώματος σχεδίασης σε απλούστερα υποκυκλώματα τα οποία είναι πιο εύκολο να σχεδιαστούν και να επαληθευτούν. Στην περίπτωση αυτή καταλήγουμε σε περισσότερα από ένα αρχεία περιγραφής υλικού) όταν διαβάζουμε τα αρχεία στο Genus με την εντολή `read_hdl` είναι πολύ σημαντικό να εισάγονται από το πιο χαμηλό στο πιο υψηλό επίπεδο. Με άλλα λόγια, δεν θα πρέπει να προηγείται κάποιο αρχείο που αρχικοποιεί κάποιο instance που ορίζεται σε αρχείο που δεν έχει διαβαστεί ακόμα.

Έχοντας διαβάσει τα αρχεία περιγραφής του κυκλώματος, πρέπει να επεξεργαστούμε λεπτομερώς το προς σύνθεση κύκλωμα. Αυτό γίνεται με το βήμα Elaboration (βλέπε Σχήμα 3) και απαιτείται μόνο για το top-level module. Η εντολή `elaborate` αυτόματα επεξεργάζεται το «κορυφαίο» αυτό κύκλωμα και όλες τις αναφορές που περιέχει. Κατά τη διάρκεια του βήματος αυτού εκτελούνται οι παρακάτω εργασίες:

- Φτιάχνονται δομές δεδομένων
- Εισάγονται τα ακολουθιακά στοιχεία του κυκλώματος
- Πραγματοποιούνται βελτιστοποιήσεις του HDL σε υψηλό επίπεδο, όπως αφαίρεση κώδικα HDL που δε χρησιμοποιείται (dead code)
- Ελέγχονται τους χρησιμοποιούμενους συμβολισμούς (semantics) στον HDL κώδικα

Σημείωση 1: Αν υπάρχουν κάποια επιπέδου πύλης (gate-level) netlist που διαβάζονται με τα αρχεία RTL, το Genus αυτόματα συνδέει τα κελιά με τις αναφορές τους στη βιβλιοθήκη τεχνολογίας κατά το βήμα αυτό. Επομένως, δεν απαιτείται κάποια επιπρόσθετη εντολή για τη σύνδεση αυτή.

Σημείωση 2: Πριν εκτελέσετε την εντολή αυτή διαβάστε την υποενότητα 3.3.1

Η γενική μορφή της εντολής αυτής είναι

`elaborate` `[-h] [-parameters {}] [`*`top_module_name`*`]`

Στο τέλος του elaboration, το Genus θα εμφανίσει οποιεσδήποτε μη επιλυόμενες αναφορές μετά από το μήνυμα «Done elaborating»:

```
Done elaborating '<top_level_module_name>'.
Cannot resolve reference to <ref01>
```

```
Cannot resolve reference to <ref02>
Cannot resolve reference to <ref03> ...
```

Εναλλακτικά, είναι καλό να ελέγχουμε το αποτέλεσμα της σύνθεσης και αν υπάρχουν κάποια άλυτα από το εργαλείο προβλήματα (unresolved references), οι οποίες θα πρέπει να αναλυθούν πριν προχωρήσουμε στη σύνθεση. Η εντολή για τον έλεγχο της σχεδίασης είναι:

```
check_design ή check_design -all
```

Μετά το βήμα του elaborate, μπορούμε να εφαρμόσουμε περιορισμούς (constraints) και άλλες λειτουργίες. Οι περιορισμοί αυτοί περιλαμβάνουν:

- Συνθήκες λειτουργίας (Operating conditions)
- Σήματα χρονισμού (ρολόγια) (Clock waveforms)
- Χρονισμό των εισόδων και εξόδων του κυκλώματος (I/O timing)

Η εφαρμογή των περιορισμών αυτών μπορεί να γίνει με διάφορους τρόπους:

- Εισαγωγή απευθείας στη γραμμή εντολής του Genus
- Με τη χρήση ενός αρχείου περιορισμών (constraints file)
- Ανάγνωση περιορισμών SDC (SDC constraints)

Προτείνεται γενικά η χρήση ενός αρχείου περιορισμών .sdc ή αρχείου tcl. Για το εργαστήριο αυτό προτιμάται η δημιουργία και χρήση ενός .sdc αρχείου που περιέχει τους περιορισμούς. Ως ελάχιστος περιορισμός απαιτείται ο ορισμός ενός σήματος ρολογιού! Θυμηθείτε, εν γένει, μιλάμε για σύγχρονη σχεδίαση. Για τη δημιουργία αυτών των περιορισμών δείτε την Ενότητα 3.4.

Πέρα από τη χρήση περιορισμών σχεδίασης, μπορούμε να εφαρμόσουμε επιπρόσθετες στρατηγικές για τη βελτιστοποίηση των επιδόσεων του κυκλώματος που συνθέτουμε. Με το Genus, μπορούμε να πραγματοποιήσουμε οποιαδήποτε από τις ακόλουθες βελτιστοποιήσεις:

- Απομάκρυνση της ιεραρχίας του κυκλώματος (ungrouping)
- Δημιουργία επιπλέον επιπέδων ιεραρχίας (grouping)
- Σύνθεση ενός υπο-κυκλώματος
- Δημιουργία συνόλων μονοπατιών στο κύκλωμα για την αλλαγή της συνάρτησης κόστους της σύνθεσης

Για παράδειγμα, τα μονοπάτια χρονισμού του κυκλώματος μπορούν να ταξινομηθούν σε διαφορετικά γκρουπ με διαφορετικό κόστος για το κάθε σύνολο:

- Input-to-Output paths (I2O)
- Input-to-Register paths (I2C)
- Register-to-Register (C2C)
- Register-to-Output paths (C2O)

όπου για κάθε γκρουπ, το μονοπάτι με τη χειρότερη καθυστέρηση οδηγεί τη συνάρτηση κόστους της σύνθεσης.

Για να εισάγουμε τους περιορισμούς στο εργαλείο, μπορούμε να χρησιμοποιήσουμε την εντολή

```
read_sdc filename.sdc
```

όπου το όνομα του αρχείου με την κατάληξη .sdc περιέχει τους περιορισμούς σύμφωνα με το στάνταρντ (πρότυπο) «Synopsys Design Constraints». Αν το αρχείο περιέχεται στο μονοπάτι που

ορίσατε για τα scripts, τότε αυτό θα διαβαστεί απευθείας από εκεί. Μετά την ανάγνωση αυτού του αρχείου αλλά και σε άλλα βήματα της διαδικασίας σχεδίασης (που έπονται του βήματος του elaboration) μπορείτε να ελέγξετε αν ο χρονισμός του κυκλώματος και οι σχετικοί περιορισμοί έχουν περιγραφεί με τέτοιο τρόπο ώστε η ανάλυση χρονισμού να προχωρήσει χωρίς ιδιαίτερα προβλήματα. Για το σκοπό αυτό μπορείτε να εκτελέσετε την εντολή:

```
check_timing_intent
```

όπου μπορείτε να δείτε περισσότερες επιλογές για την εντολή αυτή μέσου του «help».

Έχοντας εισάγει τους περιορισμούς σχεδίασης (και βελτιστοποίησης), είμαστε έτοιμοι για τη σύνθεση (βλέπε Σχήμα 3) του κυκλώματος με τις εντολές:

```
genus:root: 34> syn_generic
```

```
genus:root: 35> syn_map
```

Η εκτέλεση των εντολών αυτών μπορεί να πάρει αρκετό χρόνο ανάλογα με το μέγεθος του κυκλώματος και το είδος των εφαρμοσμένων περιορισμών από το προηγούμενο βήμα. Με την ολοκλήρωση των δύο αυτών βημάτων, έχουμε πλέον ένα αντιστοιχισμένο στην τεχνολογία gate-level netlist.

Τέλος, μπορείτε να βελτιστοποιήσετε το αποτέλεσμα της σύνθεσης από τα δύο προηγούμενα βήματα σύνθεσης με τη χρήση της παρακάτω εντολής:

```
genus:root: 35> syn_opt
```

Μετά τη σύνθεση του κυκλώματος, μπορούμε να παράγουμε λεπτομερή αναφορά για το χρονισμό και την επιφάνεια του κυκλώματος χρησιμοποιώντας το ευρύ σύνολο των εντολών `report_*`:

- Για μία λεπτομερή αναφορά επιφάνειας, χρησιμοποιείστε `report_area`
- Για να παράγετε μία λεπτομερή επιλογή πυλών και αναφορά επιφάνειας, χρησιμοποιείστε `report_gates`
- Για να παράγετε μία λεπτομερή αναφορά χρονισμού συμπεριλαμβανομένου και του χειρότερου μονοπατιού του παρόντος κυκλώματος, χρησιμοποιείστε `report_timing`
- Για να παράγετε μία αναφορά κατανάλωσης ισχύος, χρησιμοποιείστε `report_power`

Το τελευταίο βήμα της διαδικασίας σύνθεσης περιλαμβάνει την εξαγωγή του κυκλώματος μετά τη σύνθεση και την παραγωγή των gate-level netlist και των περιορισμών σε μορφή SDC.

Σημείωση: Εξορισμού οι εντολές `write_*` γράφουν την έξοδο στο `stdout`. Αν επιθυμείτε την εγγραφή σε κάποιο άλλο αρχείο τότε πρέπει να χρησιμοποιήσετε τον τελεστή επανακατεύθυνσης (>) και ένα όνομα αρχείου. Για την εγγραφή για παράδειγμα του gate-level netlist, χρησιμοποιήστε την εντολή `write_hdl`

```
genus:root: 21> write_hdl > design.v
```

Η εντολή αυτή γράφει το gate-level netlist σε ένα αρχείο με το όνομα design.v.

Για να εξαχθούν οι περιορισμοί σχεδίασης και ενδεχομένως και περιορισμοί δοκιμής, χρησιμοποιήστε την εντολή `write_script`, όπως στο παρακάτω παράδειγμα:

```
genus:root: 25> write_script > constraints.g
```

Η εντολή αυτή γράφει τους περιορισμούς σε ένα αρχείο με το όνομα constraints.g.

Για να εξαχθούν οι περιορισμοί σχεδίασης σε μορφή SDC, χρησιμοποιήστε την εντολή `write_sdc`, όπως στο παρακάτω παράδειγμα:

```
genus:root: 31> write_sdc > constraints.sdc
```

Η εντολή αυτή γράφει τους περιορισμούς SDC σε ένα αρχείο με το όνομα constraints.sdc.

Για να εξάγουμε όλα τα απαραίτητα αρχεία για τα επόμενα βήματα της ροής σχεδίασης όπου θα χρησιμοποιηθεί το εργαλείο innovus, χρησιμοποιείται η παρακάτω εντολή:

```
genus:root: 34> write_design -innovus design_name
```

Επίσης, μπορούμε να δημιουργήσουμε ένα template το οποίο μπορεί να χρησιμοποιηθεί ως βάση για επόμενες προσπάθειες σύνθεσης ενός κυκλώματος με τροποποιήσεις ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα σύνθεσης. Σε αυτήν την περίπτωση, η εντολή που χρησιμοποιείται είναι:

```
write_template [-dft] [-power] [-cpf] [-retime] [-physical] [-performance] [-area] [-no_sdc] [-n2n] [-yield] [-full] [-simple] [-split] [-multimode] -outfile file
```

Κάποια παραδείγματα χρήσης της εντολής αυτής δίνονται παρακάτω, ενώ η πλήρη σύνταξή της παρατίθεται στο Παράρτημα Α.

Παραδείγματα:

- Το ακόλουθο παράδειγμα παράγει ένα αρχείο με ένα βασικό template με τους περιορισμούς με σύνταξη SDC

```
write_template -outfile template.g
```

- Το ακόλουθο παράδειγμα παράγει ένα αρχείο με ένα βασικό template με τους περιορισμούς με σύνταξη Genus

```
write_template -no_sdc -outfile template.g
```

- Το ακόλουθο παράδειγμα προσθέτει ιδιότητες σχετικές και με την εισαγωγή DFT και με την ισχύ στο αρχείο template.g που παράγεται

```
write_template -dft -power -outfile template.g
```

- Το ακόλουθο παράδειγμα παράγει ένα template με τις ιδιότητες σχετικές με DFT και εντολές για βελτιστοποίηση netlist με netlist

```
write_template -dft -n2n -outfile template.g
```

- Το ακόλουθο παράδειγμα παράγει ένα template.g και το αρχείο ρυθμίσεων setup_template.g που περιέχει όλες τις ιδιότητες σε επίπεδο root και μεταβλητές ρυθμίσεων και τις περιλαμβάνει στο αρχείο template.g

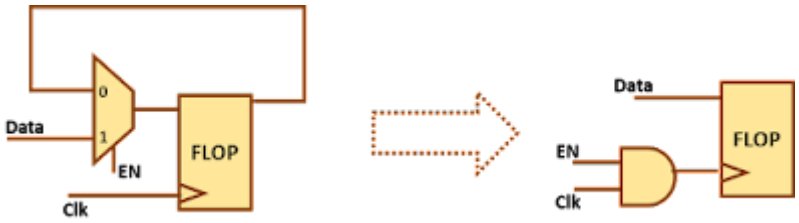
```
write_template -split -outfile template.g
```

Τέλος, μπορούμε να φύγουμε από το περιβάλλον του Genus εκτελώντας είτε `quit` ή `exit` στη γραμμή εντολών ή χρησιμοποιώντας δύο συνεχόμενες φορές τη συντόμευση <Control-c>.

3.3.1. Σύνθεση με Χρήση Φραγής Ρολογιού (Clock Gating)

Η φραγή ρολογιού (Clock Gating) είναι μια τεχνική που χρησιμοποιείται στα κυκλώματα με στόχο τη μείωση της κατανάλωσης ισχύος τους. Η ιδέα είναι ότι πολύ συχνά υπάρχουν ακολουθιακά κυκλώματα τα οποία παρά τις αλλαγές του ρολογιού δεν αλλάζουν την κατάσταση τους λόγω

κάποιας σταθερής συνθήκης. Στην περίπτωση αυτή είναι πιο αποδοτικό να σταματήσει να δέχεται τους παλμούς ρολογιού, με αποτέλεσμα να μειώνεται κατά πολύ μεγάλο ποσοστό η κατανάλωση ισχύος του κυκλώματος. Ένα πολύ χαρακτηριστικό παράδειγμα είναι ένα Flip Flop του οποίου η είσοδος, επομένως και η έξοδος παραμένει σταθερή λόγω ενός σήματος ενεργοποίησης. Σε αυτήν την περίπτωση θα μπορούσαμε να κάνουμε το ρολόι να συμπεριφέρεται σε συμφωνία με το σήμα ενεργοποίησης προκειμένου να μειώσουμε τη δυναμική κατανάλωση ισχύος, όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 4. Απλό παράδειγμα χρήσης φραγής ρολογιού.

Προκειμένου να σχεδιάσουμε ένα κύκλωμα με clock gating έχουμε την επιλογή είτε να αφήσουμε αυτόματα το εργαλείο να αναλύσει το κύκλωμα και να διαλέξει μόνο του πότε και ποιοί καταχωρητές είναι ανενεργοί. Ειδάλλως, δίνεται η δυνατότητα να εισαχθούν clock gating κελιά χειροκίνητα, κάτι που απαιτεί εισαγωγή της λογικής αυτής σε κώδικα RTL. Στο εργαστήριο θα δούμε τον πρώτο τρόπο.

Υπάρχουν δύο ειδών Intergrated Clock Gating Cells (ICGC): τα RC Clock Gating Cells και τα Non-RC Clock Gating Cells. Τα πρώτα είναι κελιά τα οποία εξάγονται από το enable pin των flops και συνήθως αρχικοποιούνται με το πρόθεμα "RC_CGIC_INST". Τα δεύτερα είναι CGIC libcells που καλούνται απευθείας από την RTL.

Για να επιτρέψουμε στο Genus να εισάγει από μόνο του Clock Gating Κελιά χρειάζεται να ενεργοποιήσουμε μια μεταβλητή **πριν από το elaboration του κώδικα**, μέσω της παρακάτω εντολής:

genus:root: 19> **`set_db lp_insert_clock_gating`** `true`

Η παραπάνω εντολή επιτρέπει την εισαγωγή Clock Gating Κελιών, ωστόσο τελικά η εισαγωγή τους γίνεται με την εντολή `syn_gen`. Μετά από αυτήν θα ήταν καλό να καλέσουμε την εντολή `report_clock_gating` προκειμένου να δούμε το σύνολο των clock gated flip-flops του κυκλώματος. Σε περίπτωση που δεν είμαστε ικανοποιημένοι με το clock gating που παράχθηκε θα πρέπει να γίνουν αλλαγές σε επίπεδο RTL για περαιτέρω βελτίωση, κάτι που απαιτεί μεγαλύτερη εμπειρία.

3.4. Δημιουργία Περιορισμών για τη Σύνθεση Κυκλώματος

Υπάρχουν πάνω από 100 εντολές για τη δημιουργία περιορισμών, οι οποίοι μπορούν να κατευθύνουν το εργαλείο σύνθεσης. Μερικές από τις εντολές αυτές συζητούνται εδώ και η πλήρη περιγραφή τους εμφανίζεται στο Παράρτημα Α.

Για τη δημιουργία ενός σήματος ρολογιού, χρησιμοποιούμε την εντολή `create_clock`. Τα παρακάτω παραδείγματα δείχνουν τη χρήση της εντολής αυτής.

`create_clock` `[get_ports clkA] -name clk1 -period 5 -waveform {0 2.5}`

- Στο παράδειγμα αυτό δημιουργείται ένα σήμα με ονομασία *clk1* στις θύρες με όνομα *clkA* και περίοδο 5 μονάδες χρόνου. Το ρολόϊ έχει μία ανερχόμενη ακμή στις 0 μονάδες χρόνου και μία κατερχόμενη ακμή στις 2.5 μονάδες χρόνου.

```
create_clock -name abc -period 1000 -comment "mycomment for clock abc"
```

- Το παραπάνω παράδειγμα παρέχει ένα σχόλιο για το το ρολόϊ abc που θα δημιουργηθεί.

```
create_clock -period 1 -name clk1 [get_ports CLK] [get_pins -hierarchical -filter "is_clock_pin==true"]
```

- Στο παραπάνω παράδειγμα δημιουργείται ένα σήμα ρολογιού με ονομασία *clk1* και περίοδο 1 και εφαρμόζεται στις θύρες *CLK* και σε όλους τους ακροδέκτες ρολογιού σε όλη την ιεραρχία του κυκλώματος

`set_clock_latency`

Η εντολή `set_clock_latency` προσδίδει μία καθυστέρηση σε μία πηγή ή δίκτυο ρολογιού για early (hold) ή late (setup) ανάλυση, το οποίο αντιστοιχεί σε ένα ή περισσότερα ρολόγια που μεταδίδονται σε συγκεκριμένους ακροδέκτες ή θύρες. Όταν χρησιμοποιείται αυτή η εντολή, θέτει έναν ακροδέκτη, θύρα, ή ιδιότητα σχετική με καθυστέρηση.

- Η καθυστέρηση του δικτύου διανομής ρολογιού είναι ο χρόνος που απαιτείται από το σήμα ρολογιού από το σημείο ορισμού μέχρι τον ακροδέκτη ρολογιού της ακολουθιακής λογικής.
- Η καθυστέρηση πηγής ρολογιού είναι η καθυστέρηση της πηγής που ορίζεται ως ο χρόνος που απαιτείται από ένα σήμα ρολογιού για να φτάσει από την πηγή του ρολογιού (π.χ. PLL) στο σημείο ορισμού του ρολογιού (π.χ. ακροδέκτη).

Προσδιορίζει για μία πηγή ή ένα δίκτυο ρολογιού, την καθυστέρηση early (σχετιζόμενη με setup) ή late (σχετιζόμενη με hold) για ένα ή περισσότερα ρολόγια που μεταδίδονται σε ένα συγκεκριμένο ακροδέκτη ή θύρα. Η καθυστέρηση αυτή έχει δύο συνιστώσες – την τιμή της πηγής και του δικτύου διανομής.

Η τιμή της **πηγής**:

- Τίθεται χρησιμοποιώντας την παράμετρο -source
- Εφαρμόζεται είτε το ρολόϊ είναι propagated είτε ιδανικό (ideal).
- Η εφαρμογή της τιμής αυτής σε μία θύρα ρολογιού έχει μεγαλύτερη προτεραιότητα από την εφαρμογή της σε μία κυματομορφή ρολογιού.

Η τιμή του **δικτύου** (clock tree delay):

- Η εξ ορισμού τιμή είναι η καθυστέρηση από τη θύρα ρολογιού στον καταχωρητή.
- Εφαρμόζεται μόνο όταν το ρολόϊ είναι ιδανικό. Όταν το ρολόϊ είναι propagated, αυτή η καθυστέρηση αντικαθίστανται από τις πραγματικές καθυστερήσεις του δέντρου ρολογιού.
- Όταν εφαρμόζεται σε μία κυματομορφή ρολογιού ή θύρα ρολογιού αγνοείται αν το ρολόϊ είναι propagated.

Η εντολή `set_clock_latency` μπορεί να χρησιμοποιηθεί με δύο τρόπους:

- Για να τεθεί η τιμή ενός ρολογιού σε μία κυματομορφή ρολογιού:
    - Χρησιμοποιήστε την ακόλουθη σύνταξη για τον ορισμό καθυστέρησης πηγής ρολογιού

set_clock_latency [-source] [-early|-late] [-min][-max] \

value obj_list

- Χρησιμοποιήστε την ακόλουθη σύνταξη για τον ορισμό καθυστέρησης του δικτύου διανομής ρολογιού

set_clock_latency [-min] [-max] value obj_list

- Δεν μπορείτε να θέσετε τις παραμέτρους -late και -early για την τιμή του δικτύου. Βάζοντας μία τιμή σε μία θύρα ή ακροδέκτη αντικαθιστά οποιαδήποτε άλλη τιμή που είχε ανατεθεί στη θύρα ή στην κυματομορφή.
- Το argument obj_list είναι η λίστα των ονομάτων των κυματομορφών ρολογιού που ορίζονται από την εντολή create_clock.

Παράδειγμα:

```
set_clock_latency -clock_gate -0.027 \
[get_pins
{top/u_exfifo_artemis_cpu__RC_CG_HIER_INST34/RC_CGIC_INST/CK}]
-clock [get_clocks {clk}]
```

set_clock_uncertainty

```
set_clock_uncertainty [-from_edge string] [-to_edge string] \
[-from clock_list | -fall_from clock_list | -rise_from clock_list]
[-to clock_list | -fall_to clock_list | -rise_to clock_list]
[-hold | -setup] [-half_cycle_jitter] [-full_cycle_jitter]
uncertainty [clock|port|inst|hinst|pin|hpin]...
```

Προσδιορίζει την αβεβαιότητα του δικτύου διανομής του ρολογιού. Προσδιορίζετε μία αβεβαιότητα είτε για ένα μόνο ρολόϊ (intra) ή για πολλαπλά (inter) ρολόγια.

- Αβεβαιότητες ρολογιού (intra) ορίζονται απευθείας σε ένα σήμα ρολογιού, θύρα, ακροδέκτη, ή εξάρτημα. Αυτές οι τιμές αβεβαιότητας αποθηκεύονται σε ιδιότητες (attributes). Όταν προσδιορίζετε ένα ρολόϊ, αυτό σημαίνει ότι η αβεβαιότητα που δηλώνετε, εφαρμόζετε σε όλα τα ακολουθιακά στοιχεία που χρονίζονται από αυτό το ρολόϊ. Όταν εφαρμόζεται σε μία θύρα ή ακροδέκτη, εφαρμόζεται σε όλα τα σήματα ρολογιού (και τα αντίστοιχα ακολουθιακά τους στοιχεία) που συνδέονται στη θύρα ή ακροδέκτη αυτό.
- Οι αβεβαιότητες μεταξύ πολλαπλών ρολογιών που ορίζονται χρησιμοποιώντας ιδιότητες/επιλογές όπως -from, -to, -rise_from, -rise_to. Αυτές μοντελοποιούνται σαν εξαιρέσεις path_adjust. Αυτές οι αβεβαιότητες προηγούνται των τιμών που δίδονται ως απλή αβεβαιότητα, όταν εφαρμόζονται σε ένα «αντικείμενο ρολογιού» (clock object). Αλλά όταν εφαρμόζεται σε μία θύρα, ακροδέκτη, ή άλλο αντικείμενο, η απλή αβεβαιότητα έχει μεγαλύτερη προτεραιότητα από την αβεβαιότητα μεταξύ ρολογιών (inter-clock). Όταν προσδιορίζετε την αβεβαιότητα μεταξύ ρολογιών (inter-clock), πρέπει να προσδιορίζονται όλοι οι συνδυασμοί αφετηρίας και προορισμού των σημάτων ρολογιού. Για παράδειγμα, αν προδιαγράψετε μονοπάτια από το

CLK1 και CLK2, τότε πρέπει να προδιαγράψετε επίσης την αβεβαιότητα από το CLK2 και CLK1 ακόμα και αν τιμές είναι οι ίδιες.

Παραδείγματα:

- Το ακόλουθο παράδειγμα προδιαγράφει ότι όλα τα μονοπάτια που οδηγούν σε καταχωρητές ή θύρες που χρονίζονται από το ρολόϊ CLKB έχει αβεβαιότητα για την ανάλυση setup 0.5 «sdc» μονάδες χρόνου και αβεβαιότητα για την ανάλυση hold 0.2 «sdc» μονάδες χρόνου.

```
set_clock_uncertainty -setup 0.5 [get_clocks CLKB]
set_clock_uncertainty -hold 0.2 [get_clocks CLKB]
```

Αν υπάρχουν μονοπάτια από άλλα σήματα ρολογιού στο CLKB, τότε αν δεν προδιαγράφεται διαφορετικά, η αβεβαιότητα ενός ρολογιού (intra-clock) επίσης μοντελοποιεί την αβεβαιότητα inter-clock.

- Το ακόλουθο παράδειγμα θέτει μία (intra-clock) αβεβαιότητα για ανάλυση setup 0.4 «sdc» μονάδων χρόνου για όλα τα μονοπάτια που τελειώνουν στον καταχωρητή reg1 και αποθηκεύονται κατά την ανερχόμενη ακμή των ρολογιών που μεταδίδονται στον ακροδέκτη reg1/ CK.

```
set_clock_uncertainty -setup -rise_to 0.4 [get_db pins *reg1/CK]
```

- Η ακόλουθη εντολή θέτει μία αβεβαιότητα (inter-clock) για ανάλυση setup κατά 0.5 «sdc» μονάδων χρόνου για όλα τα μονοπάτια που ξεκινούν από το ρολόϊ clk2 και αποθηκεύονται από το ρολόϊ clk1.

```
set_clock_uncertainty -setup -from clk2 -to clk1 0.5
```

- Η ακόλουθη εντολή θέτει μία αβεβαιότητα (inter-clock) για ανάλυση setup και hold για όλα τα ακολουθιακά στοιχεία που χρονίζονται από τα ρολόγια CLKA.

```
set_clock_uncertainty 0.5 [get_clocks CLKA]
```

set_clock_transition

```
set_clock_transition
{-min | -max} {-rise | -fall} float clock_list
```

Προσδιορίζει τη μετάβαση του ρολογιού (slew) σε συγκεκριμένα ρολόγια. Μπορείτε να προσδιορίσετε τέσσερις τιμές slew: την ελάχιστη ανόδου, την ελάχιστη καθόδου, τη μέγιστη ανόδου, και τη μέγιστη καθόδου. Οι ελάχιστες τιμές δε χρησιμοποιούνται για την ανάλυση χρονισμού αλλά θα εμφανιστούν όταν εκτελέσετε την εντολή `write_sdc`.

set_input_transition

```
set_input_transition {-min | -max} {-rise | -fall} [-clock_fall]
[-clock clock_list] float port_list
```

Η εντολή αυτή θέτει ένα συγκεκριμένο χρόνο μετάβασης για τις συγκεκριμένες θύρες εισόδου και δικατευθυντικές (inout) θύρες. Μπορείτε να ορίσετε το ελάχιστο ανόδου, ελάχιστο καθόδου,

μέγιστο ανόδου, και μέγιστο καθόδου slew. Οι ελάχιστες τιμές δε χρησιμοποιούνται για την ανάλυση χρονισμού αλλά θα εμφανιστούν όταν εκτελέσετε την εντολή `write_sdc`.

set_input_delay

```
set_input_delay [-clock clock] [-clock_fall| -clock_rise] \
[-level_sensitive] [-network_latency_included] \
[-source_latency_included] [-reference_pin {pin|port}...] \
[-rise] [-fall] [-max] [-min] [-add_delay] [-name name] \
[-group_path group] delay {port|pin|hpin|port_bus}...
```

Προσδιορίζει τις θύρες εισόδου και τις δικατευθυντικές θύρες και ακροδέκτες (που είναι έγκυρα start points) του κυκλώματος σε σχέση με μία ακμή του ρολογιού. Αν παραλείψετε τις επιλογές min και max, η καθυστέρηση υποτίθεται ότι εφαρμόζεται και για τις δύο αναλύσεις setup και hold time. Αν παραλείψετε και τις δύο επιλογές -rise και -fall, η καθυστέρηση εφαρμόζεται και για την ανερχόμενη και για την κατερχόμενη ακμή του σήματος εισόδου.

Σημείωση: Η εντολή αυτή θέτει την ιδιότητα external_delays σε συγκεκριμένους ακροδέκτες ή θύρες.

Παραδείγματα:

Για την πρώτη προδιαγραφή καθυστέρησης στη θύρα εισόδου δεν είναι απαραίτητο να προσδιορισθεί η ιδιότητα -add_delay. Αν ο χρήστης εισάγει την ιδιότητα -add_delay στην πρώτη προδιαγραφή, η ιδιότητα αυτή απλά θα αγνοηθεί.

```
set_input_delay -clock CLK1 -min 3000 [get_ports ABC]
set_input_delay -clock CLK1 -max 4000 [get_ports ABC]
```

- Το ακόλουθο παράδειγμα παρακάμπτει τις προηγούμενες αναθέσεις με αναφορά το CLK1

```
set_input_delay -clock CLK2 3500 [get_ports ABC]
```

- Το ακόλουθο παράδειγμα προσθέτει την τιμή καθυστέρησης, χωρίς να απομακρύνει τις υπάρχουσες καθυστερήσεις από προηγούμενους περιορισμούς

```
set_input_delay -clock CLK1 -add_delay 3000 [get_ports ABC]
```

- Το ακόλουθο παράδειγμα εφαρμόζει μία τιμή καθυστέρησης σε σχέση με το σήμα ρολογιού CLK3 για όλες τις εισόδους του κυκλώματος

```
set_input_delay 1000 -clock CLK3 [all_inputs]
```

- Το ακόλουθο παράδειγμα εφαρμόζει μία καθυστέρηση εισόδου σχετικά με την κατερχόμενη ακμή του ρολογιού CLK1

```
set_input_delay -clock CLK1 -clock_fall 2.0 [get_ports ABC]
```

set_output_delay

```
set_output_delay [-clock clock] [-clock_fall] [-clock_rise] \
[-level_sensitive] [-network_latency_included] \
```

```
[-source_latency_included] [-reference_pin pin|port ...] \
[-rise] [-fall] [-max] [-min] [-add_delay] [-name name] \
[-group_path group] delay {port|pin|hpin|port_bus}...
```

Η εντολή αυτή προσδιορίζει τις θύρες εξόδου και τις δικατευθυντικές θύρες, θύρες αρτηριών, και ακροδέκτες (που είναι end points) στο κύκλωμα σχετικά με μία ακμή ρολογιού. Αν παραλείψετε και τις δύο επιλογές -min και -max, το εργαλείο υποθέτει ότι η καθυστέρηση αυτή εφαρμόζεται για την ανάλυση setup και hold. Αν παραλείψετε και τις δύο επιλογές -rise και -fall, η καθυστέρηση εφαρμόζεται και για την ανερχόμενη και την κατερχόμενη ακμή του σήματος εξόδου.

Σημείωση: Η εντολή αυτή θέτει την ιδιότητα external_delay στις συγκεκριμένες θύρες ή ακροδέκτες.

set_load

```
set_load { [-pin_load] [-subtract_pin_load] [-wire_load] port... |
hnet... | port_bus ...} [-min] [-max] capacitance
```

Θέτει τη συγκεκριμένη χωρητικότητα στις ορισμένες θύρες ή διασυνδέσεις του κυκλώματος στο πιο υψηλό επίπεδο (top level design). Η χωρητικότητα που προσδιορίζετε χρησιμοποιώντας αυτήν την εντολή παρακάμπτει τη χωρητικότητα από τα wire load μοντέλα και το SPEF αρχείο.

Για θύρες, η χωρητικότητα που ανατίθεται είναι είτε η χωρητικότητα του εξωτερικού ακροδέκτη όταν η επιλογή -pin_load χρησιμοποιείται ή η εξωτερική χωρητικότητα της διασύνδεσης όταν χρησιμοποιείται η επιλογή -wire_load ή η συνολική χωρητικότητα όταν καμία από τις δύο αυτές επιλογές δεν χρησιμοποιούνται.

Παραδείγματα:

- Η ακόλουθη εντολή θέτει μία ελάχιστη χωρητικότητα ακροδέκτη 1.5 στις θύρες in1, in2

```
set_load 1.5 -min -pin_load [get_ports {in1 in2}]
```

- Η ακόλουθη εντολή θέτει μία ελάχιστη εξωτερική χωρητικότητα διασύνδεσης 1.2 στις θύρες in1, in2

```
set_load 1.2 -min -wire_load [get_ports {in1 in2}]
```

- Η ακόλουθη εντολή θέτει μία μέγιστη εξωτερική χωρητικότητα ακροδέκτη 2.6 στις θύρες in1, in2

```
set_load 2.6 -max -pin_load [get_ports {in1 in2}]
```

- Η ακόλουθη εντολή θέτει μία μέγιστη εξωτερική χωρητικότητα διασύνδεσης 3.0 στις θύρες in1, in2

```
set_load 3.0 -max -wire_load [get_ports {in1 in2}]
```

set_driving_cell

Μοντελοποιεί τη δύναμη οδήγησης ενός εξωτερικού (off-chip) κυκλώματος οδήγησης (driver) που συνδέεται στη θύρα εισόδου. Για να μοντελοποιήσει τη δύναμη οδήγησης του εξωτερικού driver, το εργαλείο ανάλυσης χρονισμού υπολογίζει ένα αντιστάθμισμα (offset) στο χρόνο άφιξης της

εισόδου και αλλάζει το χρόνο μετάβασης που χρησιμοποιείται για τον υπολογισμό της καθυστέρησης του επόμενου κελιού. Το εργαλείο ανάλυσης χρονισμού υπολογίζει το αντιστάθμισμα αφαιρώντας την καθυστέρηση μηδενικού φορτίου από τη συνολική καθυστέρηση του κελιού.

Παράδειγματα:

- Η ακόλουθη εντολή προσδιορίζει το κελί οδήγησης που συνδέεται σε θύρες εισόδου

```
genus:root: number> set_driving_cell -lib_cell BUFX4 [get_db ports
*port_pad_data_in[5]]
```

Ο περιορισμός αυτός παρακάμπτει οποιοδήποτε άλλον περιορισμό από τις εντολές set_drive ή set_input_transition που ορίζονται για την ίδια θύρα. Αν δεν προσδιορίσετε την οδήγηση ή μετάβαση χρησιμοποιώντας τους περιορισμούς αυτούς, το εργαλείο ανάλυσης χρονισμού χρησιμοποιεί μία εξ ορισμού τιμή 0 ps.

Σημείωση: Αποφύγετε τη χρήση των εξ ορισμού τιμών. Γενικά θα πρέπει να ορίσετε κάποιους οδηγούς για τις εισόδους του κυκλώματός σας.

- Η ακόλουθη εντολή δείχνει τον περιορισμό όπου δεν μπορείτε να αναθέσετε έναν buffer να οδηγήσει μία θύρα εξόδου

```
genus:root: number> set_driving_cell -lib_cell BUFX4 [get_db ports
*port_pad_data_out[9]]
```

Warning : The given attribute is not valid on a port of this direction.

[TIM-126]:The attribute 'fixed_slew' cannot be applied to output port 'port_pad_data_out[9]'.

3.5. Διαδοχικά Βήματα για τη Βασική Σύνθεση

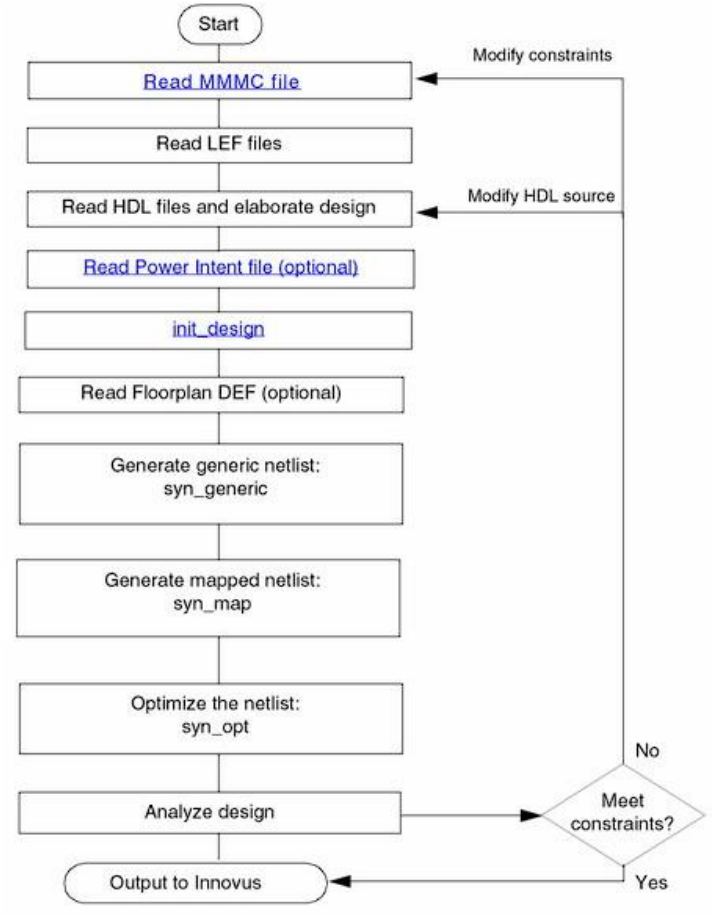
Οι εντολές που απαιτούνται για τα βήματα της σύνθεσης που περιγράφτηκαν παραπάνω, παρουσιάζονται συγκεντρωτικά παρακάτω:

```
#general setup
#-------------
set_db init_lib_search_path path          #optional
set_db init_hdl_search_path path          #optional
#load the library
#----------------------------
set_db library library_name
#enable clock-gating insertion.  (ενότητα 3.3.1)
#----------------------------
set_db lp_insert_clock_gating true
#load and elaborate the design
#-----------------------------
read_hdl design.v
elaborate
#specify timing and design constraints
#-------------------------------------
read_sdc sdc_file                         #optional
#add optimization constraints
#----------------------------                  #optional
.....
#synthesize the design
#--------------------
```

```
syn_generic
syn_map
#analyze design                              #optional steps
-----------------
report_area
report_timing
report_gates

#export design
#-------------                                      #optional steps
write_hdl > design.v
write_sdc > constraints.sdc
write_script > constraints.g

# export design for Innovus
#----------------------
write_design [-basename string] [-gzip_files] [-tcf] [-innovus] [-hierarchical]
[design]
```

3.6. MMMC

Με το ακρωνύμιο MMMC (Multi-Mode Multi-Corner) αναφερόμαστε σε ένα σύνολο ρυθμίσεων οι οποίες περιγράφουν διαφορετικές συνθήκες λειτουργίας και διεργασίας κατασκευής. Αποτελούν έναν εύχρηστο τρόπο ομαδοποίησης ρυθμίσεων και μεταβλητών ανάλογα με το στάδιο και τις ανάγκες της υλοποίησης. Η ροή σχεδίασης που ακολουθείται σε αυτήν την περίπτωση είναι ελαφρώς διαφορετική και παρουσιάζεται στο Σχήμα 5.



Σχήμα 5 Ροή Σχεδίασης με χρήση MMMC αρχείου

Με τον όρο Mode αναφερόμαστε σε ρυθμίσεις οι οποίες απευθύνονται σε διαφορετικά στάδια της κατασκευής, για παράδειγμα ένα κύκλωμα που βρίσκεται σε στάδιο ελέγχου (testing) ή ένα λειτουργικό κύκλωμα που είναι πολύ παρόμοιο με την τελική παραγωγή (production). Ακόμη μπορεί να αναφερόμαστε σε διαφορετικούς στόχους σε κάθε σχεδίαση, όπως για παράδειγμα μια σχεδίαση με στόχο την επίτευξη ρολογιού ύψιστης συχνότητας είτε της μικρότερης κατανάλωσης ενέργειας.

Με τον όρο Corner αναφερόμαστε σε επιλογές που περιγράφουν μεταβολές των συνθηκών κατασκευής καθώς ακόμη και της τάσης και της θερμοκρασίας. Με αυτόν τον τρόπο επιτυγχάνεται ο έλεγχος του κυκλώματος κάτω από διαφορετικές συνθήκες λειτουργίας.

Για να ορίσουμε διαφορετικά Corners για τη σχεδίασή μας χρησιμοποιούμε τις εντολές: `create_rc_corner` και `create_delay_corner`. Η πρώτη δημιουργεί διαφορετικά RC Corners τα οποία χρησιμεύουν στο εργαλείο για τον υπολογισμό της καθυστέρησης λόγω αντιστάσεων και χωρητικοτήτων. Το δεύτερο δημιουργεί διαφορετικά Corners υπολογισμού καθυστέρησης μέσα στα οποία μπορεί κανείς να ορίσει διαφορετικά RC Corners, καθώς ακόμη και διαφορετικές βιβλιοθήκες χρονισμού.

Για να δημιουργήσουμε διαφορετικά Modes για τα παραπάνω corners χρησιμοποιούμε την εντολή: `create_constraint_mode`.

Κάνοντας το συνδυασμό κάθε Mode και κάθε Corner παίρνουμε ένα διαφορετικό σενάριο για να εξετάσουμε ξεχωριστά. Κάθε διαφορετικό σενάριο, που ονομάζεται View, πρέπει να δηλωθεί στο MMMC αρχείο. Η εντολή με την οποία γίνεται αυτό είναι η εξής: `create_analysis_view`.

Στο τέλος, και επειδή μπορεί σε διαφορετικές επαναλήψεις να θέλουμε να ασχοληθούμε με κάποιο διαφορετικό View πρέπει να ορίσουμε εκείνα τα οποία μας ενδιαφέρουν. Αυτό που είναι σημαντικό είναι η δήλωση αυτή να γίνει μετά τη δημιουργία των Views και όχι πιο πριν με την εντολή: `set_analysis_view`.

Μέσω του MMMC αρχείου δίνεται η δυνατότητα να οριστούν και βιβλιοθήκες χρονισμού ανάλογα με τις διαφορετικές τεχνολογίες κατασκευής που μας δίνονται. Ένα πολύ σύνηθες παράδειγμα χρήσης είναι η δημιουργία 2 σετ βιβλιοθηκών χρονισμού, εκ των οποίων η μια να χρησιμοποιεί πιο γρήγορες πύλες από την άλλη. Προκειμένου να οριστούν αυτά χρησιμοποιούνται οι εντολές: `create_library_set` για τον ορισμό της βιβλιοθήκης, `create_opcond` για την δημιουργία συνθηκών λειτουργίας και `create_timing_condition` για τον τελικό συνδυασμό βιβλιοθηκών και συνθηκών λειτουργίας.

Παρακάτω ακολουθεί ένα template αρχείο με τη δομή που θα είχε ένα mmmc.tcl αρχείο.

```
create_library_set \
      -name wcl_slow \
      -timing {../libraries/timing/slow_vdd1v0_basicCells.lib}


create_opcond \
      -name op_cond_wcl_slow
      -process 45 \
      -voltage 1.0 \
      -temperature 25
```

```
create_timing_condition \
      -name timing_cond_wcl_slow \
      -opcond op_cond_wcl_slow    \
      -library_sets { wcl_slow }

create_rc_corner -name default_rc_corner \
    -pre_route_res 1.0 \
    -pre_route_cap 1.0 \
    -pre_route_clock_res 0.0 \
    -pre_route_clock_cap 0.0 \
    -post_route_res {1.0 1.0 1.0} \
    -post_route_cap {1.0 1.0 1.0} \
    -post_route_cross_cap {1.0 1.0 1.0} \
    -post_route_clock_res {1.0 1.0 1.0} \
    -post_route_clock_cap {1.0 1.0 1.0}

create_delay_corner \
      -name delay_corner_wcl_slow \
      -early_timing_condition timing_cond_wcl_slow  \
      -late_timing_condition timing_cond_wcl_slow \
      -early_rc_corner default_rc_corner \
      -late_rc_corner default_rc_corner

create_constraint_mode \
      -name functional_wcl_slow \
      -sdc_files {../constraints/constraints.sdc}

create_analysis_view \
      -name view_wcl_slow \
      -constraint_mode functional_wcl_slow \
      -delay_corner delay_corner_wcl_slow

set_analysis_view \
      -setup { view_wcl_slow } \
      -hold {view_wcl_slow}
```

Τέλος για να διαβάσουμε το αρχείο στο genus χρησιμοποιούμε την εντολή `read_mmmc`. Με αυτόν τον τρόπο δεν χρειάζεται να ορίσουμε ποια βιβλιοθήκη χρονισμού χρησιμοποιούμε (`set_db library library_name`), ούτε να διαβάσουμε σε ξεχωριστό βήμα τους περιορισμούς SDC (`read_sdc`). Όλα γίνονται συγχρόνως κατά την ανάγνωση του MMMC αρχείου. Τέλος μετά την ανάγνωση του MMMC αρχείου γίνεται η κλήση της εντολής init_design.

```
#general setup
#-------------
set_db init_lib_search_path path          #optional
set_db init_hdl_search_path path          #optional
set_db lef_library lef_library

#enable clock-gating insertion.  (ενότητα 3.3.1)
#----------------------------
set_db lp_insert_clock_gating true
#load MMMC file, elaborate and initialize the design
#-----------------------------
read_mmmc mmmc.tcl

read_hdl

elaborate
init_design

#specify timing and design constraints
#-------------------------------------
read_sdc sdc_file                         #optional
#add optimization constraints
#----------------------------                  #optional
.....
#synthesize the design
#--------------------
syn_generic
syn_map
#analyze design                           #optional steps
-----------------
report_area
report_timing
report_gates

#export design
#-------------                                 #optional steps
write_hdl > design.v
write_sdc > constraints.sdc
write_script > constraints.g

# export design for Innovus
#----------------------
write_design [-basename string] [-gzip_files] [-tcf] [-innovus] [-hierarchical]
```

4. Logic Equivalence Check (LEC)

4.1. Εισαγωγή

Κατά τη σύνθεση του κώδικα μέσω του Genus θέλουμε να επιβεβαιώσουμε ότι δεν αλλοιώθηκε το κύκλωμα από την αρχική μορφή του σε κώδικα RTL σε επίπεδο λειτουργικότητας (synthesized netlist). Για το λόγο αυτό χρειαζόμαστε εργαλεία τα οποία να μπορούν να συγκρίνουν το κύκλωμα σε κάθε βήμα της ροής σχεδίασης σε σχέση με την αρχική περιγραφή RTL. Αυτό επιτυγχάνει το εργαλείο της Cadence με όνομα Conformal Equivalence Checking.

4.2 Εντολές

Η σύγκριση γίνεται σταδιακά ανάμεσα στα σημεία που έχει φτάσει ο συνθέσιμος κώδικας. Οι συγκρίσεις που προτείνεται να γίνουν είναι οι ακόλουθες:

- Γλώσσα περιγραφής υλικού (RTL) -> Elaborated Code
- Elaborated Code -> syn_generic Code
- syn_generic Code -> syn_map Code

Για να γίνει η σύγκριση αυτή είναι σημαντικό να αποθηκεύεται το κάθε ενδιάμεσο βήμα κατά τη διάρκεια της σύνθεσης. Αυτό είναι εφικτό με την παρακάτω εντολή:

```
write_netlist -lec > ${revised_design}
```

```
write_do_lec -top ${top_level_design} -golden_design ${golden_design} -revised_design ${revised_design} -log_file ${golden-revised}.lec.log > ${golden-revised}.do
```

Το golden design είναι η βάση αναφοράς ενώ το revised design είναι ο κώδικας του οποίου την λογική ισοδυναμία θέλουμε να ελέγξουμε σε σχέση με το golden.

Για παράδειγμα, για τις συγκρίσεις παραπάνω θα θέλαμε έναν κώδικα τέτοιας μορφής:

```
...


# Elaboration
elaborate ${top_level_design}


# Verification rtl vs elab step
write_netlist -lec > elab.v
write_do_lec -top ${top_level_design} -golden_design rtl -revised_design elab.v -log_file rtl_elab.lec.log > rtl_elab.do


...


syn_generic
# Verification elab vs generic step
write_netlist -lec > generic.v
write_do_lec -top ${top_level_design} -golden_design elab.v -revised_design generic.v -log_file elab_generic.lec.log > elab_generic.do
```

```
...
```

syn_map

```
# Verification generic vs fv_map step
write_do_lec -top ${top_level_design} -golden_design generic.v -revised_design fv_map -log_file generic_fvmap.lec.log > generic_fvmap.do

...
```

Έτσι με αυτόν τον τρόπο έχουμε καταφέρει να αποθηκεύσουμε σε 3 αρχεία .do, τρεις συγκρίσεις netlist σε διαφορετικές στιγμές της σύνθεσης. Τώρα μένει να αξιοποιήσουμε αυτά τα αρχεία με το Conformal Equivalence Checking. Ανοίγουμε ένα καινούριο terminal και εκτελούμε:

```
lec -XL -nogui -dofile example.do
```

όπου αντί για example.do χρησιμοποιούμε το όνομα των αρχείων όπως τα εξάγαμε από το Genus.

Το αποτέλεσμα αυτής της ανάλυσης είναι ένα τεράστιο output, το κυριότερο όμως κομμάτι του βρίσκεται στο Verification Report και ένα παράδειγμα φαίνεται παρακάτω:

```
// Command: report_verification (-hier) -verbose
```

```
================================================================================
                            Verification Report
================================================================================
Category                                                                  Count
================================================================================
1. Non-standard modeling options used:                                      0
     Tri-stated output:                                   checked
     Revised X signals set to E:                          yes
     Floating signals tied to Z:                          yes
     Command "add clock" for clock-gating:                not used
================================================================================
2. Incomplete verification:                                                 0
     All primary outputs are mapped:                      yes
     Not-mapped DFF/DLAT is detected:                     no
     All mapped points are added as compare points:       yes
     All compared points are compared:                    yes
     User added black box:                                no
     Black box mapped with different module name:         no
     Empty module is not black boxed:                     no
     Command "add ignore outputs" used:                   no
     Always false constraints detected:                   no
================================================================================
3. User modification to design:                                             0
     Change gate type:                                    no
     Change wire:                                         no
     Primary input added by user:                         no
================================================================================
4. Conformal Constraint Designer clock domain crossing checks recommended:  0
     Multiple clocks in the design:                       no
================================================================================
5. Design ambiguity:                                                        1
     Duplicate module definition:                         yes *
     Black box due to undefined cells:                    no
     Golden design has abnormal ratio of unreachable gates:    no
     Ratio of golden unreachable gates:                   0%
     Revised design has abnormal ratio of unreachable gates:   no
     Ratio of revised unreachable gates:                  0%
     All primary input bus ordering is consistent:        yes
     All primary output bus ordering is consistent:       yes
================================================================================
6. Compare Results:                                                      PASS
     Number of EQ compare points:                         1326
     Number of NON-EQ compare points:                     0
     Number of Aborted compare points:                    0
     Number of Uncompared compare points :                0
```

5. Σχεδίαση σε φυσικό επίπεδο (Implementation)

Η ενότητα αυτή περιγράφει τα βήματα που απαιτούνται για τη διαδικασία της σχεδίασης ενός ολοκληρωμένου κυκλώματος σε φυσικό επίπεδο με το εργαλείο innovus. Εντολές σχετικές με την εκκίνηση του εργαλείου παρέχονται στην υποενότητα 5.1. Τρόποι για την παροχή βοήθειας ως προς τη χρήση του εργαλείου παρέχονται στην υποενότητα 5.2. Η συνολική περιγραφή της διαδικασίας σχεδίασης παρουσιάζεται στην υποενότητα 5.3.

5.1. Εκκίνηση του εργαλείου

Το εργαλείο σχεδίασης των κυκλωμάτων σε φυσικό επίπεδο, των οποίων έχει προηγηθεί η σύνθεσή τους με το Genus, ονομάζεται innovus και καλείται από ένα τερματικό με την εντολή:

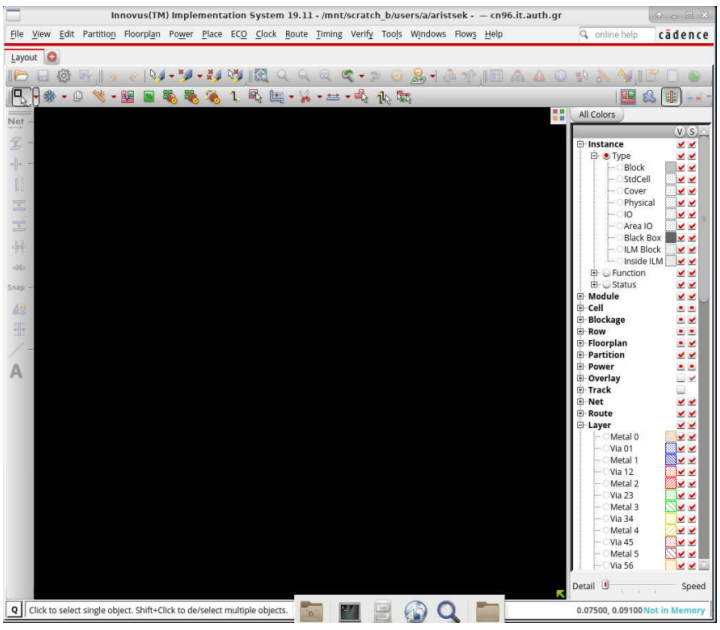
```
innovus
```

Το εργαλείο τυπώνει κάποια πληροφορία σχετικά π.χ. με τον τύπο άδειας που καλείται και την έκδοσή του κλπ και, εν συνεχεία, εμφανίζεται η γραμμή εργασίας:

> Κάθε φορά που καλείται το εργαλείο δημιουργεί αρχεία εξόδου, όπως και στην περίπτωση του Genus, με καταλήξεις xx.cmd, xx.log και xx.logv, όπου xx είναι ο αύξων αριθμός των αρχείων. Τα αρχεία με κατάληξη .cmd περιέχουν όλες τις εντολές που εκτελέσατε στη γραμμή εντολών ή μέσω του gui για κάθε συνεδρία, τα αρχεία .log περιέχουν διάφορα μηνύματα και πληροφορίες σχετικά με τη συνεδρία (session), ενώ τα αρχεία .logv περιέχει περισσότερες λεπτομέρειες για ό,τι εκτελέστηκε και είναι χρήσιμο για αποσφαλμάτωση (debugging).

```
innovus 1>
```

Το γραφικό περιβάλλον του εργαλείου εικονίζεται στο Σχήμα 6.



Σχήμα 6. Γραφικό περιβάλλον του εργαλείου σχεδίασης innovus®.

5.2. Βοήθεια για τη χρήση του εργαλείου

Στη γραμμή εντολής εμφανίζονται διάφορα μηνύματα, τα οποία μπορεί να είναι απλές προειδοποιήσεις (**WARN) ή σφάλματα (**ERROR), ανάλογα με τις εντολές και λειτουργίες που εκτελούνται, μετά το πέρας της εντολής που δώσαμε. Τα πρώτα δεν αποτελούν συνήθως σοβαρό

πρόβλημα, ωστόσο κάποιες προειδοποιήσεις μπορούν να επηρεάσουν τα επόμενα στάδια της σχεδίασης, οπότε είναι καλό να υπάρχει γνώση τους. Τα σφάλματα είναι σημαντικά και πρέπει πάντοτε να επιλύονται προτού προχωρήσει η σχεδίαση. Όπως και στην περίπτωση του Genus, για να ληφθούν περισσότερες πληροφορίες για ένα είδος προειδοποίησης ή σφάλματος, χρησιμοποιούμε την εντολή help ως εξής:

```
innovus 1> help IMPEXT-6197
```

όπου IMPEXT-6197 είναι το όνομα της προειδοποίησης. Επίσης, υπάρχουν αρκετοί τρόποι για να λάβετε βοήθεια σε διάφορα θέματα όπως η σύνταξη εντολών και argument αυτών. Κάποιοι από αυτούς τους τρόπους περιγράφονται εδώ. Ο πιο απλός τρόπος είναι με τη χρήση της εντολής help *command_name*. Για παράδειγμα για την εντολή timeDesign:

```
innovus 2> help timeDesign
```

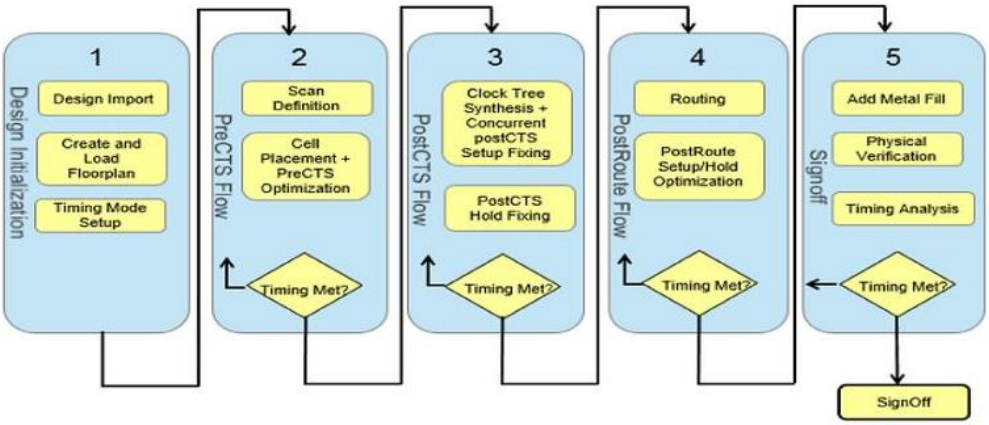
επιστρέφει τη σύνταξη της αντίστοιχης εντολής. Εναλλακτικά, αν δεν είστε σίγουροι για τη σύνταξη μιας εντολής, μπορείτε να την αναζητήσετε χρησιμοποιώντας μόνο μέρος αυτής και να πατήσετε το πλήκτρο <Tab>, ομοίως με το Genus. Το τύπωμα κάποιων χαρακτήρων και το πλήκτρο <Tab> εμφανίζουν όλες τις εντολές που περιέχουν τους χαρακτήρες αυτούς. Για παράδειγμα:

```
innovus 3> time <Tab>
```

Επιστρέφει: `time timeDesign timestamp`

5.3. Βήματα της Διαδικασίας

Τα βήματα της ροής σχεδίασης σε φυσικό επίπεδο μέσω του innovus παρουσιάζονται στο Σχήμα 7, όπου υπάρχουν πέντε στάδια. Το πρώτο αποτελεί την αρχικοποίηση του κυκλώματος, όπου κάνουμε εισαγωγή του design (κυκλώματος) στο εργαλείο, χωροθέτηση, και επιλογή των ρυθμίσεων για τους χρονικούς υπολογισμούς. Το επόμενο στάδιο ονομάζεται PreCTS και αποτελεί το βήμα πριν τη σύνθεση του δέντρου ρολογιού, όπου εισάγουμε τα σκαν φλιπ φλοπ και εκτελούμε την τοποθέτηση. Στο στάδιο μετά τη σύνθεση του δέντρου ρολογιού (PostCTS) δημιουργούμε το δέντρο ρολογιού σε φυσικό επίπεδο και εκτελούμε διάφορες βελτιστοποιήσεις για να διορθωθούν τυχόν αρνητικά slack στους χρόνους προετοιμασίας και διατήρησης. Το τέταρτο στάδιο ονομάζεται PostRoute και αποτελεί το στάδιο μετά τη δρομολόγηση, όπου δρομολογούνται όλες οι συνδέσεις εσωτερικά του κυκλώματος και γίνεται για άλλη μια φορά βελτιστοποίηση ως προς τα slack. Το στάδιο αυτό αποτελεί το πιο χρονοβόρο για μεγάλα κυκλώματα. Τέλος, στο στάδιο του SignOff, εκτός της επαλήθευσης χρονισμού, εισάγουμε ένα είδος κελιού που ονομάζεται “Fill” για την ικανοποίηση της πυκνότητας κάθε μετάλλου στο φυσικό σχέδιο, εφαρμόζουμε DRC (Design Rule Check) για να ελέγξουμε αν υπάρχουν παραβάσεις στους κανόνες σχεδίασης της βιβλιοθήκης που χρησιμοποιούμε και εκτελούμε μια τελευταία ανάλυση χρονισμού. Όλα τα παραπάνω στάδια περιγράφονται στις επόμενες υποενότητες.



Σχήμα 7. Η αλληλουχία των βημάτων χρησιμοποιώντας το εργαλείο του innovus®.

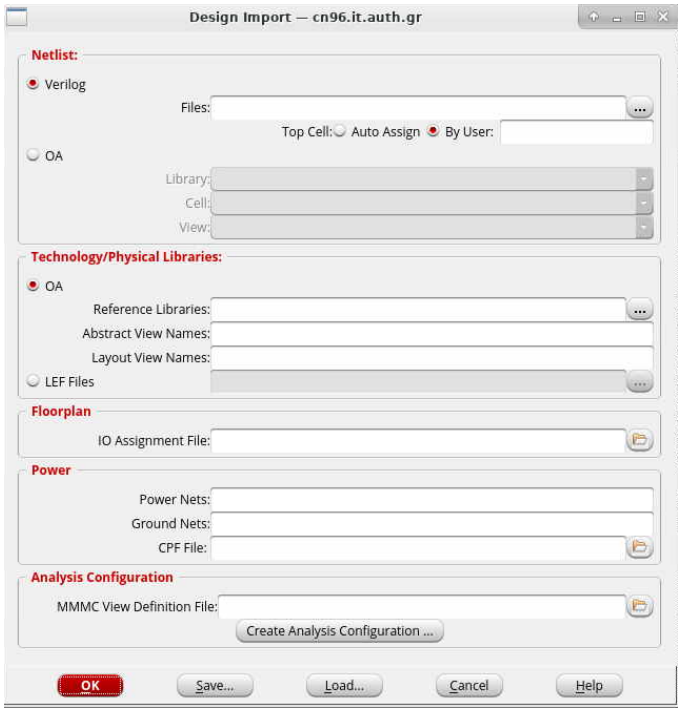
5.3.1. Αρχικοποίηση του κυκλώματος (Design Initialization)

Το πρώτο βήμα αποτελεί η αρχικοποίηση του κυκλώματος, η οποία περιλαμβάνει την εισαγωγή των απαραίτητων αρχείων που παράχθηκαν από το genus μαζί με τις βιβλιοθήκες (**Design Import**), τη χωροθέτηση (**Floorplan**), καθώς και τη ρύθμιση του τρόπου υπολογισμού χρονισμού στο κύκλωμα (**Timing Mode Setup**).

Η εισαγωγή των απαραίτητων αρχείων μπορεί να γίνει μέσω του gui με την επιλογή File → Import Design από τη γραμμή εργαλείων. Η επιλογή αυτή θα ανοίξει ένα ξεχωριστό παράθυρο που εικονίζεται στο Σχήμα 8.

Ξεκινώντας, συμπληρώνουμε το πεδίο Netlist, επιλέγοντας το αρχείο Verilog που προέκυψε από το genus, χρησιμοποιώντας το κουμπί δίπλα από το πεδίο Files. Αυτό θα ανοίξει ένα ξεχωριστό παράθυρο, όπου θα πρέπει να γίνει μετάβαση στο κατάλληλο directory και να επιλεχθεί το αρχείο αυτό, το οποίο θα έχει ονομασία genus.v, *εφόσον ακολουθήθηκαν ακριβώς τα βήματα για τη εξαγωγή των απαραίτητων αρχείων κατά τη σύνθεση*. Μόλις γίνει η εισαγωγή του αρχείου αυτού, είναι δυνατή είτε η επιλογή αυτόματης ανάθεσης του κορυφαίου module (Top Cell), είτε η ανάθεσή του από τον ίδιο το χρήστη (By user:).

Στη συνέχεια, γίνεται συμπλήρωση του πεδίου των βιβλιοθηκών, όπου μόλις επιλέξουμε το LEF Files, το πεδίο δίπλα θα ενεργοποιηθεί και με παρόμοιο τρόπο μπορεί να γίνει η επιλογή του αρχείου LEF της βιβλιοθήκης που χρησιμοποιούμε, κάνοντας μετάβαση στο κατάλληλο directory.



Σχήμα 8. Το παράθυρο εισαγωγής του design στο innovus®.

Αν επιθυμείτε να προστεθούν στο υπό σχεδίαση κύκλωμα, ακροδέκτες (I/O pads) για τη συσκευασία του κυκλώματος, τότε στο πεδίο Floorplan μπορούμε να προσθέσουμε ένα αρχείο, το οποίο περιλαμβάνει πληροφορίες σχετικά με τον τρόπο τοποθέτησης των I/O στην περιφέρεια του φυσικού σχεδίου. Αν δεν επιθυμείτε την προσθήκη των ακροδεκτών τότε το βήμα αυτό παραλείπεται. Η διαδικασία της προσθήκης των pads περιγράφεται σε ξεχωριστή ενότητα παρακάτω.

Στο πεδίο Power πρέπει να σημειωθεί η κατάλληλη ονομασία των δικτύων τροφοδοσίας και γείωσης, η οποία θα παίξει ρόλο στο επόμενο στάδιο της σχεδίασης. Επίσης, μπορεί να προστεθεί το αρχείο CPF (Common Power Format), το οποίο περιέχει πληροφορίες σχετικά με τη δομή του δικτύου ισχύος, εφόσον αυτό είναι διαθέσιμο. Για τα πλαίσια του παρόντος εργαστηρίου δεν είναι απαραίτητη η προσθήκη του.

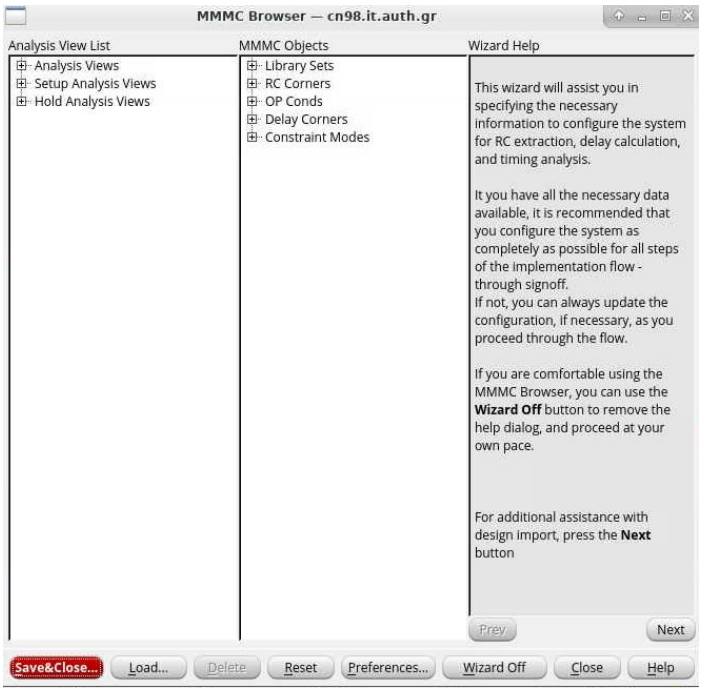
Verilog files → Περιέχει τα ονόματα των Verilog αρχείων που έχουν πληροφορίες για το Netlist σε επίπεδο πύλης.

LEF files → Βιβλιοθήκη με στοιχεία και δεδομένα φυσικού επιπέδου για τα εξαρτήματα σε format .lef που περιέχει επίσης τα επίπεδα (μετάλλου) δρομολόγησης και τους DRC κανόνες.

IO Assignment file → Αυτό το αρχείο περιέχει πληροφορίες σχετικά με τη σειρά των I/O για να δώσει στο εργαλείο τη δυνατότητα να τοποθετήσει τα pads στην περιφέρεια του design. Αν δε δοθεί το συγκεκριμένο αρχείο, τότε το εργαλείο θα τοποθετήσει τα I/O στην περιφέρεια του κυκλώματος με τυχαίο τρόπο.

MMMC View Definition file → Περιέχει δείκτες στη βιβλιοθήκη χρονισμού (Liberty) και το αρχείο περιορισμών SDC.

Τέλος, πρέπει να δημιουργηθεί το κατάλληλο MMMC (multi-mode, multi-corner) αρχείο, για αυτό το λόγο στο τελευταίο βήμα της εισαγωγής, χρησιμοποιούμε την επιλογή Create Analysis Configuration, η οποία θα ανοίξει επίσης ένα διαφορετικό παράθυρο που φαίνεται στο Σχήμα 9.



Σχήμα 9. Παράθυρο για τον ορισμό MMMC.

Η συμπλήρωση των απαραίτητων πληροφοριών για τη δημιουργία του συγκεκριμένου αρχείου βασίζεται στο αρχείο genus.mmmc.tcl που δημιουργήθηκε από το genus και βρίσκεται στον ίδιο φάκελο με το αρχείο του Netlist (genus.v). Η δημιουργία του εκάστοτε χαρακτηριστικού (MMMC Object ή Analysis View) γίνεται κάνοντας διπλό κλικ σε οποιοδήποτε από αυτά στη λίστα, όπως φαίνεται στο Σχήμα 9.

Αρχικά, πρέπει να γίνει συμπλήρωση των απαραίτητων συνόλων βιβλιοθηκών (Library Sets). Με διπλό κλικ θα ανοίξει ένα καινούριο παράθυρο, όπου πρέπει να συμπληρωθεί το όνομα του συνόλου, καθώς και να προστεθεί η βιβλιοθήκη χρονισμού (Timing Library Files) που θέλουμε να χρησιμοποιήσουμε. Στη συνέχεια, γίνεται η συμπλήρωση των RC Corners με παρόμοιο τρόπο, όπου προσθέτουμε το όνομα του Corner που μελετάμε, τη θερμοκρασία, ένα σύνολο από factors και τα αρχεία Cap Table ή QRC (*.tch). Τις απαραίτητες πληροφορίες σχετικά με τη θερμοκρασία και τα factors μπορούμε να τις πάρουμε άμεσα από το αρχείο genus.mmmc.tcl.

Σε δεύτερο στάδιο, αρχικοποιούμε τις καταστάσεις λειτουργίας (OP Conds), όπου πρέπει να συμπληρώσουμε το όνομα της κατάστασης, τα PVT (Process, Voltage, Temperature), αλλά και τη βιβλιοθήκη χρονισμού που χρησιμοποιούμε (Library File). Όπως αναφέρθηκε και προηγουμένως, οι απαραίτητες πληροφορίες για το PVT βρίσκονται στο αρχείο genus.mmmc.tcl.

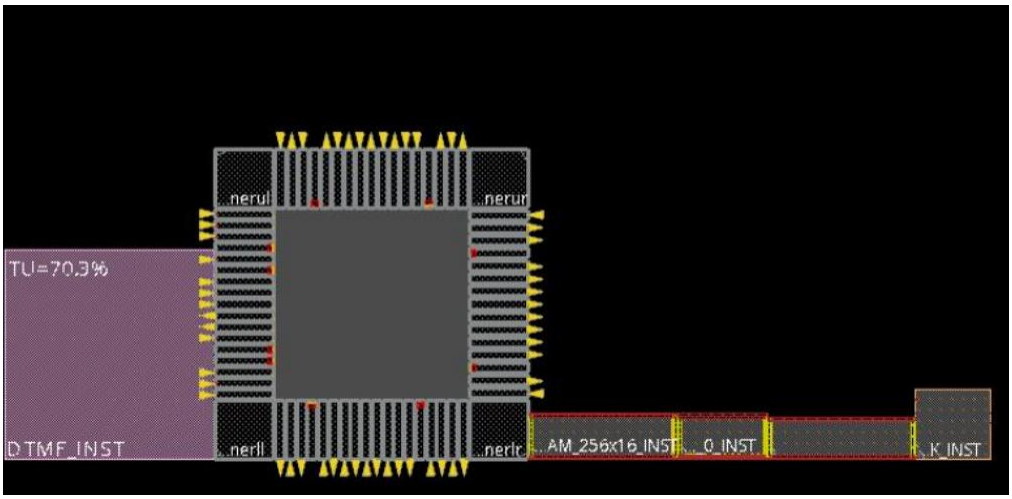
Εφόσον γίνουν οι παραπάνω διεργασίες, μπορούμε να ξεκινήσουμε τη δημιουργία των Delay Corners, τα οποία χρειάζονται τρεις βασικές πληροφορίες για συμπλήρωση. Αυτές είναι το όνομα, το σύνολο βιβλιοθηκών που δημιουργήθηκε, καθώς και το αντίστοιχο RC Corner. Ως επιλογή τύπου, αφήνουμε την default επιλογή Single/BCWC (εμείς θα εξετάσουμε ένα mode μόνο, συνεπώς μελετάμε την περίπτωση single). Τα υπόλοιπα πεδία αφήνονται κενά.

Τέλος, για τους περιορισμούς (Constraint Modes), σε αυτά πρέπει να γίνει προσθήκη του ονόματος, καθώς και του αρχείου sdc που παράχθηκε από το genus και βρίσκεται στον κατάλληλο φάκελο που προέκυψε μετά από εξαγωγή (genus.default_emulate_constraint_mode.sdc).

Αφού ολοκληρωθούν τα παραπάνω βήματα, θα πρέπει να υπάρχει ένα αντικείμενο σε καθένα από τα πέντε διαφορετικά MMMC Objects. Ως τελικό στάδιο, επιλέγουμε το Analysis Views, όπου θα πρέπει να προστεθεί το κατάλληλο όνομα του view που θέλουμε να δημιουργήσουμε, καθώς και το Constraint Mode και Delay Corner, τα οποία φτιάξαμε. Μόλις δημιουργηθεί το view, δημιουργούμε επίσης ένα Setup/Hold Analysis View (σε αυτή την περίπτωση χρειάζεται μόνο να επιλέξουμε το view που δημιουργήθηκε προηγουμένως στο Analysis View).

Μόλις τελειώσουμε με τα παραπάνω, πηγαίνουμε στην επιλογή Save&Close και αποθηκεύουμε το αρχείο σε μορφή .view στον προορισμό που θέλουμε. Στη συνέχεια, το παράθυρο του Σχήμα 9 θα κλείσει και θα εμφανιστεί το αρχείο στο πεδίο MMMC View Definition File. Έπειτα μπορούμε να τελειώσουμε την εισαγωγή πατώντας OK. Αυτό θα οδηγήσει στην εμφάνιση ενός πλέγματος στο γραφικό περιβάλλον του innovus, εφόσον φυσικά έχουν διεκπεραιωθεί σωστά όλες οι διεργασίες.

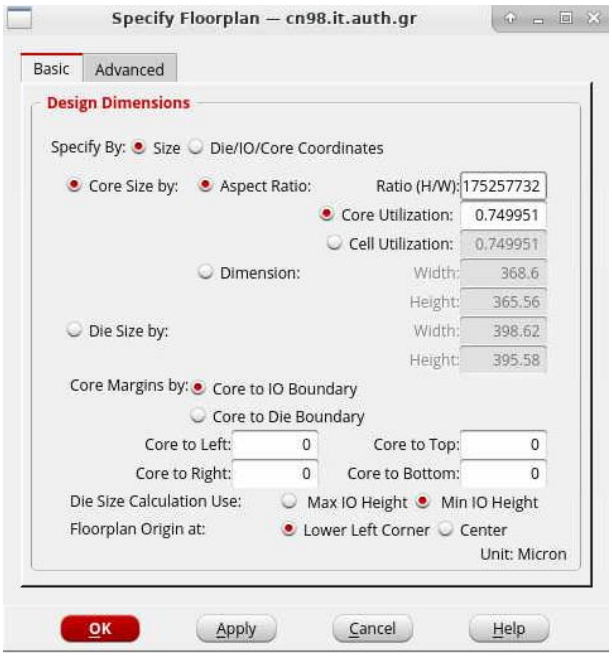
Αν θέλουμε να προσαρμόσουμε το design στο παράθυρο, τότε πρέπει να πατήσουμε το πλήκτρο f. Επίσης, στην πάνω δεξιά μεριά του εργαλείου μπορούν να φανούν τέσσερις επιλογές σχετικά με τον τρόπο εμφάνισης του φυσικού σχεδίου. Από αριστερά προς τα δεξιά, η πρώτη επιλογή είναι το Floorplan view και μπορεί να χρησιμοποιηθεί για να μελετήσουμε το design με τρόπο που φαίνεται στο Σχήμα 10. Η δεύτερη επιλογή ονομάζεται Amoeba view και μπορεί να επιλεγεί μετά την τοποθέτηση για να παρατηρήσουμε τον τρόπο που έγινε η ανάθεση των partitioned τμημάτων πάνω στο φυσικό σχέδιο. Η τρίτη επιλογή είναι το Physical view και χρησιμοποιείται για να μελετηθεί το design σε επίπεδο μετάλλων και η τέταρτη επιλογή είναι η Layout 3D και μόλις επιλεγεί θα μεταβούμε σε ένα νέο παράθυρο που θα περιλαμβάνει ότι φαίνεται εκείνη τη στιγμή στο κυρίως παράθυρο σε μορφή τριών διαστάσεων (**Προσοχή:** Η βέλτιστη λύση είναι να επιλέγεται κάθε φορά ένα σημείο του κυκλώματος για 3D view και όχι ολόκληρο το κύκλωμα, λόγω του μεγέθους του).



Σχήμα 10. Το Floorplan View ενός design. Αριστερά τα ροζ τετράγωνα αποτελούν τις μονάδες που διαχωρίστηκαν από το partition του genus. Το τετράγωνο στο κέντρο αποτελεί την περιοχή που σχεδιάζεται το design. Τα κελιά στη δεξιά μεριά αποτελούν μπλοκ που μπορούν να προστεθούν μέσα στο κύριο design.

Το επόμενο βήμα της αρχικοποίησης αποτελεί η χωροθέτηση, η οποία μπορεί να εκτελεστεί μέσω της επιλογής **Floorplan → Specify Floorplan**. Όπως φαίνεται στο Σχήμα 11, μέσω του συγκεκριμένου παραθύρου είναι δυνατή η ρύθμιση διαφόρων πληροφοριών σχετικά με τις διαστάσεις του πυρήνα. Αυτό μπορεί να επιτευχθεί είτε ρυθμίζοντας το λόγο ύψος/πλάτος (Ratio

H/W) και το ποσοστό χρήσης του πυρήνα (Core Utilization), είτε μέσω άμεσης συμπλήρωσης των διαστάσεων αυτού (Width/Height). Όσον αφορά τα Core Margins, η επιλογή αυτή αφορά την προσθήκη ενός καναλιού ανάμεσα στα περιθώρια του πυρήνα και των I/O και δημιουργείται για την προσθήκη των δακτυλίων στο βήμα της σύνθεσης του δικτύου διανομής ισχύος. Έτσι είναι δυνατή η ρύθμιση του μεγέθους του κενού σε μικρόμετρα από το περιθώριο του πυρήνα μέχρι τα I/O για κάθε πλευρά του design. Επίσης, στην καρτέλα Advanced μπορεί να γίνουν πιο προχωρημένες επιλογές που αφορούν προσαρμογές πάνω στις γραμμές του πλέγματος, όπως επιλογή κενού ανάμεσα σε αυτές ή αλλαγή του προσανατολισμού της κατώτερης γραμμής.



Σχήμα 11. Το παράθυρο για τη διεργασία της χωροθέτησης.

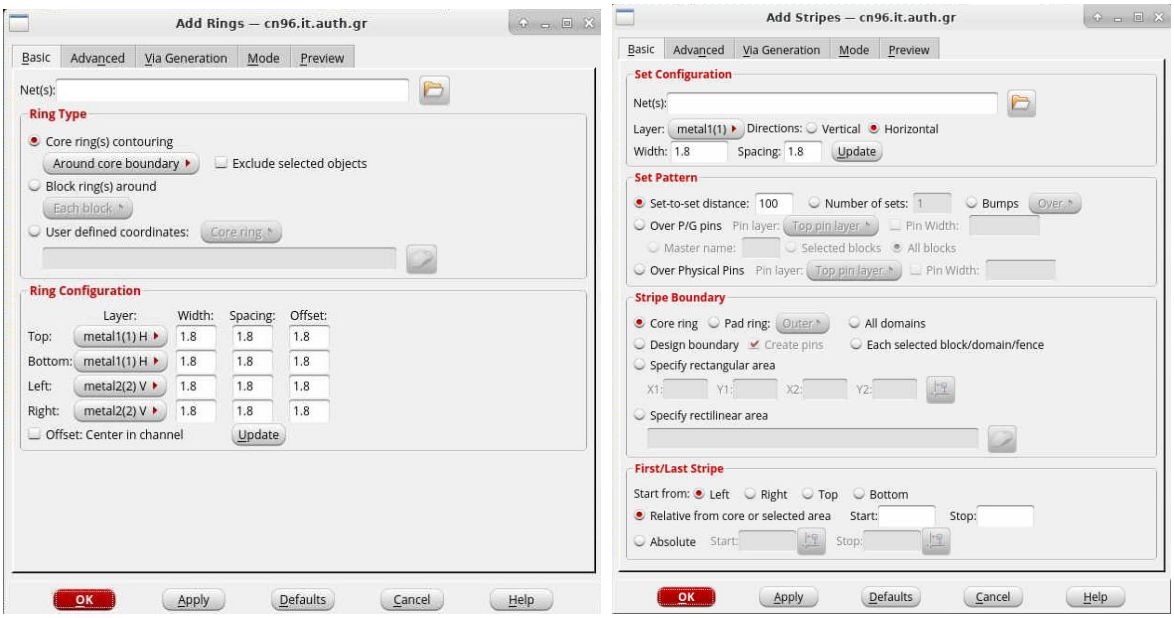
Το επόμενο βήμα αποτελεί η δημιουργία του δικτύου διανομής ισχύος (Power Distribution Network) μέσω των δακτυλίων και των γραμμών. Μπορούμε να ξεκινήσουμε τη διαδικασία μέσω της επιλογής **Power → Power Planning → Add Ring** για την προσθήκη δακτυλίων. Αυτή η επιλογή θα ανοίξει το παράθυρο του Σχήματος 10. Σε πρώτο στάδιο, πρέπει να γίνει επιλογή της γείωσης και της τροφοδοσίας (**Προσοχή:** τα ονόματα πρέπει να είναι έτσι όπως ονομάστηκαν κατά τη διεργασία της εισαγωγής του design) μέσω του εικονιδίου με το φάκελο στο πεδίο Net(s).

Στην περιοχή Ring Type, η πρώτη επιλογή αφορά τη δημιουργία δακτυλίων μόνο γύρω από την περιφέρεια του πυρήνα (core boundary) ή των I/O (I/O boundary) αναλόγως, η δεύτερη επιλογή δημιουργεί δακτυλίους και γύρω από τα υπάρχοντα μπλοκ στο design μας, ενώ η τρίτη δημιουργεί σε μια περιοχή που επιλέγουμε εμείς εισάγοντας τις συντεταγμένες της.

Όσον αφορά την περιοχή Ring Configuration, εκεί πρέπει να επιλεχθεί το μέταλλο που θέλουμε να χρησιμοποιήσουμε για κάθε πλευρά του δακτυλίου, καθώς και το πάχος, το κενό ανάμεσα τους και το offset. Οι επιλογές για αυτά τα τρία είναι σημαντικό να γίνουν λαμβάνοντας υπόψη την επιλογή που έγινε για την περιφέρεια κατά τη χωροθέτηση (Core to IO boundary). Επίσης, μπορεί να επιλεχθεί το Center in channel αν είναι επιθυμητό να βρίσκονται οι δακτύλιοι στο κέντρο του καναλιού ανάμεσα στον πυρήνα και τα I/O και δεν θέλουμε να προσθέσουμε offset.

Το επόμενο βήμα του δικτύου ισχύος αφορά τη δημιουργία των γραμμών. Αυτή μπορεί να γίνει μέσω της επιλογής **Power → Power Planning → Add Stripe**. Το παράθυρο που θα εμφανιστεί απεικονίζεται στο Σχήμα 12. Με παρόμοιο τρόπο πρέπει να επιλεχθούν τα ονόματα της γείωσης και

της τροφοδοσίας από το εικονίδιο του φακέλου στο πεδίο Net(s). Ταυτόχρονα μπορεί να επιλεγεί το μέταλλο που θα χρησιμοποιηθεί για τις γραμμές, καθώς και αν θέλουμε αυτές να είναι οριζόντιες ή κάθετες, το πάχος τους (Width) και την απόσταση ανάμεσα στην γραμμή γείωσης και της τροφοδοσίας (Spacing).



Σχήμα 12. Το παράθυρο για την προσθήκη δακτυλίων (αριστερά) και γραμμών (δεξιά) στο φυσικό σχέδιο.

Πριν συνεχίσουμε τη διαδικασία της δρομολόγησης του δικτύου ισχύος με τη δημιουργία των followpins, πρέπει να γίνει συσχέτιση των δικτύων τροφοδοσίας και της γείωσης με τους αντίστοιχους ακροδέκτες των κελιών της βιβλιοθήκης που χρησιμοποιούμε. Αυτό μπορεί να συμβεί μέσω των εντολών

```
globalNetConnect <name_of_PG_net> -type pgpin -pin <name_of_PG_pin_in_LEF> -inst*
```

```
globalNetConnect <name_of_PG_net> -type <tiehi|tielo> -instanceBasename *
```

τις οποίες πρέπει να χρησιμοποιήσουμε δύο φορές (μια για την τροφοδοσία και μια για την γείωση). Η επιλογή του τύπου στη δεύτερη εντολή είναι `tiehi` για την περίπτωση της τροφοδοσίας και `tielo` για την περίπτωση της γείωσης.

Επίσης, είναι σημαντική η δημιουργία των ακροδεκτών τροφοδοσίας και γείωσης στο φυσικό επίπεδο, καθώς δεν υπάρχουν από πριν στο netlist. Για αυτό τον λόγο χρησιμοποιούμε την εντολή `createPGPin`, η οποία ορίζεται ως εξής:

```
createPGPin <name_of_pin> -net <name_of_PG_net> -geom <MetalX> <Coordinates>
```

> Οι συντεταγμένες αφορούν τα κάτω-αριστερά και πάνω-δεξιά σημεία της σύνδεσης σε μορφή (x,y). Για παράδειγμα, η επιλογή *–geom Metal5 0 2 5 7*, θα δημιουργήσει μια ορθογώνια σύνδεση στο μέταλλο 5 με κάτω-αριστερά σημείο το (0,2) και πάνω-δεξιά το (5,7) στο διδιάστατο επίπεδο που σχεδιάζουμε φυσικά το κύκλωμα. Για το κύκλωμα ως αρχή των δύο αξόνων (0,0) θεωρείται το κάτω δεξιά σημείο του πυρήνα.

όπου ορίζουμε το όνομα του ακροδέκτη (π.χ. vdd), το όνομα του net που θέλουμε να γίνει σύνδεση και το όνομα του μετάλλου που θέλουμε να χρησιμοποιήσουμε (π.χ. Metal5) μαζί με τις τέσσερις απαραίτητες συντεταγμένες στο διδιάστατο χώρο (Rectangle Co-ordinates). Η εντολή αυτή θα δημιουργήσει μια ορθογώνια σύνδεση πάνω στο κύκλωμά μας. Η σύνδεση των ακροδεκτών αυτών πρέπει να γίνει με τους δακτύλιους τροφοδοσίας και γείωσης που δημιουργήσαμε προηγουμένως.

Προσοχή: Δώστε μεγάλη βαρύτητα στα μέταλλα που θα χρησιμοποιήσετε για τους ακροδέκτες τροφοδοσίας και γείωσης.

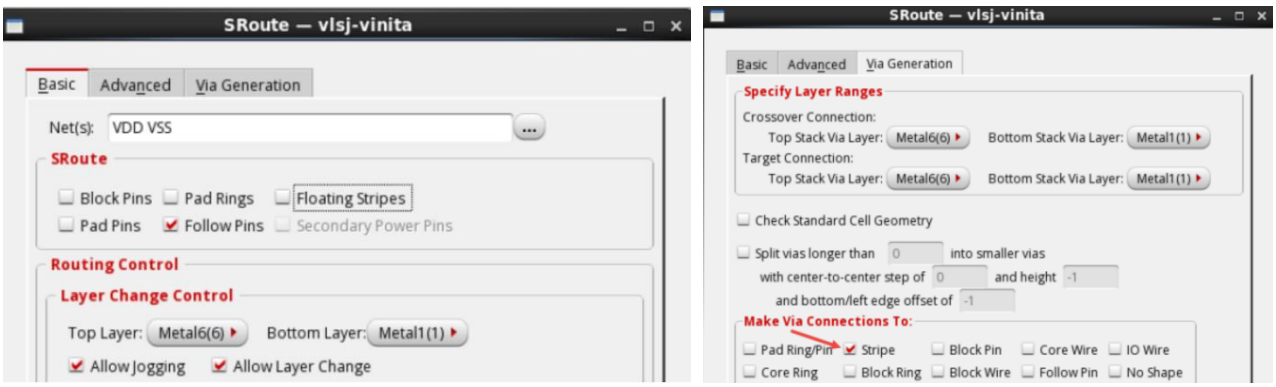
Για να μετρήσουμε τις αποστάσεις πάνω στο σχέδιο, μπορούμε να επιλέξουμε την επιλογή **Create Ruler** στην κάτω γραμμή εργαλείων ή να πατήσουμε το «**k**» ως shortcut. Αυτό θα δημιουργήσει ένα χάρακα στο σημείο που έχουμε το ποντίκι. Αν θέλουμε να ακυρώσουμε τη δημιουργία του χάρακα πατάμε «Esc», αλλιώς κάνουμε κλικ στο τελικό σημείο που θέλουμε για να μετρήσουμε την απόσταση. Μόλις δημιουργήσουμε το χάρακα, για να τον διαγράψουμε, πρέπει να μεταβούμε στην επιλογή **Clear All Rulers** στην πάνω γραμμή εργαλείων ή να πατήσουμε το «**Shift+k**».

Για να δημιουργήσουμε via (κατακόρυφη σύνδεση μεταξύ δύο μετάλλων) χρησιμοποιούμε την εντολή `editPowerVia`, που ορίζεται ως εξής:

```
editPowerVia -top_layer <MetalX> -area {<Coordinates>} -add_vias 1
-bottom_layer <MetalY>
```

όπου ορίζουμε το ανώτερο και το κατώτερο επίπεδο μετάλλου που θέλουμε να συνδέσουμε με το συγκεκριμένο via, καθώς και τις συντεταγμένες (Rectangle Coordinates) που δείχνουν την περιοχή που θέλουμε να το δημιουργήσουμε, με παρόμοια λογική.

Για το τελευταίο βήμα δημιουργίας του δικτύου ισχύος, πρέπει να επιλέξουμε **Route → Special Route**. Αυτή η επιλογή θα ανοίξει το παράθυρο του Σχήμα 13.



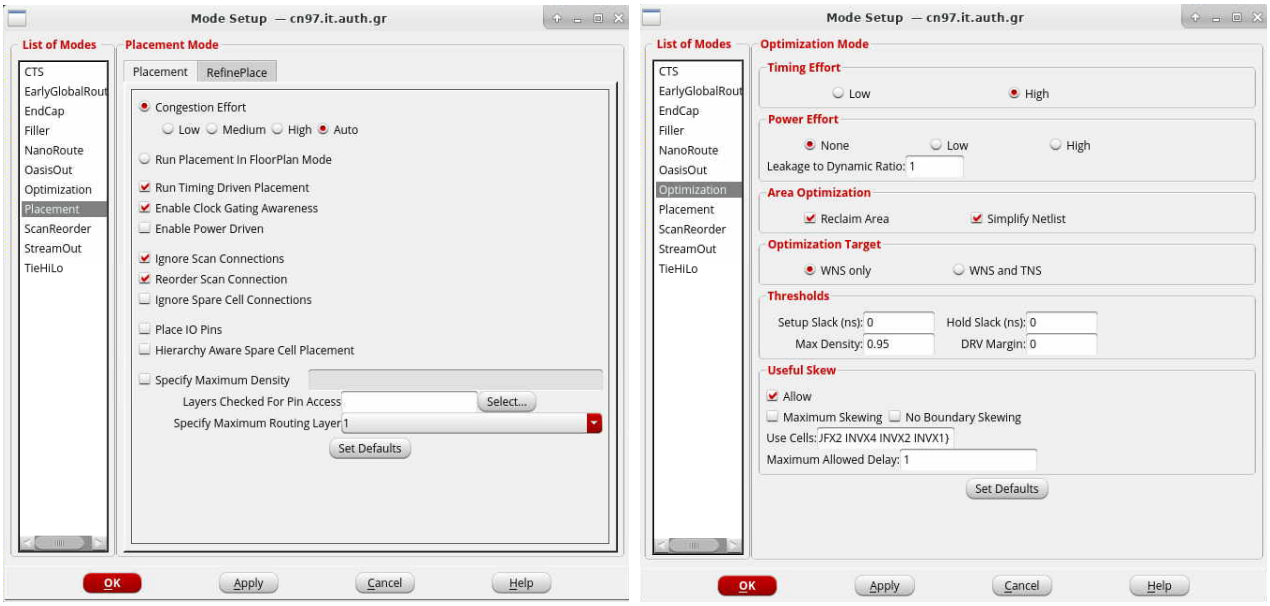
Σχήμα 13. Το παράθυρο για το Special Routing.

Όπως και στις προηγούμενες ρυθμίσεις, πρέπει να γίνει επιλογή των απαραίτητων Net(s) τροφοδοσίας και γείωσης. Οι επιλογές στο πεδίο SRoute δείχνουν τους διάφορους τύπους δρομολόγησης της ισχύος που μπορούν να επιτευχθούν από το εργαλείο, με σημαντικότερη την επιλογή **Follow Pins**, καθώς αφορά τη σύνδεση του δικτύου ισχύος με τα κελιά της βιβλιοθήκης που είναι στο φυσικό σχέδιο. Όσον αφορά το πεδίο Layer Change Control, οι ρυθμίσεις δείχνουν τον τρόπο που συμβαίνει αλλαγή του μετάλλου για το δίκτυο ισχύος, όπου τα Top Layer και Bottom Layer αποτελούν το ανώτερο και το κατώτερο μέταλλο αντίστοιχα, τα οποία το εργαλείο μπορεί να χρησιμοποιήσει για να δρομολογήσει το δίκτυο ισχύος. Οι επιλογές **Allow Jogging** και **Allow Layer Change** οδηγούν σε διαφορετικά αποτελέσματα ανάλογα με το ποια/ες επιλογές είναι ενεργοποιημένες κάθε φορά ως εξής:

- Αν και οι δύο επιλεχθούν, τότε το εργαλείο εκτελεί αλλαγές ανάμεσα σε επίπεδα μετάλλου και μπορεί να χρησιμοποιήσει μέταλλα οριζόντια και κατακόρυφα (jogging) για να αποφύγει DRC παραβάσεις (συνίσταται)
- Αν δεν επιλεχθεί το **Allow Layer Change,** αλλά επιλεχθεί το Jogging, τότε το εργαλείο χρησιμοποιεί μόνο ένα συγκεκριμένο μέταλλο οριζόντια και κατακόρυφα όποτε είναι δυνατό. Αυτή η επιλογή μπορεί να οδηγήσει σε δρομολόγηση με μη-επιθυμητή κατεύθυνση.
- Αν δεν επιλεχθεί το **Allow Jogging**, αλλά επιλεχθεί το Layer Change, τότε το εργαλείο εκτελεί τη δρομολόγηση μόνο σε μια κατεύθυνση, αλλά μπορεί να γίνει αλλαγή μετάλλου για να αποφευχθούν DRC παραβάσεις.
- Αν δεν επιλεχθεί καμία από τις δύο, τότε είναι εφικτή η δρομολόγηση μόνο σε ένα επίπεδο μετάλλου και σε ευθείες γραμμές.

Στην καρτέλα Via Generation, μπορεί να γίνει έλεγχος του τρόπου με τον οποίο θέλουμε το εργαλείο να εισάγει τα vias κατά τη δρομολόγηση του δικτύου ισχύος. Στο Σχήμα 13 φαίνονται ενδεικτικά οι επιλογές που υπάρχουν, όπως ρύθμιση του εύρους των μετάλλων που θέλουμε να υπάρχει δημιουργία vias, καθώς και ρύθμιση σε ποια τμήματα του δικτύου θέλουμε να τα δημιουργήσουμε (π.χ. αν θέλουμε στις γραμμές, επιλέγουμε Stripe).

Τέλος, έχουμε τη δυνατότητα να κάνουμε ρυθμίσεις για την τοποθέτηση μέσω της επιλογής **Tools → Set Mode → Mode Setup**. Από το παράθυρο που θα εμφανιστεί, επιλέγουμε το Placement, όπως φαίνεται στο Σχήμα 14.



Σχήμα 14. Παράθυρο για τις ρυθμίσεις της τοποθέτησης, καθώς και άλλων modes.

Για την τοποθέτηση μπορούμε να ρυθμίσουμε το **effort** που δείχνει πόσο προσπάθεια θα καταβάλλει το εργαλείο για να αντιμετωπίσει προβλήματα συμφόρησης. Σε αντίθετη περίπτωση μπορούμε να τρέξουμε το **Floorplan Mode** για να εκτελέσει το εργαλείο μια γρήγορη τοποθέτηση, προκειμένου να γίνει αναγνώριση της εφαρμοσιμότητας του netlist (τα κελιά μπορεί να μην τοποθετηθούν σε νόμιμες θέσεις). Επίσης, το **timing driven placement** εκτελεί την τοποθέτηση λαμβάνοντας υπόψη κρίσιμα μονοπάτια στο design, ενώ το **clock gating awareness** αναγνωρίζει clock gating πύλες, εφόσον υπάρχουν (πρέπει να γίνει φόρτωση του specification file για το δέντρο ρολογιού, προτού τρέξουμε την τοποθέτηση με αυτή την επιλογή). Η επιλογή **power driven** αναγνωρίζει και περιορίζει κρίσιμα μονοπάτια για να μειώσει την ισχύ μετάβασης. Τέλος, το **Place**

IO Pins κάνει το εργαλείο να προσθέσει τα I/O ανάλογα με την τοποθέτηση, προσπαθώντας ταυτόχρονα να βρει τις βέλτιστες θέσεις για αυτά.

Για τη βελτιστοποίηση (Optimization), το παράθυρο φαίνεται στο ίδιο σχήμα. Μπορούμε να επιλέξουμε το **effort** της βελτιστοποίησης χρονισμού, αν θέλουμε να γίνει βελτιστοποίηση ως προς την ισχύ, ενώ όσον αφορά το μέγεθος του design, το **Reclaim Area** δημιουργεί επιπλέον χώρο μειώνοντας το μέγεθος των κελιών (downsize) και διαγράφοντας buffers διατηρώντας ταυτόχρονα τα worst negative slack (WNS) και total negative slack (TNS) σταθερά, ενώ το **Simplify Netlist** απλοποιεί το design με διάφορους τρόπους. Επίσης, μπορούμε να δώσουμε στο εργαλείο την επιλογή βελτίωσης είτε μόνο του WNS, είτε και των δύο (WNS & TNS), να προστεθούν threshold για διάφορα μεγέθη ή να προσθέσουμε μια λίστα κελιών που μπορούν να χρησιμοποιηθούν για τη δημιουργία χρήσιμης στρέβλωσης (useful skew).

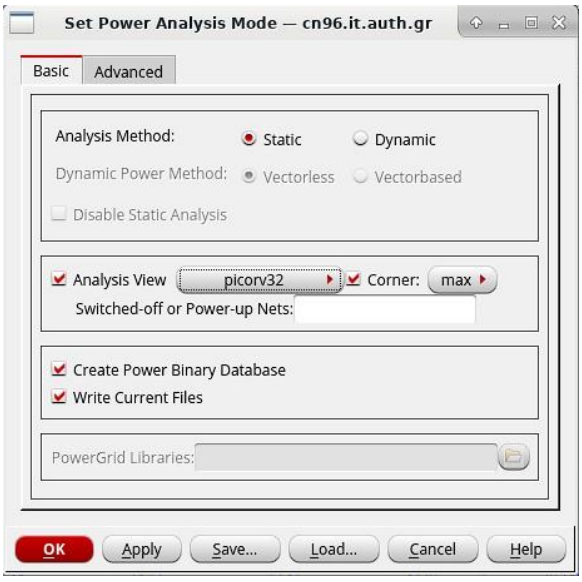
> Ο λόγος **Leakage to Dynamic Ratio** δείχνει σε ποιο τύπο ισχύος θέλουμε να δώσουμε μεγαλύτερη βαρύτητα για βελτιστοποίηση και παίρνει τιμές από 0.0 έως 1.0. Η τιμή 1.0 δείχνει ότι θέλουμε βελτιστοποίηση ως προς την ισχύ διαρροής, ενώ η τιμή 0.0 δείχνει ότι θέλουμε να δώσουμε βαρύτητα μόνο στη δυναμική ισχύ.

5.3.2. Power Rail Analysis

Μετά τη δρομολόγηση του δικτύου ισχύος δίνεται η δυνατότητα ανάλυσής του. Η παροχή της τάσης VDD και της γείωσης VSS είναι πολύ σημαντικός παράγοντας λειτουργίας του κυκλώματός μας. Για το λόγο αυτό είναι πολύ σημαντικό να είμαστε βέβαιοι ότι η σωστή τάση τροφοδοσίας καταλήγει στα κελιά μας, δίχως να πέφτει κάτω από ένα επιτρεπτό κατώφλι. Το Innovus παρέχει εργαλεία τα οποία οπτικοποιούν αυτήν την πτώση τάσης και επιβεβαιώνουν ότι η τάση που μεταδίδεται σε κάθε σημείο του κυκλώματος είναι ικανοποιητική. Η διαδικασία αυτή ονομάζεται Early Power Rail Analysis και μπορεί να γίνει ακόμα και πριν από την τοποθέτηση των κελιών.

Το πρώτο βήμα είναι να γίνει Power Analysis.

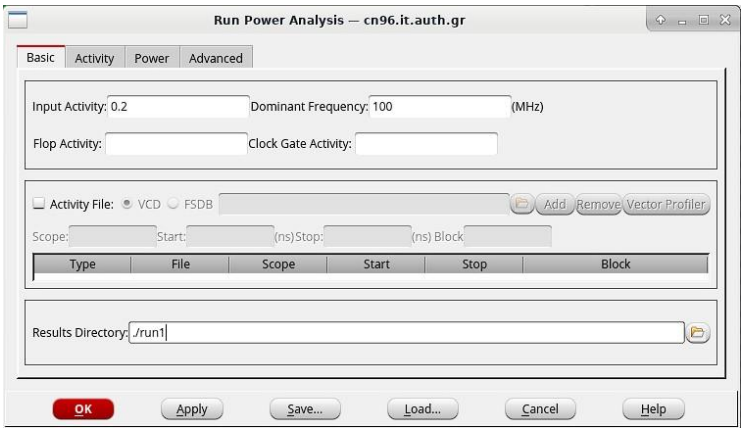
1. Διαλέγουμε Power - Power Analysis - Setup.



Σχήμα 15. Run Power Analysis.

2. Στο πεδίο Analysis Method διαλέγουμε Static.
3. Τσεκάρουμε το Analysis View και επιλέγουμε το view που έχουμε ορίσει στο import.

4. Διαλέγουμε την επιλογή max για το Corner.
5. Πατάμε OK.
6. Για να τρέξουμε Power Analysis, διαλέγουμε Power - Power Analysis - Run.

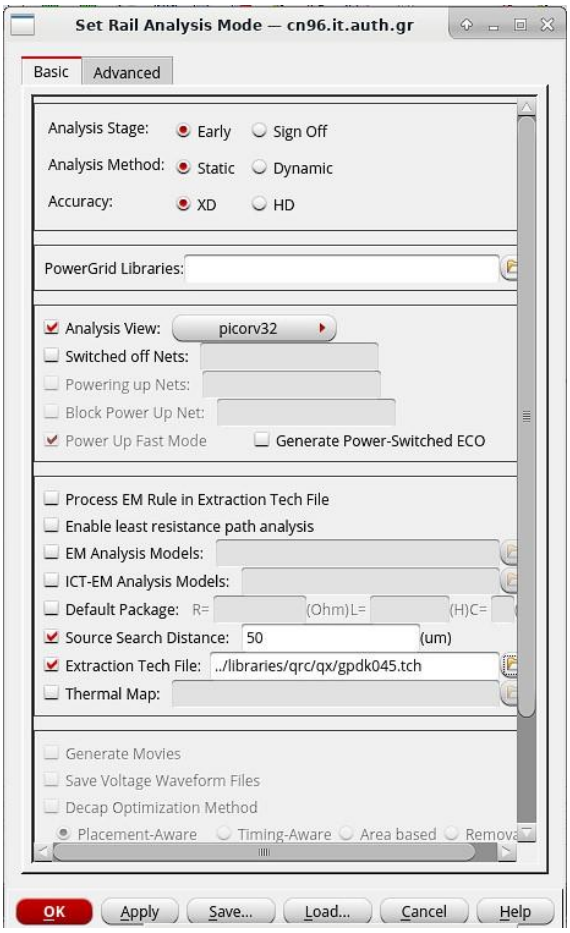


Σχήμα 16. Setup Rail Analysis.

7. Αφήνουμε όλες τις προκαθορισμένες ρυθμίσεις και εισάγουμε το directory ./run1 στο πεδίο Results Directory.
8. Πατάμε OK.

Στη συνέχεια κάνουμε Rail Analysis βασιζόμενοι στα αποτελέσματα του Power Analysis.

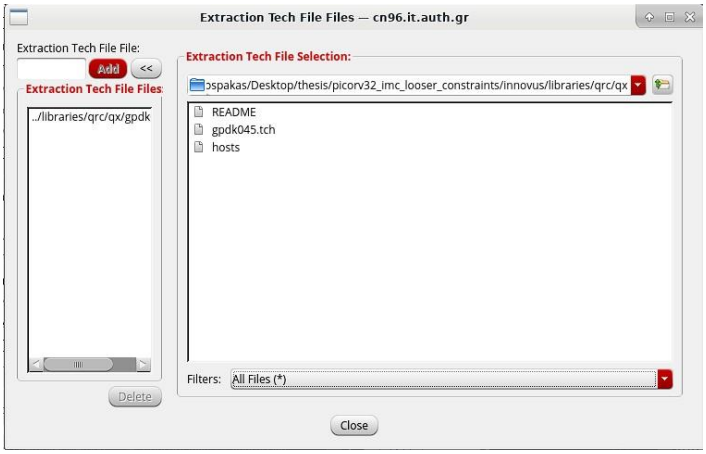
1. Επιλέγουμε Power - Rail Analysis – Setup



Σχήμα 17. Setup Rail Analysis.

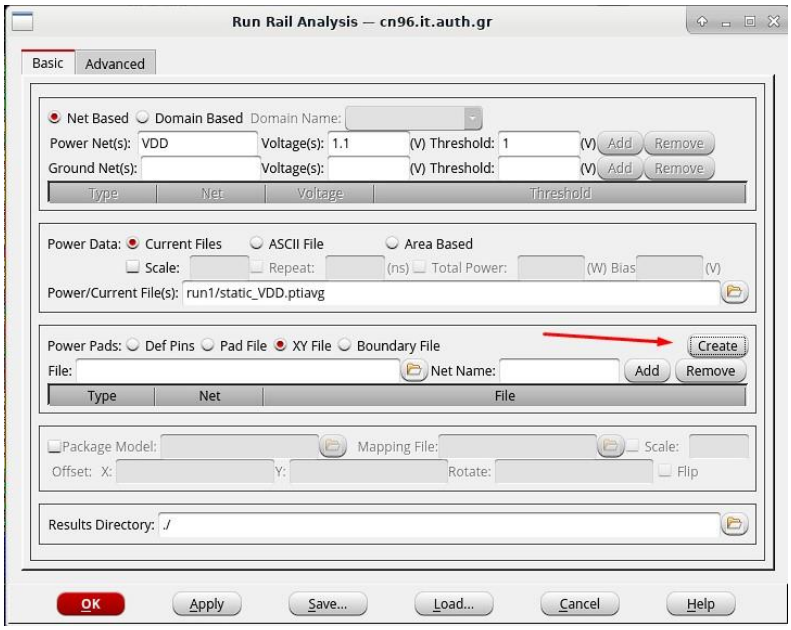
2. Τσεκάρουμε την επιλογή Early στο πεδίο Analysis Stage.

3. Τσεκάρουμε το Analysis View και επιλέγουμε το view που έχουμε ορίσει το import.
4. Διαλέγουμε το Extraction Tech File, όπως φαίνεται στο Σχήμα 18.



Σχήμα 18. Set Extraction Tech File.

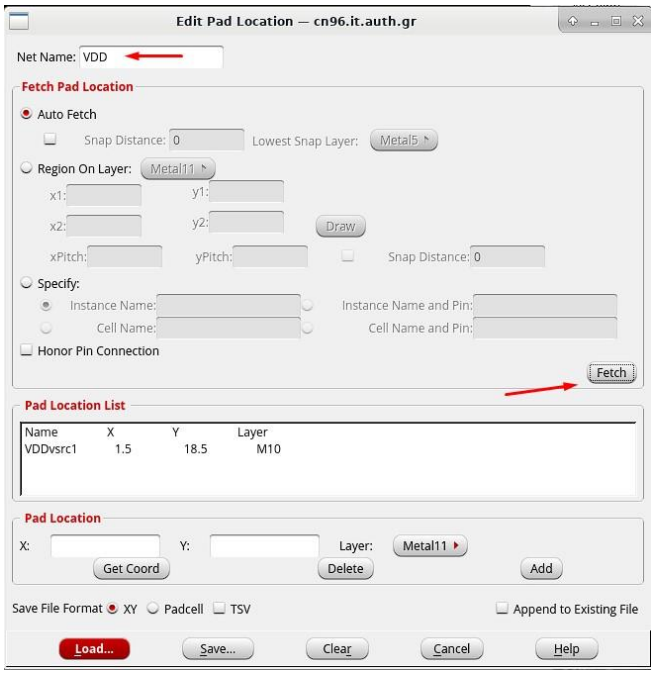
5. Συμπληρώνουμε όλα τα υπόλοιπα πεδία σύμφωνα με τη φωτογραφία στο βήμα 1.
6. Πατάμε OK.
7. Επιλέγουμε Power - Rail Analysis - Run.



Σχήμα 19. Run Rail Analysis.

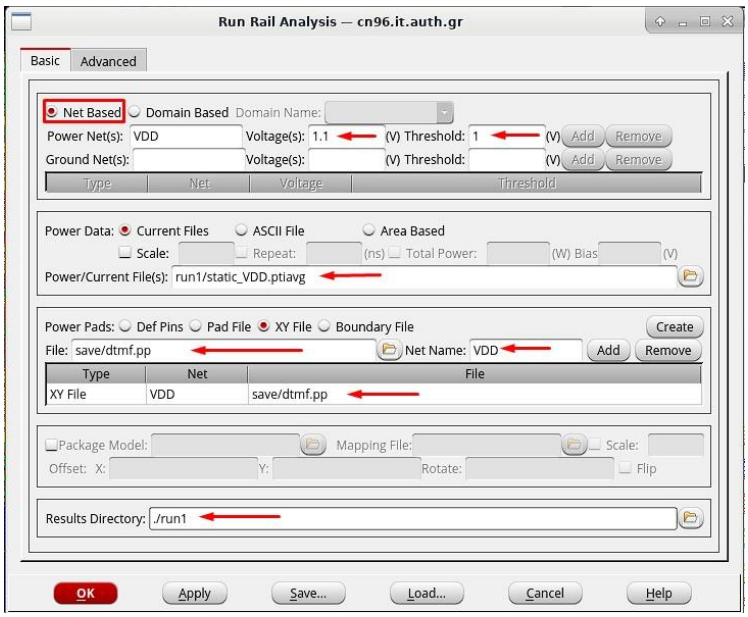
8. Επιλέγουμε Net Based και θέτουμε VDD (ή το όνομα που έχουμε δώσει στην VDD μας κατά τη δημιουργία του project).
9. Βάζουμε στο πεδίο Voltage(s) τη μέγιστη θεωρητική τάση που δίνουμε ως είσοδο στο κύκλωμα μας. Στη συγκεκριμένη βιβλιοθήκη η τάση VDD είναι ίση με 1.1V.
10. Βάζουμε στο πεδίο Threshold την τάση στην οποία θέλουμε το Innovus να μας εμφανίσει σήμανση αν υπάρχει πτώση. Χρησιμεύει κατά την ανάλυση, ώστε τα οπτικά αποτελέσματα να έχουν νόημα.
11. Αν υπάρχει VSS στην επιλογή Ground Nets τη σβήνουμε.
12. Επιβεβαιώνουμε ότι η επιλογή Current Files είναι επιλεγμένη.
13. Για το πεδίο Power/Current Files πηγαίνουμε στο directory run1 (που δημιουργήσαμε στο Power Analysis) και επιλέγουμε το αρχείο static VDD.ptiavg

14. Για το Results Directory βάζουμε ./run1
15. Στα Power Pads έχουμε επιλεγμένο το XY File. Θα χρειαστεί να παράγουμε αυτό το αρχείο.
16. Πατάμε Create και εμφανίζεται το παρακάτω παράθυρο.



Σχήμα 20. Edit Pad Location.

17. Βάζουμε στο Net Name το όνομα VDD.
18. Διαλέγουμε την επιλογή Auto Fetch. Πατάμε fetch και βλέπουμε ότι εμφανίζονται στοιχεία στο Pad Location List.
19. Επιλέγουμε Save.
20. Δίνουμε το όνομα vdd.pp στο αρχείο και το αποθηκεύουμε. (ή όποιο άλλο όνομα θέλουμε)
21. Πατάμε Cancel στο Edit Pad Location Form (το παράθυρο που ανοίξαμε τελευταίο) και επιστρέφουμε στο προηγούμενο.
22. Εκεί βάζουμε ως αρχείο των Power Pads το vdd.pp που δημιουργήσαμε προηγουμένως, για Net Name βάζουμε VDD και πατάμε Add.
23. Επιβεβαιώνουμε ότι το παράθυρο μας μοιάζει με το παρακάτω.

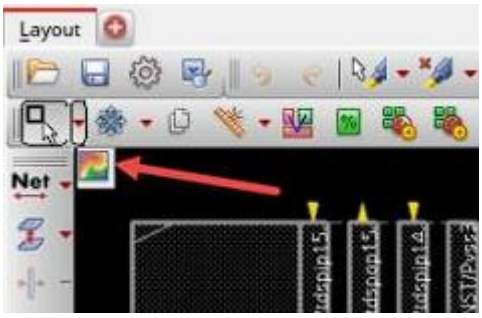


Σχήμα 21. Run Rail Analysis.

24. Πατάμε OK για να τρέξουμε Rail Analysis.

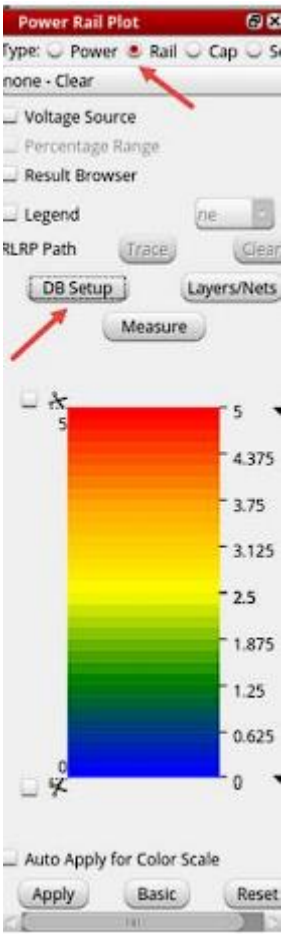
Τέλος οπτικοποιούμε τα αποτελέσματα του Rail Analysis.

1. Για την οπτικοποίηση επιλέγουμε Power - Report - Power Rail Result.
2. Κάνουμε δύο φορές κλικ στο παράθυρο όπως φαίνεται στο παρακάτω βελάκι.



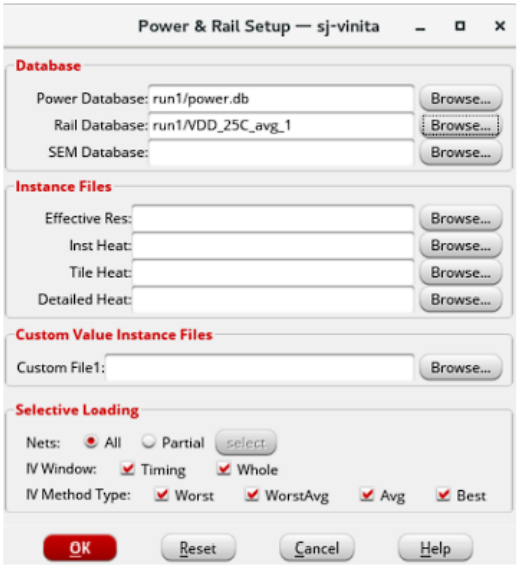
Σχήμα 22. Visualizing Step 1.

3. Στο παράθυρο που ανοίγει διαλέγουμε Rail.



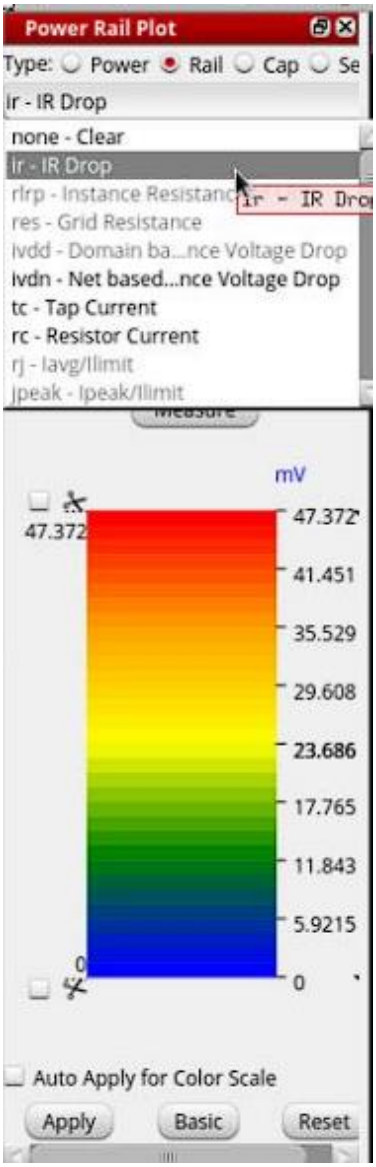
Σχήμα 23. Power Rail Plot.

4. Επιλέγουμε DB Setup και το συμπληρώνουμε ακολούθως.



Σχήμα 24. DB Setup.

5. Πατάμε OK.
6. Διαλέγουμε ir - IR Drop από την λίστα που εμφανίζεται.



Σχήμα 25. Choose IR Drop.

7. Παρατηρούμε το οπτικοποιημένο αποτέλεσμα.

5.3.3. Ροή πριν τη σύνθεση του δέντρου ρολογιού (PreCTS)

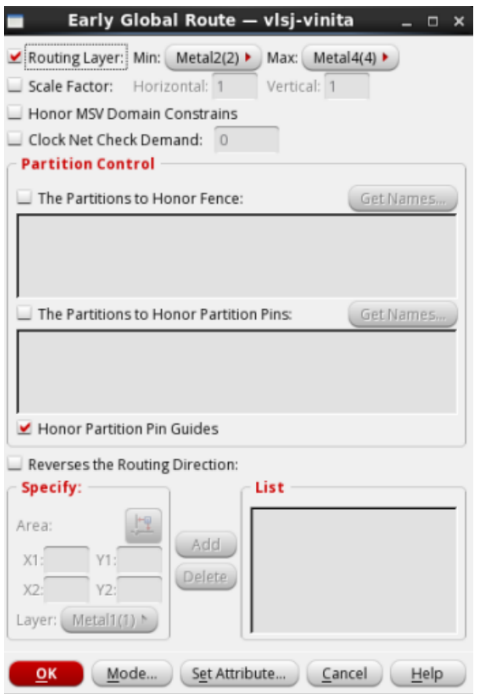
Μετά τη δημιουργία του δικτύου ισχύος, μπορούμε να συνεχίσουμε με το επόμενο βήμα της φυσικής σχεδίασης που είναι η τοποθέτηση. Για να ελέγξουμε τις ρυθμίσεις της τοποθέτησης μπορούμε να χρησιμοποιήσουμε την εντολή `getPlaceMode`. Μετά την εκτέλεση της εντολής, μπορούμε να παρατηρήσουμε ότι το `-place_global_timing_effort` είναι ρυθμισμένο στην επιλογή **medium** από προεπιλογή, ενώ το `-place_global_cong_effort` είναι επιλεγμένο στο **auto**. Αν εμφανιστούν προκλήσεις σχετικά με το χρονισμό ή τη συμφόρηση, τότε οι επιλογές αυτές μπορούν να προσαρμοστούν αναλόγως.

Για να τρέξουμε την τοποθέτηση ταυτόχρονα με τη βελτιστοποίηση του συγκεκριμένου βήματος, χρησιμοποιούμε την εντολή `place_opt_design`, η οποία χρειάζεται λίγα λεπτά για να ολοκληρωθεί. Μόλις τερματίσει, μπορούμε να δούμε το αποτέλεσμα μέσω της επιλογής **Physical view**. Για να ελέγξουμε αν η τοποθέτηση των κελιών έχει γίνει σωστά μπορούμε να μεταβούμε στην επιλογή **Place → Check Placement**, και να κάνουμε κλικ στο **OK.** Αν υπάρχει κάποια

παράβαση στο κύκλωμα, τότε μπορούμε να μεταβούμε στην επιλογή **Place → Refine Placement**, και να επιλέξουμε να γίνει διατήρηση δρομολόγησης έως τώρα (Preserve Routing) και το **OK**.

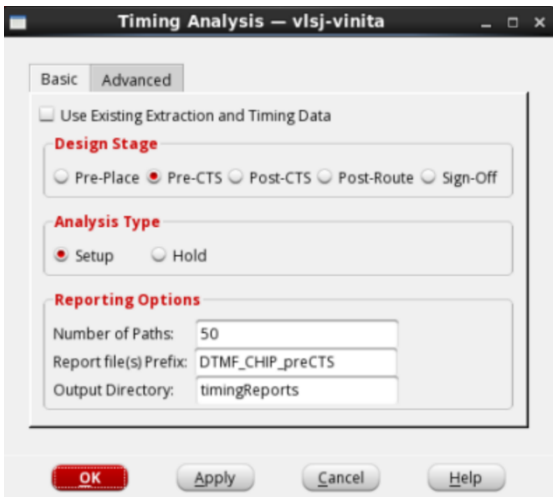
Μετά το πέρας της τοποθέτησης, υπάρχει δυνατότητα ελέγχου αν κατά τη δρομολόγηση θα εμφανιστεί συμφόρηση σε κάποια από τα σημεία του φυσικού σχεδίου. Η διαδικασία αυτή ονομάζεται Early Global Route και μπορεί να ενεργοποιηθεί μέσω της επιλογής **Route → Early Global Route** από τη γραμμή εργασίας. Η επιλογή αυτή θα εμφανίσει το παράθυρο του Σχήμα 26. Σε αυτό μπορούν να ρυθμιστούν διάφορες πληροφορίες, όπως τα στρώματα δρομολόγησης (Routing Layers) μέσω ενός εύρους ή η κλιμάκωση των οριζόντιων και κατακόρυφων tracks κατά έναν αριθμό (Scale Factor). Άλλες επιλογές αποτελούν το Honor MSV Domain Constraints, η οποία αναγκάζει τη διεργασία (Early Global Route) να ακολουθήσει τις ρυθμίσεις του power domain και να κάνει δρομολόγηση όσο το δυνατόν περισσότερο σε ένα συγκεκριμένο power domain μόνο, ενώ το Clock Net Check Demand χρησιμοποιείται για να δώσουμε ένα συγκεκριμένο πάχος στις συνδέσεις του ρολογιού. Στα πλαίσια της εργασίας αυτής θα γίνει ανάλυση και χρήση περισσότερο της πρώτης ρύθμισης.

Η εξαγωγή μιας αναφοράς για τα παρασιτικά (βασισμένα στο Early Global Route) μπορεί να γίνει αν επιλεχθεί **Timing → Extract RC** από τη γραμμή εργασίας. Το αρχείο που θα εξαχθεί θα έχει τη μορφή SPEF και είναι απαραίτητο για να γίνει ανάλυση χρονισμού. Ομοίως μπορεί να γίνει εξαγωγή του αρχείου καθυστερήσεων SDF αν επιλεχθεί **Timing → Write SDF** (επειδή δεν έχει γίνει ακόμα σύνθεση του δέντρου ρολογιού, μετά την τοποθέτηση, είναι σημαντικό να επιλέξουμε το **Ideal Clock**).



Σχήμα 26. Το παράθυρο για την έναρξη της διεργασίας Early Global Route.

Προκειμένου να τρέξουμε μια ανάλυση χρονισμού σε κάποιο στάδιο της φυσικής σχεδίασης, επιλέγουμε **Timing → Report Timing** κάτι που θα ανοίξει το παράθυρο του Σχήμα 27.



Σχήμα 27. Το παράθυρο για την επιλογή των ρυθμίσεων της ανάλυσης χρονισμού.

Επειδή στην περίπτωση μας δεν έχει γίνει ακόμα σύνθεση του δέντρου ρολογιού, επιλέγουμε το στάδιο **PreCTS**, σε αντίθετη περίπτωση επιλέγουμε το στάδιο που βρισκόμαστε κάθε φορά εκείνη τη στιγμή. Αφήνουμε την επιλογή **Setup** στον τύπο ανάλυσης για να δημιουργήσουμε αναφορές για το χρόνο προετοιμασίας ή **Hold** για το χρόνο διατήρησης. Επίσης, αναγράφονται και άλλες πληροφορίες, όπως το όνομα της αναφοράς, ο αριθμός των μονοπατιών για τα οποία θα εξαχθεί η αναφορά, αλλά και το Directory εξόδου. Επίσης, μπορούμε να επιλέξουμε το **Timing → Debug Timing** για να εμφανιστεί το παράθυρο Απεικόνισης αναφοράς χρονισμού (Display/Generate Timing Report), οπού μπορεί να φανεί αν υπάρχουν μονοπάτια με αρνητικό slack.

> Η επιλογή **Pre-Place** θεωρεί μηδενικά μοντέλα για το wire-load αγνοώντας ταυτόχρονα τις συνδέσεις με υψηλό Fanout. Αυτή η επιλογή είναι κατάλληλη για να γίνει έλεγχος αν υπάρχουν καθόλου σφάλματα στο αρχείο περιορισμών πριν γίνει εκτέλεση της τοποθέτησης για πρώτη φορά.

Για εξαγωγή πληροφοριών σχετικά με την ισχύ που καταναλώνεται στο κύκλωμα και την επιφάνεια, μπορούμε να χρησιμοποιήσουμε την εντολή `report_power` και την εντολή `report_area` αντίστοιχα, σε οποιοδήποτε στάδιο της σχεδίασης βρισκόμαστε. Επίσης, μπορεί να γίνει χρήση της εντολής `reportNetStat` για να εκτυπωθεί ο αριθμός κάθε είδους πύλης που χρησιμοποιείται και της εντολής `report_route -summary` για να τυπωθούν πληροφορίες σχετικά με το μήκος δρομολόγησης των συνδέσεων, καθώς και τον αριθμό των via.

5.3.4. Ροή μετά τη σύνθεση του δέντρου ρολογιού (PostCTS)

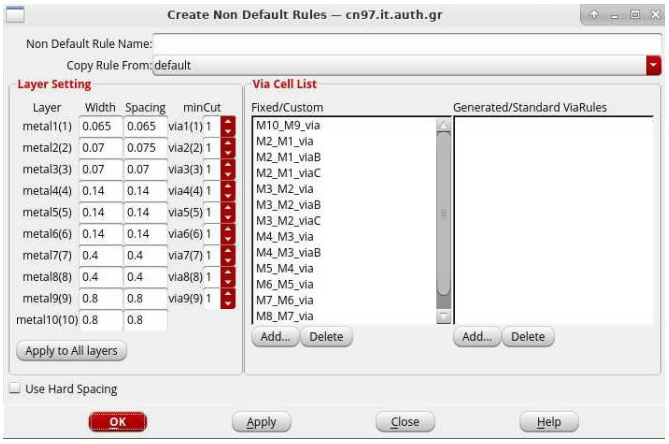
Αφού εκτελέσουμε την τοποθέτηση ή τη βελτιστοποίηση μετά το στάδιο του PreCTS, πρέπει να γίνει σύνθεση του δέντρου ρολογιού με περιορισμούς στο ποιοι buffers ή ποιες πύλες θα χρησιμοποιηθούν και τον τύπο της δρομολόγησης που θα υλοποιήσουμε.

> Υπενθυμίζεται ότι η εξαγωγή των αναφορών σε ξεχωριστό αρχείο στο cd μπορεί να γίνει μέσω του χαρακτήρα «>». Για παράδειγμα, η εντολή *report_power > power.txt* θα τυπώσει τα αποτελέσματα ισχύος απευθείας σε .txt αρχείο στο cd.

Σε πρώτο στάδιο πρέπει να δημιουργηθεί ένας μη-προκαθορισμένος κανόνας δρομολόγησης για το δέντρο ρολογιού (Non-Default Rule). Ο κανόνας αυτός αφορά το πλάτος όλων των μετάλλων (Width), καθώς και το ελάχιστο κενό ανάμεσα σε παρόμοια μέταλλα (Spacing). Συνήθως,

επιλέγουμε διπλάσιο ή τριπλάσιο πάχος μετάλλων και τουλάχιστον διπλάσιο κενό, εφόσον οι τιμές δεν παραβιάζουν τις ελάχιστες και μέγιστες τιμές των DRC κανόνων που υπάρχουν στο LEF. Αυτοί οι κανόνες τυπικά χρησιμοποιούνται για τη δρομολόγηση των ρολογιών ή σε ευαίσθητα nets.

Για να ξεκινήσουμε τη διαδικασία δημιουργίας των Non Default Rule (NDR) πρέπει να επιλέξουμε **Edit → Create Non Default Rule**. Ανάλογα με το πάχος και το κενό που θέλουμε, εφαρμόζουμε τις αλλαγές για κάθε επίπεδο μετάλλου και εισάγουμε ένα όνομα για το NDR που μόλις φτιάξαμε. Ύστερα, πατάμε το **OK** για να τερματίσουμε τη διεργασία δημιουργίας των NDR.



Σχήμα 28. Παράθυρο για τη δημιουργία των NDR.

Αφού τελειώσουμε με την παραπάνω διαδικασία, μπορούμε να ξεκινήσουμε τη δημιουργία του νέου τύπου δρομολόγησης μέσω της εντολής `create_route_type`. Μέσω της εντολής αυτής μπορεί να γίνει ρύθμιση διάφορων χαρακτηριστικών της δρομολόγησης, όπως τα μέγιστα και ελάχιστα επίπεδα μετάλλου που είναι επιτρεπτά για χρήση μέσω των χαρακτηριστικών `-top_preferred_layer <number>` και `-bottom_preferred_layer <number>` μαζί με τον αριθμό του μετάλλου αντίστοιχα, να γίνει προσθήκη κάποιου NDR με το `-non_default_rule <NDR_name>` ακολουθούμενο με το όνομα του NDR, να ονομάσουμε τον τύπο δρομολόγησης μέσω του `-name` και το όνομα που θέλουμε, να κάνουμε shielding μέσω του `-shield_net <net_name>` ακολουθούμενο από το όνομα του net που θέλουμε να χρησιμοποιήσουμε ως shield (π.χ.VDD) ή να επιλέξουμε το effort της δρομολόγησης μέσω του `-preferred_routing_layer_effort <high|medium|low>` .

Έπειτα επιλέγουμε τις ρυθμίσεις του/των δέντρου/ων ρολογιού μέσω της εντολής `set_ccopt_property`. Αυτή χρησιμοποιείται για να ρυθμίσει κάποιες από τις επιλογές των χαρακτηριστικών του/των δέντρου/ων ρολογιού. Αρχικά πρέπει να σημειωθεί ότι για τα δέντρα ρολογιού αναγνωρίζονται δύο βασικά τμήματα (net types), το **trunk** και τα **leaves** που αποτελούν τα τμήματα του δέντρου που τροφοδοτούν τα **clock gates** και τα **sinks** αντίστοιχα. Με την παραπάνω εντολή μπορούν να οριστούν ποια nets θα χρησιμοποιούν ποιον τύπο δρομολόγησης, χρησιμοποιώντας τα χαρακτηριστικά `-route_type <name>` και `-net_type <top|trunk|leaf>`. Επίσης, μπορούμε να προσθέσουμε τους διαθέσιμους buffers ρολογιού μέσω του χαρακτηριστικού `buffer_cells <list_of_clock_buffers>` της παραπάνω εντολής ή να ορίσουμε τα λογικά κελιά που μπορούν να χρησιμοποιηθούν μέσω του `logic_cells <list_of_logic_cells>`. Επιπλέον, μπορούμε να θέσουμε την επιθυμητή στρέβλωση μέσω του `target_skew <number>` και το μέγιστο ρυθμό μετάβασης μέσω του `target_max_trans <number>`. Για να δούμε όλες τις πιθανές επιλογές της εντολής μπορούμε να χρησιμοποιήσουμε την `set_ccopt_property * -help`.

Αφού επιλέξουμε τα χαρακτηριστικά του/των δέντρου/ων ρολογιού μπορούμε να συνεχίσουμε με τη δημιουργία του spec αρχείου, το οποίο θα περιέχει πληροφορίες σχετικά με το δέντρο από όποιες επιλογές κάναμε πριν, αλλά και από αυτές που εισήγαμε στο αρχείο sdc στα βήματα της σύνθεσης. Η δημιουργία του αρχείου αυτού μπορεί να γίνει μέσω της εντολής:

```
create_ccopt_clock_tree_spec -file <name.spec>
```

> Αν θέλουμε να επιλέξουμε πολλαπλά κελιά που έχουν μικρή διαφορά στο όνομα, μπορούμε να γράψουμε τα πρώτα γράμματα και έπειτα να προσθέσουμε τον αστερίσκο. Για παράδειγμα, το «AND*» υποδεικνύει ότι θέλουμε να χρησιμοποιήσουμε όλα τα κελιά που το όνομά τους ξεκινάει από «AND».

Για να δημιουργήσουμε το/τα δέντρο/α ρολογιού, χρησιμοποιούμε την εντολή:

```
ccopt_design
```

Η παραπάνω εντολή εκτελεί ταυτόχρονη σύνθεση του/των δέντρου/ων ρολογιού, βελτιστοποιώντας τα μονοπάτια δεδομένων (datapaths) για να μην υπάρχει παραβίαση των χρονισμών. Μόλις τερματίσει η διεργασία, είναι εφικτή η οπτικοποίηση του/των δέντρου/ων μέσω της επιλογής **Clock → CCOpt Clock Tree Debugger** και κάνοντας κλικ στο **OK**. Αυτό θα ανοίξει ένα παράθυρο με το δέντρο, το οποίο σχεδιάστηκε κατά τη σύνθεση. Επιλέγοντας από το νέο παράθυρο το **View → Enable clock path browser** θα ανοίξει ένα νέο παράθυρο με τα διαθέσιμα analysis views και τα skew groups μαζί με τις τιμές στρέβλωσης και ελάχιστης/μέγιστης καθυστέρησης. Κάνοντας δεξί κλικ σε κάποιο από τα groups και επιλέγοντας **Highlight,** το μονοπάτι που θέλουμε (Max/Min Paths) και το χρώμα, αυτό θα χρωματίσει το μονοπάτι πάνω στο φυσικό σχέδιο στο Innovus, αλλά και στο παράθυρο που δείχνει την οργάνωση του δέντρου ρολογιού. Για να διαγράψουμε τα χρώματα, επιλέγουμε **View → Clear Highlight → Clear All**.

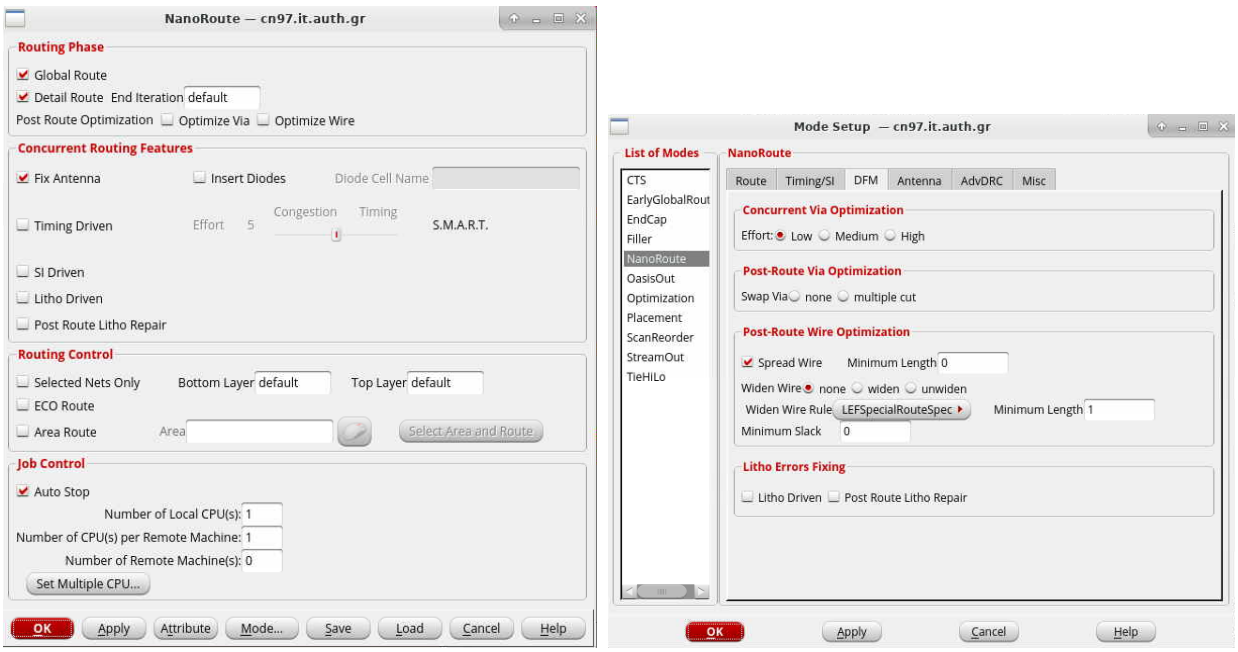
> Αν κάποιο από τα χρώματα δε φαίνεται στο φυσικό σχέδιο, μπορούμε να σκοτεινιάσουμε το view μέσω του *F12*. Πατώντας το *F12* τρίτη φορά θα επαναφέρει τη φωτεινότητα στην αρχική κατάσταση.

Αν υπάρχουν παραβάσεις σε κάποιο από τα slack, τότε μπορούμε να τρέξουμε τη βελτιστοποίηση του design μέσω της εντολής `optDesign -postCTS`. Η εντολή αυτή μπορεί να τρέξει και σε οποιοδήποτε από τα άλλα στάδια (PreCTS, PostCTS, PostRoute).

Για περισσότερες πληροφορίες σχετικά με το/τα δέντρο/α ρολογιού, μπορούμε να χρησιμοποιήσουμε την εντολή `report_ccopt_clock_trees`. Αναφορά σχετικά με την στρέβλωση (Skew) και τις καθυστερήσεις (Insertion Delays) στο ρολόι μπορεί να προκύψει μέσω της εντολής `report_ccopt_skew_groups`.

5.3.5. Ροή μετά τη δρομολόγηση (PostRoute)

Για να ξεκινήσουμε τη διεργασία της δρομολόγησης επιλέγουμε **Route → NanoRoute → Route,** η οποία θα εμφανίσει το παράθυρο στο Σχήμα 29.



Σχήμα 29. Το παράθυρο επιλογής των ρυθμίσεων για το Nanoroute.

Στο παράθυρο αυτό μπορούμε να επιλέξουμε το **Mode**, το οποίο θα ανοίξει ένα νέο παράθυρο, όπως εικονίζεται στο ίδιο σχήμα. Στην καρτέλα DFM μπορούμε να επιλέξουμε το **effort** για σύμφωνη βελτιστοποίηση του αριθμού των via που δημιουργούνται στο κύκλωμα κατά τη δρομολόγηση, καθώς και βελτιστοποίηση των διασυνδέσεων εφόσον το επιθυμούμε.

Εφόσον επιλεγούν οι επιθυμητές ρυθμίσεις, κλείνουμε το παράθυρο του Setup και κάνουμε τις υπόλοιπες επιλογές στο αρχικό παράθυρο (Σχήμα 29 αριστερά). Σιγουρευόμαστε ότι τόσο η επιλογή **Global Route**, όσο και η επιλογή **Detail Route** είναι ενεργοποιημένες, έτσι ώστε το εργαλείο να εκτελέσει τη γενική και τη λεπτομερή δρομολόγηση όλου του κυκλώματος. Στο επόμενο τμήμα του παραθύρου μπορούμε να επιλέξουμε διάφορα θέματα που θέλουμε το εργαλείο να διορθώσει κατά τη διεργασία της δρομολόγησης, όπως το **Antenna Effect** ή το **Signal Integrity (SI)** των σημάτων του κυκλώματος. Επίσης, μπορεί να γίνει ρύθμιση του **effort** της δρομολόγησης και πόσο βαρύτητα θέλουμε να δωθεί στο χρονισμό (**Timing)** ή στη συμφόρηση **(Congestion)**. Στις υπόλοιπες ρυθμίσεις μπορούμε να ελέγξουμε πράγματα, όπως την περιοχή που θέλουμε να γίνει δρομολόγηση (αν θέλουμε να κάνουμε σε όλο το κύκλωμα, τότε αγνοούμε αυτές τις επιλογές) ή τον αριθμό από CPUs που θέλουμε να χρησιμοποιήσει ο server για να εκτελέσει τη διεργασία.

Σημείωση: Η δρομολόγηση είναι η πιο χρονοβόρα διαδικασία κατά τη σχεδίαση ενός κυκλώματος σε φυσικό επίπεδο, ανάλογα με το μέγεθος του κυκλώματος. Μεγάλα κυκλώματα μπορεί να χρειαστούν ώρες ή και μέρες για να δρομολογηθούν πλήρως από το εργαλείο!

Μετά τη διεργασία της δρομολόγησης μπορούμε να εξάγουμε τις αναφορές για τα slack, με τον τρόπο που αναφέρθηκε μερικές υποενότητες παραπάνω. Επίσης, εάν υπάρχουν παραβάσεις των χρόνων προετοιμασίας και διατήρησης, τότε μπορούμε να τρέξουμε την εντολή `optDesign -postRoute -setup -hold`, για να τις διορθώσουμε.

5.3.6. Signoff

Το τελευταίο βήμα της φυσικής σχεδίασης είναι ο έλεγχος για την ύπαρξη παραβάσεων στο κύκλωμά μας. Για αυτό τον σκοπό, μπορούμε να κάνουμε επιβεβαίωση της ορθότητας των κανόνων, γνωστό και ως Design Rule Check (DRC), αν πάμε στην επιλογή **Verify → Verify DRC**, στη γραμμή εργαλείων. Αυτή η επιλογή θα ανοίξει ένα νέο παράθυρο, όπου μπορούμε να

επιλέξουμε την περιοχή που θέλουμε να ελέγξουμε ή να απενεργοποιήσουμε κανόνες, εφόσον το επιθυμούμε. Για ελέγξουμε όλο το κύκλωμα λαμβάνοντας υπόψη όλους τους κανόνες, επιλέγουμε απευθείας την επιλογή **OK.** Ο έλεγχος DRC παραβάσεων θα ελέγξει έναν αριθμό από υποπεριοχές στο κύκλωμα και μετά την περάτωση του θα επιστρέψει τον αριθμό των παραβάσεων που υπάρχουν στο κύκλωμά μας.

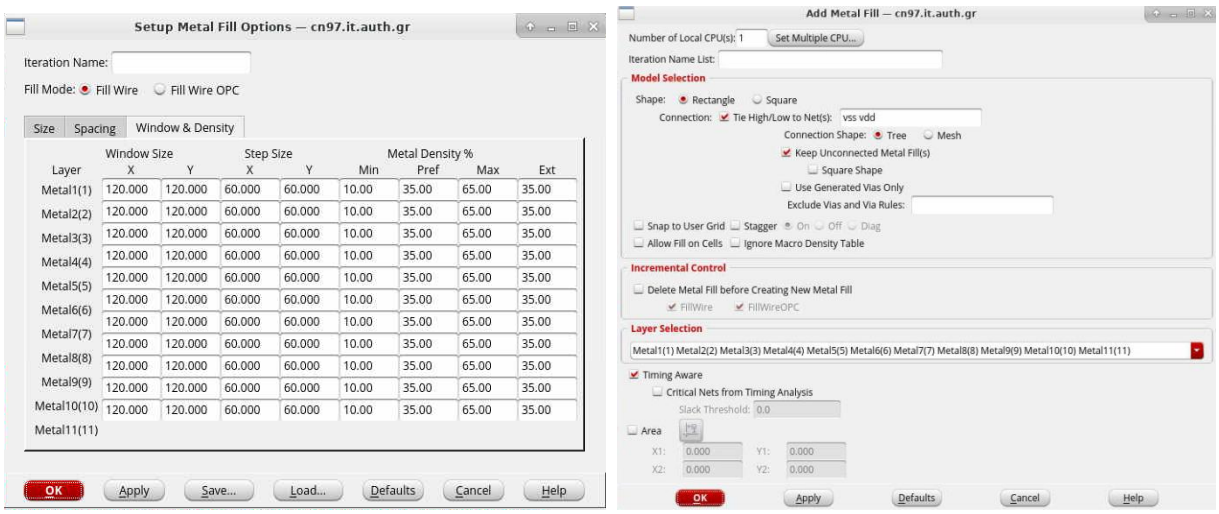
Αν θέλουμε να ελέγξουμε την ορθότητα των συνδέσεων στο κύκλωμα, μπορούμε να μεταβούμε στην επιλογή **Verify → Verify Connectivity.** Αυτό θα ανοίξει ένα νέο παράθυρο, όπου μπορούμε να επιλέξουμε τον τύπο των συνδέσεων ή απευθείας τα Nets που θέλουμε να ελέγξουμε, καθώς και τους τύπους ελέγχου που θέλουμε να κάνουμε (π.χ. ο έλεγχος τύπου *UnConnected Pin* ελέγχει αν υπάρχει κάποιος ασύνδετος ακροδέκτης).

Σημείωση: Οι παραπάνω διεργασίες μπορούν να γίνουν και από το Command Line, χρησιμοποιώντας της εντολές `verify_drc` και `verifyConnectivity` αντίστοιχα.

Σε περίπτωση που υπάρχουν παραβάσεις DRC στο κύκλωμά μας, μπορούμε να τις μελετήσουμε, χρησιμοποιώντας το παράθυρο από την επιλογή **Tools → Violation Browser**, όπου θα αναγράφονται πληροφορίες σχετικά με το που βρίσκεται η παράβαση πάνω στο σχέδιο, καθώς και για το λόγο που έχει εμφανιστεί.

Επίσης, μπορούμε να τυπώσουμε αρκετές χρήσιμες πληροφορίες για το κύκλωμά μας, χρησιμοποιώντας την εντολή `checkDesign -netlist`, όπως για την επιφάνεια, τον αριθμό των κελιών που κάνει χρήση το κύκλωμα, τα I/O και τον αριθμό από συνδέσεις με υψηλό Fanout.

Το τελευταίο βήμα της φυσικής σχεδίασης είναι η προσθήκη των κελιών γεμίσματος, διαδικασία γνωστή και ως Filler Insertion. Αυτή μπορεί να γίνει χρησιμοποιώντας την επιλογή **Route → Metal Fill → Setup/Add.** Η πρώτη επιλογή (Setup) θα ανοίξει το αριστερά παράθυρο του Σχήμα 30, όπου μπορούμε να επιλέξουμε τις ρυθμίσεις της διαδικασίας γεμίσματος, όπως το μήκος, πάχος των μετάλλων, το επιτρεπτό κενό ανάμεσά τους, καθώς και τα επιθυμητά ελάχιστα και μέγιστα όρια πυκνότητας για το κάθε μέταλλο.



Σχήμα 30. Τα παράθυρα για τη δημιουργία των Metal Fills.

Η δεύτερη επιλογή (Add) θα ανοίξει το δεξιά παράθυρο του ίδιου σχήματος, όπου μπορούμε να επιλέξουμε το σχήμα των κελιών γεμίσματος (Shape), αν επιθυμούμε να γίνει σύνδεση τους με την τροφοδοσία και τη γείωση ή όχι (Keep Unconnected Metal Fills), αν θέλουμε τα μέταλλα να επιτρέπεται να τοποθετηθούν πάνω από τα ήδη υπάρχοντα κελιά στο κύκλωμά μας (Allow Fill on Cells), αν θέλουμε η προσθήκη αυτών των κελιών να λάβει υπόψη το χρονισμό του κυκλώματος

(Timing Aware), κτλ. Επίσης, μπορούμε να ορίσουμε τα επίπεδα μετάλλου που θέλουμε να κάνουμε προσθήκη των Fills (Layer Selection).

5.3.7. Sign-Off Static time Analysis

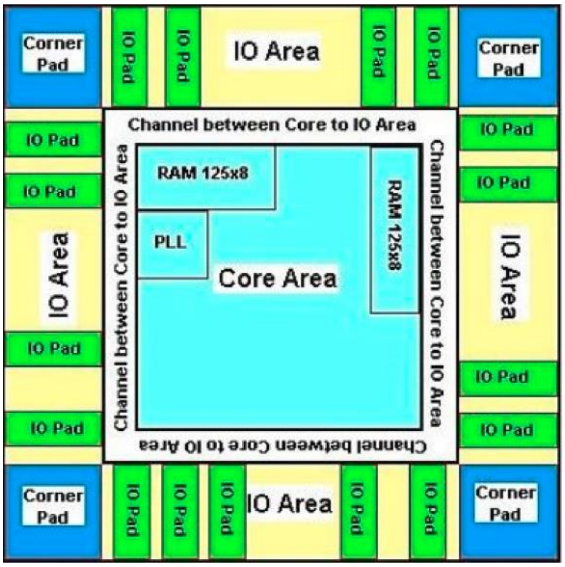
Έφοσον έχουμε ολοκληρώσει τη σχεδίαση και τον έλεγχο των παραβάσεων σχεδίασης, το τελικό βήμα για να είμαστε βέβαιοι για την ορθή λειτουργία του κυκλώματος είναι η στατική ανάλυση χρονισμού σε επίπεδο Sign-Off. Την ανάλυση αυτή αναλαμβάνει το εργαλείο της Cadence που ονομάζεται Tempus, και μπορεί να γίνει είτε μέσω της εκκίνησης του ξεχωριστά είτε μέσω κλήσης εντολών από το περιβάλλον του Innovus. Η διαφορά ανάμεσα στη στατική ανάλυση χρονισμού που είδαμε προηγουμένως είναι στη λεπτομέρεια του αρχείου των παρασιτικών του κυκλώματος. Αυτή τη φορά για τον υπολογισμό των καθυστερήσεων χρησιμοποείται το λεπτομερέστερο αρχείο παρασιτικών με κατάληξη .spef, δίνοντας μας καλύτερη εκτίμηση για την πραγματική λειτουργία του κυκλώματος. Προκειμένου να παράξουμε το αρχείο αυτό χρησιμοποιούμε πρώτα την εντολή `setExtractRCMode -engine postRoute -effortLevel signoff -coupled true` η οποία θέτει το επίπεδο λεπτομέριας του αρχείο υπολογισμού των παρασιτικών στο επίπεδο signoff. Στη συνέχεια εκτελούμε την εντολή `extractRC` η οποία δημιουργεί ένα directory με όνομα timing μέσα στο οποίο παράγεται το συγκεκριμένο αρχείο. Στη συνέχεια με την εκτέλεση της εντολής `signoffTimeDesign -reportOnly` γίνεται κλήση του Tempus για τη δημιουργία αναφοράς ανάλυσης του στατικού χρονισμού τόσο για το χρόνο setup όσο και για το χρόνο hold, χρησιμοποιώντας τα αρχείο παρασιτικών που μόλις παρήχθησαν. Σε περίπτωση που δεν έχουμε ορίσει σε προηγούμενο βήμα το μήκος καναλιού της τεχνολογίας, μπορεί το Innovus να παραπονεθεί, ωστόσο αυτό μπορεί να οριστεί με την ακόλουθη εντολή: `setDesignMode -process 45`. Δίνεται ακόμα η δυνατότητα ενοποίησης των εντολών εξαγωγής των παρασιτικών (`extractRC`) και της κλήσης του Tempus (`signoffTimeDesign`), αν παραλειφθεί η επιλογή `reportOnly` στην εντολή `signoffTimeDesign`. Με αυτόν τον τρόπο εξάγουμε τη λεπτομερέστερη ανάλυση χρονισμού του κυκλώματος.

```
...

setExtractRCMode -engine postRoute -effortLevel signoff -coupled true
extractRC
signoffTimeDesign -reportOnly

...
```

5.4. I/O Pads

Τα pins εισόδου και εξόδου αποτελούν τη διεπαφή με την οποία το κύκλωμα μας επικοινωνεί με το εξωτερικό περιβάλλον. Όταν σχεδιάζουμε το υψηλότερο επίπεδο ιεραρχίας ενός κυκλώματος δεν θέλουμε τα pins εισόδου και εξόδου να είναι απλώς λογικές συνδέσεις, αλλά θα θέλαμε να είναι φυσικές συνδέσεις, οι οποίες έχουν πραγματικές διαστάσεις και ιδιότητες κατά την προσομοίωση. Για τον λόγο αυτό είναι ιδιαίτερα σημαντικό να ορίσουμε οντότητες οι οποίες περιγράφουν τα pads εισόδου και εξόδου. Η δήλωση των pads είναι κάτι το οποίο πρέπει να γίνει χειροκίνητα καθώς το αποτέλεσμα της σύνθεσης δεν περιλαμβάνει τα κελιά αυτά.

Πέρα από τα pads εισόδου και εξόδου ένα κύκλωμα θεωρείται πλήρες όταν περιέχει στο εξωτερικό του pad ring, και συγκεκριμένα στις γωνίες, τέσσερα corner pads, τα οποία έχουν διπλή χρησιμότητα. Η μια είναι να συμπληρώνουν το κενό σχηματίζοντας ένα παραλληλόγραμμο και η σημαντικότερη είναι να παρέχουν συνέχεια στην παροχή τροφοδοσίας και γείωσης που είναι απαραίτητα για την λειτουργία των pads. Το παρακάτω σχήμα δείχνει πως περίπου θα πρέπει να φαίνεται το τελικό αποτέλεσμα.



Σχήμα 31. Παράδειγμα τοποθέτησης I/O Pads περιλαμβανομένων και αυτών στις γωνίες του ολοκληρωμένου.

5.4.1. Τα κελιά των Pads

Χρησιμοποιώντας τη βιβλιοθήκη GDPK 45nm παρατηρούμε την ύπαρξη ενός lef αρχείου με όνομα "giolib045.lef", το οποίο περιέχει περιγραφές για διαφορετικά pads corners. Μπορούμε να διαλέξουμε οποιοδήποτε τύπο pad ταιριάζει στις ανάγκες του κάθε pin. Προκειμένου να το εμφανίσουμε στο netlist μας θα πρέπει να το δηλώσουμε στον παραγόμενο κώδικα περιγραφής υλικού από το genus και συγκεκριμένα στο αρχείο genus.v. Ας δούμε ένα παράδειγμα εισαγωγής Pad και Corners.

5.4.2. Εισαγωγή στο αρχείο genus.v

Έστω ότι το Genus συνθέτει ένα αρχείο με όνομα genus.v για ένα module με όνομα example_module. Προκειμένου να εμφανίσουμε τα pads και τα corners θα πρέπει αρχικά να δημιουργήσουμε ένα καινούριο module με οποιοδήποτε όνομα επιθυμούμε μέσα στο ίδιο αρχείο genus.v. Προτείνεται η ονοματοδοσία example_module_pads προκειμένου να υπάρξει μια συνοχή στα ονόματα. Σε αυτό το module θα πρέπει να δημιουργήσουμε τις ίδιες εισόδους με το example_module και στη συνέχεια θα αρχικοποιήσουμε τα pads και τα corners. Στο τέλος θα καλέσουμε το example_module και θα του θέσουμε τις εισόδους όπως ακριβώς τις περιμένει, όπως φαίνεται στο παράδειγμα παρακάτω.

```
module example_module_pads(input1, input2, output1, output2);
    input input1;
    input [3:0] input2
    output output1;
    output [3:0] output2;
```

```
    wire input1;
    wire [3:0] input2
    wire output1;
    wire [3:0] output2;

    # Insert Pads
    PADDB pad_input1(.Y(input1), .VDD(VDD), .VSS(VSS));

    PADDB pad_input1(.Y(input2[0]), .VDD(VDD), .VSS(VSS));
    PADDB pad_input1(.Y(input2[1]), .VDD(VDD), .VSS(VSS));
    PADDB pad_input1(.Y(input2[2]), .VDD(VDD), .VSS(VSS));
    PADDB pad_input1(.Y(input2[3]), .VDD(VDD), .VSS(VSS));

    PADDB pad_output1(.Y(output1), .VDD(VDD), .VSS(VSS));

    PADDB pad_output2(.Y(output2[0]), .VDD(VDD), .VSS(VSS));
    PADDB pad_output2(.Y(output2[1]), .VDD(VDD), .VSS(VSS));
    PADDB pad_output2(.Y(output2[2]), .VDD(VDD), .VSS(VSS));
    PADDB pad_output2(.Y(output2[3]), .VDD(VDD), .VSS(VSS));

    # Insert Corners
    padIORINGCORNER pad_corner0();
    padIORINGCORNER pad_corner1();
    padIORINGCORNER pad_corner2();
    padIORINGCORNER pad_corner3();

    example_module example(input1, input2, output1, output2);
endmodule

module example_module(input1, input2, output1, output2);
...

Generated by genus

...
endmodule
```

5.4.3. VDD/VSS Pads

Προκειμένου να δημιουργήσουμε τις συνδέσεις VDD και VSS του κυκλώματος με τον εξωτερικό κόσμο με τη μορφή Pads και όχι Pins θα πρέπει να προστεθούν στον παραπάνω κώδικα οι διασυνδέσεις αυτές, όπως φαίνεται παρακάτω.

```
module example_module_pads(input1, input2, output1, output2, VSS, VDD);
    input input1;
    input [3:0] input2
    output output1;
    output [3:0] output2;

    wire input1;
    wire [3:0] input2
    wire output1;
    wire [3:0] output2;

    input VSS, VDD;
    supply0 VSS, VDD;

    ...
```

5.4.4. Δημιουργία .io αρχείου

Κατά την εισαγωγή στο Innovus θα πρέπει να θυμηθούμε να προσθέσουμε στις βιβλιοθήκες lef το κατάλληλο αρχείο (του οποίου η διαδρομή (path) στη συστοιχία είναι: mnt/apps/prebuilt/eda/designkits/GPDK/giolib045/lan/flow/rfkit/reference_libs/GPDK045/giolib045_v3.3/lef/giolib045.lef) που περιέχει την περιγραφή για τα pads. Εισάγοντας πλέον το αλλαγμένο αρχείο genus.v θα παρατηρήσουμε πλέον ότι έχει γίνει η σύνδεση των pads και περιφερειακά στο Pad Ring θα εμφανίζονται πλέον όσα Pads έχουμε ορίσει. Θα παρατηρήσουμε επίσης ότι όλα τα Corners έχουν μαζευτεί κάτω αριστερά. Προκειμένου να ορίσουμε τις θέσεις των Corners με σωστό τρόπο και σε περίπτωση που θέλουμε να αλλάξουμε τη σειρά ή τη θέση των Pads το Innovus μας δίνει ένα αρχείο με κατάληξη .io το οποίο μπορούμε να αποθηκεύσουμε, να τροποποιήσουμε και στη συνέχεια να ξαναφορτώσουμε στο εργαλείο.

Πατάμε File - Save - I/O File και στη συνέχεια επιλέγουμε Sequence και το Generate template IO file και αποθηκεύουμε το αρχείο μας.

Στη συνέχεια ανοίγουμε το αρχείο που αποθηκεύσαμε και θα παρατηρήσουμε ότι περιγράφει τον τρόπο με τον οποίο τοποθετούνται τα pads και τα corners. Σε περίπτωση που θέλετε να παίξετε με τη θέση των pads μπορείτε να πειραματιστείτε με όποιον τρόπο θέλετε, αυτό που έχει τη μεγαλύτερη σημασία όμως είναι να τοποθετήσετε τα Corner pads με τον σωστό τρόπο, προκειμένου να εξασφαλίσετε ηλεκτρική συνοχή. Παρακάτω φαίνεται το πως περιγράφουμε τις θέσεις των Corner pads των οποίων ορίσαμε προηγουμένως. Ο προσανατολισμός (orientation) παίζει ρόλο στην περιστροφή του μπλοκ, προκειμένου να μπορεί να συνδεθεί με σωστό τρόπο όλο

το Pad Ring και να δημιουργούνται οι απαραίτητες συνδέσεις VDD και VSS. Ακόμη μπορούμε να ορίσουμε την απόσταση που θέλουμε να έχουν τα pads μεταξύ τους θέτοντας στην αρχή του αρχείου την τιμή space.

…

```
(globals
    version = 3
    io_order = default
    space = 10 #units in um
)

...
(bottomleft
(inst name="pad_corner0"      orientation=R0    )
)
(bottomright
(inst name="pad_corner1"      orientation=R90   )
)
(topleft
(inst name="pad_corner2"      orientation=270   )
)
(topright
(inst name="pad_corner3"      orientation=R180  )
)
```

Παράρτημα Α

Το παράρτημα αυτό περιέχει κάποιες σημαντικές εντολές που σχετίζονται με τη δημιουργία περιορισμών για την καθοδήγη της σύνθεσης. Προέρχονται από το εγχειρίδιο αναφοράς εντολών του genus και κυρίως από τα κεφάλαια 20 και 8 (έκδοση 19.1, Σεπτέμβριος 2020). Οι εντολές και η περιγραφή τους δίνονται μόνο στα Αγγλικά για ευνόητους λόγους.

`create_clock`

```
create_clock
[-add]
[-name clock] [-domain clock_domain]
-period float [-waveform float]
[-apply_inverted {port|pin|hpin}]
[port|pin|hpin] [-comment string]
```

Creates a clock object and defines its waveform. If you do not specify any sources, but you specify the -name option, a virtual clock is created.

Options and Arguments	Description
`-add`	Allows to specify multiple clocks on the same source for simultaneous analysis with different clock waveforms. If you omit this option and a clock was already defined on a pin or port, this definition would overwrite the previous one.
`-apply_inverted {`*`port	pin	hpin`*`}`	Specifies inverted sources of the clock. **Note**: This is non-SDC option.
`-comment` *`string`*	Specifies a comment to be tagged to this command.
`-domain` *`clock_domain`*	Specifies the clock domain to which the clock belongs. Default: domain_1 **Note**: This is non-SDC option.
`-name` *`clock`*	Specifies the name of the clock. The specified name must be unique. If a clock with this name was already defined, the clock definition will be replaced. If you omit this option, the clock gets the same name as the first clock source (inverted or non-inverted). If no sources were specified for this clock, the clock name defaults to create_clock_n, where n is 1,2, and so on.
`-period` *`float`*	Specifies the length of the clock period.
`{`*`port	pin	hpin`*`}`	Specifies the non-inverted sources of the clock.
`-waveform` *`float...`*	Specifies the rise and fall edge times of the clock waveform over one clock period. The first value corresponds to the first rising transition after time zero. The numbers should represent one full clock period. If you omit this option, a default waveform is assumed: the leading edge occurs at 0 and the trailing edge occurs at the midpoint of the period, such that a symmetric clock is generated.

set_clock_latency

```
set_clock_latency
[-min ] [-max]
[-rise ] [-fall]
[-early] [-late]
[-source] [-clock clock_list] [-clock_gate]
latency {clock | port | pin | hpin}...
```

Options and Arguments	Description
{clock | pin | hpin | port}	Specifies a list of clock waveform names or a list of pins to associate with the clock value. The pin can be a port or an instance pin.
-clock *clock_list*	Associates clock value constraints, placed at the pin level, with a specific clock. This makes it possible to specify a different value per clock. If no clocks are specified, the latency value applies to all clocks that propagate to the specified pin or port. **Note**: You cannot use the -clock parameter with clock waveform objects.
-clock_gate	Allows setting the local clock value on pin or port objects. This setting overwrites the existing value of clock gating cells and is used when performing timing checks against the specified pins or ports.
-early | late	Specifies clock arrival time with respect to the early or the late time of the clock signal. In setup analysis, launch path is the late path and capture path is the early path. In hold analysis, launch path is the early path and the capture path is the late path. If neither parameter is specified, the default is both early and late. Use these parameters only with the -source parameter. **Note**: You cannot use these parameters with network value. Specify the clock value at the given operating corner. **Note**: The -min and -max parameters refer to the operating corner and the -early and -late parameters refer to the clock arrival time at the source. Use the -early and -late parameters with the -source parameter to specify the four different clock latencies
-fall	Indicates that the specified latency value applies to a fall edge.
latency	Specifies the latency value. This is a floating number.
-max	Specifies the clock value for a particular operating corner. Apply the -max parameter to the maximum operating corner.
-min	Specifies the clock value for a particular operating corner. Apply the -min parameter to the minimum operating corner.
-rise | fall	Specifies the -rise or -fall parameter to the waveform at register clock pins for ideal value and to the waveform at the source for source value. Ideal network value does not change polarity when crossing negative unate arcs.

-source	Specifies a source value. If this option is not specified, the specified latency value is considered to be a network latency.

set_clock_transition

```
set_clock_transition {-min | -max} {-rise | -fall} float
clock_list
```

Options and Arguments	Description
clock_list	Specifies a list of clocks only with ideal mode in the design, which affects the slew times on all register clock pins in the transitive fanout of the specified clocks. Specified set_clock_transition values on register clock pins in the fanout of a clock with propagated latency are ignored.
-fall	Specifies the falling edge (transition value) of the register clock pins on which the slew time is asserted.
float	Specifies the slew value for the specified clock(s).
-max	Indicates that the specified value is a maximum slew value.
-min	Indicates that the specified value is a minimum slew value.
-rise	Specifies the rising edge (transition value) of the register clock pins on which the slew time is asserted.

set_input_transition

```
set_input_transition {-min | -max} {-rise | -fall} [-clock_fall] \
[-clock clock_list] float port_list
```

Options and Arguments	Description
-clock	**Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-clock_fall	**Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-fall	Indicates that the specified value is a falling transition value.
float	Specifies the transition time for the specified port(s).
-max	Indicates that the specified value is a maximum transition value.
-min	Indicates that the specified value is a minimum transition value.
port_list	Specifies the port(s) to which the transition time applies.
-rise	Indicates that the specified value is a rising transition value

set_input_delay

```
set_input_delay [-clock clock] [-clock_fall| -clock_rise] \
[-level_sensitive] [-network_latency_included] \
```

```
[-source_latency_included] [-reference_pin {pin|port}...] \
[-rise] [-fall] [-max] [-min] [-add_delay] [-name name] \
[-group_path group] delay {port|pin|hpin|port_bus}...
```

Options and Arguments	Description
-add_delay	Specifies to add the specified input delay information for the specified pins or ports. Use this option to capture the delay information if multiple paths lead to an input pin or port that are relative to different clocks or clock edges. If you omit this option, the specified delay information overwrites all existing input delays for the specified pins or ports.
-clock *clock*	Specifies the reference clock. The input delay is defined relative to this clock. If no clocks are specified, the timing paths from the specified pins and ports will be unconstrained as captured in the report_timing report.
-clock_fall	Specifies to use the falling edge of the reference clock as reference edge. Default: -clock_rise
-clock_rise	Specifies to use the rising edge of the reference clock as reference edge. Default: -clock_rise
delay	Specifies the input delay value. This is a floating number.
-fall	Specifies that the delay is measured with respect to the falling edge of the input signal.
-group_path *group*	Specifies the name of the group to which paths ending at the specified ports or pins must be added.
-level_sensitive	Specifies that the constraint comes from a level-sensitive latch. By default, the constraint comes from an edge-triggered flip-flop.
-max	Indicates that the specified delay applies for setup analysis.
-min	Indicates that the specified delay applies for hold analysis. **Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-name *name*	Associates a name with the specified timing constraint.
-network_latency_included	Specifies that the input delay includes the clock network latency values. This implies that the tool does not need to consider the clock network latency values when optimizing this path. Sets the clock_network_latency_included attribute to true.
{port | pin | hpin | port_bus}	Specifies the input and inout pins, ports, and port buses to which the specified input delay value applies.
-reference_pin {pin | port}	Adds source and network latency of the specified reference pin to the input delay value in propagated mode. A reference pin is

	a leaf pin in the fanout cone of the clock source of the clock that you specify using the -clock parameter. **Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
`-rise`	Specifies that the delay is measured with respect to the rising edge of the input signal.
`-source_latency_included`	Specifies that the input delay includes the clock source latency values. This implies that the tool does not need to consider the clock source latency values when optimizing this path. Sets the clock_source_latency_included attribute to true.

`set_output_delay`

```
set_output_delay [-clock clock] [-clock_fall] [-clock_rise] \
[-level_sensitive] [-network_latency_included] \
[-source_latency_included] [-reference_pin pin|port ...] \
[-rise] [-fall] [-max] [-min] [-add_delay] [-name name] \
[-group_path group] delay {port|pin|hpin|port_bus}...
```

Options and Arguments	**Description**
`-add_delay`	Specifies to add the specified output delay information for the specified pins or ports. Use this option to capture the delay information if multiple paths lead to an output pin or port that are relative to different clocks or clock edges. If you omit this option, the specified delay information overwrites all existing output delays for the specified pins or ports.
`-clock` *`clock`*	Specifies the reference clock. The output delay is defined relative to this clock. If no clocks are specified, the timing paths to the specified pins and ports will be unconstrained as captured in the report_timing report.
`-clock_fall`	Specifies to use the falling edge of the reference clock as reference edge. *Default*: -clock_rise
`-clock_rise`	Specifies to use the rising edge of the reference clock as reference edge. *Default*: -clock_rise
`delay`	Specifies the output delay value. This is a floating number.
`-fall`	Specifies that the delay is measured with respect to the falling edge of the output signal.
`-group_path group`	Specifies the name of the group to which paths starting at the specified ports or pins must be added.
`-level_sensitive`	Specifies that the constraint goes to a level-sensitive latch.

-max	Indicates that the specified delay applies for setup analysis.
-min	Indicates that the specified delay applies for hold analysis. Note: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-name *name*	Associates a name with the specified timing constraint.
-network_latency_included	Specifies that the output delay includes the clock network latency values. This implies that the tool does not need to consider the clock network latency values when optimizing this path. Sets the clock_network_latency_included attribute to true.
{port | pin | hpin | port_bus}	Specifies the output and inout pins, ports, and port buses to which the specified output delay value applies.
-reference_pin {pin | port}	Adds source and network latency of the specified reference pin to the output delay value in propagated mode. A reference pin is a leaf pin in the fanout cone of the clock source of the clock that you specify using the -clock parameter. **Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-rise	Specifies that the delay is measured with respect to the rising edge of the output signal.
-source_latency_included	Specifies that the output delay includes the clock source latency values. This implies that the tool does not need to consider the clock source latency values when optimizing this path. Sets the clock_source_latency_included attribute to true.

set_load

```
set_load { [-pin_load] [-subtract_pin_load] [-wire_load] \
port... | hnet... | port_bus ...} [-min] [-max] capacitance
```

Options and Arguments	Description
capacitance	Specifies the capacitance. This is a floating number. The capacitance is in units from the target technology library.
-max	Specifies the maximum capacitance. The maximum capacitance is used to compute worst-case delays in BcWc analysis mode and late path delays in OCV mode. It is also used to check the maximum capacitance design rule.
-min	Specifies the minimum capacitance. The minimum capacitance is used to compute best-case delays in BcWc analysis mode and early path delays in OCV analysis mode. **Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-pin_load	Specifies the load of pins connected to this port. Sets the

	external_pin_cap attribute on the port. This option applies only to ports. This is the default option if neither the -pin_load option nor the -wire_load option is not specified.
{port | hnet | port_bus}	Specifies the port, net, or port_bus objects to which the specified capacitance applies.
-subtract_pin_load	Subtracts the capacitance of the pins connected to the net of the port from the specified capacitance value before it applies it to the ports. This option applies only to ports. Can be used if nets are specified in the object list.
-wire_load	Interprets the specified capacitance as the external wire capacitance of the port. Use this option only for ports. Use this option with the port_net_list argument if the list contains only ports. Sets the external_wire_cap attribute on the port.

set_driving_cell

```
set_driving_cell
{-cell cell | -lib_cell cell} [-library library] [-from_pin pin] \
[-pin pin] [-input_transition_rise float] \
[-input_transition_fall float] [-rise] [-fall] [-max | -min] \
[-multiply_by integer] [-no_design_rule] [-dont_scale] \
[-clock clock_list] [-clock_fall] port_list
```

Options and Arguments	Description
-cell *cell*	Specifies the name of the lib_cell used to drive the port.
-clock	**Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-clock_fall	**Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-dont_scale	**Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-fall	Indicates that the information applies only to a falling transition on the ports.
float	Specifies the transition time for the specified port(s).
-from_pin *pin*	Specifies an input pin on the specified cell. The command uses the drive of the timing arc from this pin to the pin specified with the -pin option.
-input_transition_fall *float*	Specifies the input falling transition time associated with the -from_pin option.

	Default: 0
-input_transition_rise *float*	Specifies the input rising transition time associated with the -from_pin option. Default: 0
-lib_cell *cell*	Specifies the name of the lib_cell used to drive the port.
-library *library*	Specifies the name of the library to which the lib_cell belongs. If this option is omitted, the command searches in all available libraries.
-max	Indicates that the driver cell information applies to setup analysis. By default, if both the -max and -min options are omitted, the information applies to setup analysis.
-min	Specifies the early drive assertion. **Note**: Genus ignores this option for optimization and reporting purposes. Physical flows will honor it as part of Innovus initial placement. The write_sdc command will write the option out for consumption by downstream tools.
-multiply_by *integer*	Specifies the total number of external drivers. The number of additional external drivers is the specified number minus 1 and this value will be set on the external_non_tristate_drivers port attribute. Default: 1
-no_design_rule	Indicates to ignore the design rule violations seen by the external driver pin.
-pin *pin*	Specifies the output pin of the lib_cell which will be used to drive the port. If this option is omitted, the command chooses an output pin at random.
port_list	Specifies the port(s) to which the external driver applies.
-rise	Indicates that the information applies only to a rising transition on the ports. By default, if both the -fall and -rise options are omitted, the information applies to the rising and falling transitions on the ports.

write_template

```
write_template [-dft] [-power] [-cpf] [-retime] [-physical] [-performance] [-area] [-no_sdc] [-n2n] [-yield] [-full] [-simple] [-split] [-multimode] -outfile file
```

Παράγει ένα φιλικό προς το χρήστη template script με τις εντολές και ιδιότητες που απαιτούνται για να τρέξουμε το Genus. Χρησιμοποιήστε τις παραπάνω επιλογές τις εντολής για να συμπεριλάβετε συγκεκριμένες ομάδες εντολών και ιδιοτήτων στο script αυτό.

Options and Arguments	**Description**
-area	Writes out a template script for area-critical designs
-cpf	Writes out a template script for the Common Power Format (CPF) based flow and a template CPF file (template.cpf). The template CPF file is read in the template script with the read_cpf command.

-dft	Writes out a template script for the test synthesis (DFT) flow.
-full	Writes out DFT, power, and retiming commands and attributes along with the basic template.
-multimode	Writes out a template script for multi-mode analysis.
-n2n	Writes out the template script for netlist to netlist optimization in Genus. Use with the -dft and -power options to include the DFT and power attributes and commands.
-no_sdc	Writes out clock delays and input and output delays in the Genus format.
-outfile *string*	Specifies the name of the file to which the template script is to be written.
-performance	Writes out a template script which enables some advanced optimization algorithms that can improve QoR and that have an impact on the runtime.
-physical	Writes out a template script for the physical flow.
-power	Writes out power attributes and commands.
-retime	Writes out retiming attributes and commands.
-simple	Writes out a simple template script. You cannot use this option with the -split, -area, -dft, -cpf, -multimode, -physical, -power, -retime, or -full options.
-split	Writes out a template script with a separate setup file which contains the root attributes and setup variables. If you specified the -dft option with the -split option, an additional file is created which contains the DFT design attributes, test clock and scan chain information. If you specified the -power option with the -split option, an additional file is created which contains the leakage and dynamic power, and clock-gating setup information.
-yield	Writes out a template script for yield.

Παραδείγματα:

- Το ακόλουθο παράδειγμα εξάγει το template script «template.g», ένα υπόδειγμα ρυθμίσεων «setup_template.g» και ένα αρχείο «power_template.g» και τα περιλαμβάνει στο αρχείο «template.g»

```
write_template -split -dft -power -outfile template.g
```

- Το ακόλουθο παράδειγμα παράγει ένα απλό template script χωρίς γκρουπ μονοπατιών και κόστος και χωρίς καθόλου μεταβλητές και ιδιότητες σχετικές με ισχύ ή DFT

```
write_template -simple -outfile template.g
```

- Το ακόλουθο παράδειγμα παράγει ένα template script για κυκλώματα που είναι κρίσιμα ως προς την επιφάνεια

```
write_template -area -outfile template.g
```

signoffTimeDesign

`signoffTimeDesign [-help] [-noEcoDB] [-noExpandedViews] [-outDir` *`string`*`] [-prefix` *`string`*`] [-reportFullClockPath] [-reportOnly]`

Runs signoff timing analysis using extraction (Quantus) and Tempus in batch mode, and generates timing reports and ECO DB for each view. The command prints out a timing summary table per view and all the views combined.

Options and Arguments	Description
-help	Prints out the command usage.
-noEcoDB	Disables generation of ECO timing DB.
-noExpandedViews	Disables printing of expanded timing per view.
-outDir *string*	Specifies the directory to save reports.
-prefix *string*	Specifies prefix for timing report file names.
-reportFullClockPath	When this parameter is specified, the detailed timing report that is generated contains the full clock path.
-reportOnly	Skips parasitic extraction and use the parasitics in memory.

setExtractRCMode

`setExtractRCMode [-help] [-reset] [-assumeMetFill` *`scalevalue`*`] [-capFilterMode {relOnly | relAndCoup | relOrCoup}] [-compressOptMemRCDB {true | false}] [-coupled {true | false}] [-coupling_c_th` *`value_in_fF`*`] [-defViaCap {true | false}] [-effortLevel {low | medium | high | signoff}] [-engine {preRoute | postRoute}] [-extraCmdFile fileName] [-extractionFillStreamMapFile` *`fileName`*`] [-hardBlockObs {true | false}] [-incremental {true | false}] [-layerIndependent {0 | 1}] [-lefTechFileMap` *`fileName`*`] [-localCpu number_of_cpu] [-pvs_fill {true | false}] [-qrcCmdFile` *`fileName`*`] [-qrcCmdType {auto | partial | custom}] [-qrcOutputMode {rcdb | spef}] [-qrcRunMode {concurrent | sequential}] [-relative_c_th` *`value`*`] [-signoff_stream_layer_map` *`fileName`*`] [-total_c_th` *`value_in_fF`*`] [-tQuantusModelFile` *`fileName`*`] [-tsvSubcktFile` *`file_name`*`] [-turboReduce {true | false | auto}] [-useQrcOAInterface {true | false}] [-useShieldingInDetailMode {true | false}] [-viaCap {true | false}]`

Runs signoff timing analysis using extraction (Quantus) and Tempus in batch mode, and generates timing reports and ECO DB for each view. The command prints out a timing summary table per view and all the views combined.

`-help`

Outputs a brief description that includes type and default information for each `setExtractRCMode` parameter.

For a detailed description of the command and all of its parameters, use the man command: `man setExtractRCMode`.

Takes into account the metal fill in the design. For postRoute extraction, only extraction engine with `-effortLevel low` supports this parameter.

In postRoute extraction with `-effortLevel low`, *`scalevalue`* is not required and will be ignored. Detail extraction uses the `FILLACTIVESPACING` values from LEF to account for metal fill in the design. If the `FILLACTIVESPACING` values are not specified in the LEF file, the software uses the default value, which is `0.6um`.

In preRoute extraction, when calculating the capacitance of a wire segment in layer *`N`*, this parameter always assumes layer *`N-1`* and layer *`N+1`* to be filled. Specify one of the following values:

`-assumeMetFill` *`scalevalue`*

- `0`: Assume no metal fill in layer *`N`*.
  - When you do not specify this parameter, the software assumes no metal fill in layer *`N`*. This is equivalent to setting the *`scalevalue`* parameter to `0`.
- `1`: Assume full metal fill at minimum spacing distance.
  - Where there is a minimum of signal-to-signal spacing, the cross-coupling capacitance will be extracted and reported.
  - Where there is no wire within the minimum spacing, a capacitance to ground will mimic minimum-spaced metal fill in that location.
- x: Capacitance value is calculated once with full metal fill and once without metal fill. The user-specified scale value (decimal value between 0 and 1) determines the interpolation point between the two calculated values, for example, `0.5`

Default: 0

`-capFilterMode { relOnly | relAndCoup | relOrCoup }`

Specifies the capacitance filtering mode. The filtering modes are conditions which define how to use the `-coupling_c_th` and `-relative_c_th` values, or their combination. Based on the filtering mode specified, the software determines the nets in the design, which will have their coupling capacitance lumped to the ground.

Note: TQuantus only supports the `relandCoup` mode of this parameter. It does not support the `relOnly` and `relOrCoup` modes as they are for old nodes only.

Note: The behavior of the `-total_c_th` filter is independent of the `-capFilterMode` parameter setting. Nets that have a total capacitance value lower than the value specified with the `-total_c_th` parameter will have their coupling capacitances grounded regardless of the -

`capFilterMode` setting.

You can use one of the following modes:

- `relOnly`: Grounds the coupling capacitance of nets based on the threshold value set by the `-relative_c_th` parameter.
- `relAndCoup`: Grounds the coupling capacitance of nets only when it is lower than the threshold value specified with the `-relative_c_th` parameter as well as the `-coupling_c_th` parameter. In this mode, you enforce maximum restriction on grounding the coupling capacitances.
- `relOrCoup`: Grounds the coupling capacitance of nets if it is lower than the threshold value specified with one of the `-relative_c_th` or `-coupling_c_th` parameters.

Default: For TQuantus, IQuantus, and Standalone Quantus extraction, the default value is `relAndCoup`. For postRoute extraction with `-effortLevel low`, the default value is dependent on the process node specified with the `setDesignMode -process` command. If the process node is 130nm or below, the default value is `relAndCoup` and if the process node is above 130nm, the default value is `relOnly`.

`-compressOptMemRCDB {true | false }`

Generates compressed RCDB data.

Default: `false`

Note: This functionality reduces the disk usage but leads to marginal increase in run time.

`-coupled {true | false}`

Separately outputs grounded and coupling capacitance. The `-coupled` false value is typically used for static timing analysis. Signal Integrity analysis requires the coupled mode.

Default: `true` (for postRoute extraction)

Note: The TQuantus extraction engine supports this parameter.

`-coupling_c_th` *`value_in_fF`*

Specifies the threshold value that determines when the extractor lumps a net's coupling capacitance to ground. This parameter is applicable only when the `-coupled` parameter is set to `true`.

The software decouples the coupling capacitance of nets when the total coupling capacitance between the pair of nets is lower than the threshold specified with this parameter. The actual decoupling also depends on the setting of the `-capFilterMode` parameter.

Default: `0.2fF`. If the `setDesignMode -process` *`process_node`* command is used, the software assigns the recommended default value for this parameter based on the process node specified.

Note: See the `setDesignMode` command for the list of default values

that are applied based on the process node setting.

Note: The TQuantus extraction engine supports this parameter.

`-defViaCap {true | false}`

Takes into account the via capacitance in the preRoute RC extraction results. Specify a value of `true`, if you are experiencing correlation issues caused by deviations in clock latency when running preRoute RC extraction.

Default: `false`

Note: For 20nm and below nodes, the default value of this parameter is set to `true`.

`-effortLevel { low | medium | high | signoff }`

Specifies the postRoute engine variant that you want to use for performing RC extraction.

Default: `low`

Note: TQuantus (`-effortLevel medium`) is the default extraction engine in the postRoute flow for 65 nm and below designs. However, TQuantus and IQuantus require a Quantus Techfile. Therefore, Native Detailed (`-effortLevel low`) engine remains the default engine if no Quantus Techfile has been defined.

This parameter supports the following values:

- `low`: Invokes the native detailed extraction engine. This is the same as specifying the -engine postRoute setting.
- `medium`: Invokes the TQuantus extraction mode. This engine supports distributed processing.

  TQuantus engine is recommended for process nodes < 65nm.

  Note: This setting does not require a Quantus license.

- `high`: Invokes the Integrated-Quantus (IQuantus) extraction engine.

  IQuantus provides superior accuracy compared to TQuantus. IQuantus is recommended for extraction after ECO. In addition, IQuantus supports distributed processing.

  Note: IQuantus requires a Quantus-XL license.

- `signoff`: Invokes the Standalone Quantus extraction engine. This engine choice provides the highest accuracy. The engine has several runModes, thereby, providing maximum flexibility.

`-engine { preRoute | postRoute }`

Specifies the extraction engine to use.

Default: `preRoute`

- `preRoute`: Uses preRoute engine. RC extraction is done by the fast density measurements of the surrounding wires; coupling is not reported.

  Use this mode to perform extraction on pre-routed designs after clock tree synthesis.

- `postRoute`: Uses postRoute engine. RC extraction is done by the detailed measurement of the distance to the surrounding wires; coupling is reported. The `-effortLevel` parameter further specifies which postRoute engine is used for balancing performance versus accuracy needs.

`-extraCmdFile` *`fileName`*

Specifies the Quantus extra command file for TQuantus/IQuantus extraction. The extra command file supports the following variables:

- `log_file -dump_options {`**`true`** `| false}`
- `extraction_setup -copy_port_to_obs {`**`true`**`|false}`
- `extraction_setup -remove_net_pin_overlap {true |`**`false`**`}`
- `metal_fill -type {grounded |` **`floating`** `| virtual}`
- `output_setup -compress_temporary_files {true |` **`false`**`}`

Notes:

- TQuantus only supports advanced virtual metal fill in `metal_fill` in the extra command file. It does not support the grounded and floating options.
- When the `metal_fill` type is virtual, provide the absolute paths to the vmf rule file and the metal schema file. For example,

```
metal_fill \

  -type virtual \

  -enable_advanced_virtual_fill true \

  -vmf_metal_scheme_file
  .../saveTestcase_vmf/missing_vmf_files/abc.vmf_me
  tal_scheme \
```

```
-vmf_rule_file
.../saveTestcase_vmf/missing_vmf_files/abc.vmf_ru
le
```

- The `-extraCmdFile` parameter becomes "non-portable" if the vmf rule file and/or schema file have been specified. This means that the two file paths will not be automatically modified or ported. You can modify the file paths, if required. To make the extra command file portable, ensure that the vmf rule file and schema files are available at the locations specified in the extra command file.
- The **bold** text in the above example indicates that these values are used by default when the commands are not explicitly specified in the extra command file. For example, IQuantus extracts the "**`floating`**" metal fills by default when either the `-extraCmdFile` option is not specified or the `metal_fill` variable is not specified in the extra command file.
- This parameter is not available on the IBM platform.

This parameter is specified only when `-pvs_fill` parameter is set to `true`. When the `-pvs_fill` parameter is set to `true`, a Quantus stream/oasis map Tcl command is required to map the PVS layer/dtype data to the corresponding Quantus layers.

If the `pvs_fill.data` is generated based on stream (GDS), the map file should be in the following format:

```
#layer gds_layer gds_dtype

<layer_name> <layer_number> <data_type>

…
```

If the `pvs_fill.data` is generated based on OASIS, the map file should be in the following format:

`-extractionFillStreamMapFile` *`fileName`*

```
#layer oasis_layer oasis_dtype

<layer_name> <layer_number> <data_type>

…
```

Where,

- `layer_name` is the layer name in the design
- `layer_number` is the layer ID in the GDS/OASIS file
- `data_type` is the datatype specified in the GDS/OASIS file

For example,

```
M1  31  0
```

```
M1  31  1

M2  32  0

M2  32  1
```

Note: This parameter only affects signoff Quantus extraction.

Default: `" "`

`-hardBlockObs {true | false}`

Makes cell geometries on metal layers from obstruction, and power and ground pin shapes visible to the extractor for hard blocks only. The software recognizes the following three types of hard blocks:

- Hard macro (`CLASS BLOCK` in LEF file)
- Blackbox (`CLASS BLACKBOX` in LEF file)
- Partition (Committed during the Innovus session)

Specify a value of `true` when routing goes over the listed hard blocks. A value of `true` removes the optimism for such routes, and provides better correlation with sign-off extraction without significantly impacting runtime.

Note: This parameter is not supported when the extraction engine is TQuantus.

Default: `false`

`-incremental {true | false}`

Uses incremental IQuantus extraction mode.

If the incremental mode is set after performing initial extraction on full design, extraction checks for design changes, and if the amount of changes are below threshold, extraction is rerun in incremental mode on the modified areas only.

Incremental mode can provide significant performance improvement during signal integrity optimization phase.

Default: `true`

`-layerIndependent {0 | 1}`

By default, this parameter is set to `1`. In this mode, layer-independent extraction is enabled. This mode of extraction is recommended for designs with layer stacks that have significant variations in RC characteristics. This mode uses advanced techniques to avoid timing jumps due to differences in routing-layer assignments between preRoute and postRoute stages.

When set to `0`, layer-independent extraction is disabled.

Note: PreRoute extraction scale-factors should be computed without

setting this mode.

Default: `1`

`-lefTechFileMap` *`fileName`*

Specifies the location of the optional layer map file for the IQuantus and Standalone Quantus engines.

Note: For IQuantus, layer mapping between the LEF or DEF file and the Quantus technology file is done automatically. This parameter is only necessary if this mechanism fails. If necessary, you can use a copy of the automatically generated layer map file `extLogDir/IQRC_*/<design>.layermap.log` and modify it for your own use.

Layer map file is required for use with Standalone Quantus.

For TSV designs, you are required to provide the name of the layer map file for IQuantus and Standalone Quantus.

Note: If used, the layer map file must be in the Common Command Language (CCL) format.

The following is an example of the layer map file format:

```
extraction_setup \
-technology_layer_map \
# <LEF/DEF Name> #<Tech File Name>
M1 METAL_1 \
VIA1 VIA_1 \
M2 METAL_2 \
```

Note: For cell-level IQuantus layer map, the DEF/LEF layer names are case sensitive, but ICT layer names are case-insensitive.

Note: TQuantus is not impacted by this parameter because it always automatically generates the layer map.

`-localCpu` *`number_of_cpus`*

Specifies the number of CPUs to be used for IQuantus extraction.

Note:

- This option is only applicable for IQuantus extraction engine.
- When specified, it overrides the `setMultiCpuUsage -localCpu` option for IQuantus.

Default: As specified in the `setMultiCpuUsage -localCpu` setting.

`-pvs_fill`

When set to `true`, the PVS fill data attached to the Innovus database is sent to Quantus so that the effects of the fill data are extracted. This parameter only affects signoff Quantus extraction. It has no effect on tQuantus or iQuantus extraction done inside Innovus. This parameter requires that either the `-extractionFillStreamMapFile` parameter or the `-signoff_stream_layer_map` parameter is specified.

Default: `false`

`-qrcCmdFile` *fileName*

Specifies the user created Quantus command file to be used for running Standalone Quantus. This command file should be in the new CCL syntax.

Note: This parameter is only valid if `-effortLevel signoff` and `-qrcCmdType` partial or custom is selected.

`-qrcCmdType {auto | partial | custom}`

Specifies whether to derive the settings for running Quantus from the Innovus settings, or from user-created command file using the `-qrcCmdFile` parameter. The `-qrcCmdType` parameter is only valid if `-effortLevel signoff` is selected. This parameter supports the following values:

- `auto`: Derives the Quantus settings automatically from the Innovus parameter settings.

  Note: In this mode, the `-lefTechFileMap` parameter is required to be specified.

- `partial`: Appends user commands from the qrcCmdFile to the settings derived from the Innovus parameter settings. You must provide the partial mode command file in CCL syntax. Use the `partial` command file only when you add extraction directives for those options that cannot be set in Innovus.

  Following are some important restrictions:

  - Do not specify the following options in the partial mode command file:
    - Threshold setting
    - Coupled/decoupled mode
    - Quantus Techfile (s)
    - Operating Temperature
  - Do not specify the "`-directory_name`" and "`-temporary_directory_name`" in the `output_setup` section of the command file
  - Do not specify output file using `output_setup -file_name` command in the partial command file

  Note: The multi-temperature multi-corner extraction (MTMC) feature is enabled in the partial mode only if process technology is not specified in the user command file. In MTMC extraction, for RC corners with same Quantus technology file but different temperatures, resistance is extracted for multiple corners while capacitance is extracted only for a single corner.

  Note: There is no checking for syntax or content of this file. This file is appended as is to the automatically created Quantus

command file.

- `custom`: Gets the complete Quantus settings from the user-created command file. Except for input and output, no settings from Innovus parameters are used. You must provide the complete Quantus command file in CCL syntax. Following are some important restrictions for using custom command file:
  - Do not specify an input DEF/OA DB in the custom mode command file
  - Do not specify the "`-directory_name`" and "`-temporary_directory_name`" and "`-file_name`" in the `output_setup` section of the command file.
  - Specify the output SPEF file with the following convention for the name:`<design.spef>`.
    In the MMMC mode, the output SPEF files will be named: `<design>_<corner>_<Temperature>.spef`.
    Here "design" refers to the top cell name.
    Note: Do not prefix or postfix the output file name.
  - Specify the Quantus Techfile(s) to be used. In the MMMC mode, the corner names should match the names used in the `create_rc_corner` command
  - Provide Quantus Techfile layermapping information in the custom command file
  - If GDS is provided for cell contents, provide a Quantus Techfile to GDS layer number matching

  Note: The MTMC extraction feature is not enabled in the custom mode of Quantus extraction.

  Note: There is no checking for syntax or content of this command file prior to running the extraction. The command file will be used as is. If the output file is not specified correctly, the resulting parasitic file will not be read into the Innovus RC database.

`-qrcOutputMode {rcdb | spef}`

Sets the Quantus output write mode to RCDB or SPEF.

Note: This parameter is only valid if `-effortLevel` is `signoff`.

Default: `spef`

`-qrcRunMode {concurrent | sequential}`

Specifies the Quantus run mode in MMMC designs.

Default: *concurrent*

Note*:* For MMMC designs, in Quantus partial command type, do not specify the output SPEF file name and RC corner name in the command file to run Quantus in concurrent or sequential mode.

For MMMC designs, in Quantus custom command type, do not specify the output SPEF file name and RC corner name together in the

command file. Quantus sequential mode is not supported in custom command type.

This parameter supports the following values:

- `Concurrent`: Runs Quantus for all active corners concurrently.
- `Sequential`: Runs Quantus for all active corners sequentially.

`-relative_c_th` *value*

Sets a ratio threshold value that determines when the extractor lumps a net's coupling capacitance to ground. This parameter is applicable only when the `-coupled` parameter is set to `true`.

If the total coupling capacitance between a pair of nets is less than the percentage (specified with this parameter) of the total capacitance of the net with the smaller total capacitance in the pair, the coupling capacitance between these two nets will be considered for grounding. The actual decoupling also depends on the setting of the `-capFilterMode` parameter.

Default: `1`.

The range of this value is 0 to 1. If the `setDesignMode -process` *`process_node`* command is used, the software assigns the recommended default value for this parameter based on the process node specified.

Note: See the `setDesignMode` command for the list of default values that are applied based on the process node settings.

Note: The tQuantus extraction engine supports this parameter.

`-reset`

Resets the extraction status of the loaded design and the `setExtractRCMode` parameters to their default values.

The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setExtractRCMode` parameters to their default values. If you specify parameters after `-reset` , the software resets only those parameters to their default values. Any values that do not already match the default value and are reset will be reported.

This parameter is specified only when the `-pvs_fill` parameter is set to `true`. When the `-pvs_fill` parameter is set to `true`, a Quantus stream/oasis map Tcl command is required to map the PVS layer/dtype data to the corresponding Quantus layers.

`-signoff_stream_layer_map` *`fileName`*

If the `pvs_fill.data` is generated based on stream (GDS/OASIS), the map file should be in the following format:

```
#layer gds_layer gds_dtype use-type color-ID

<layer_name> <layer_number> <data_type> <use-type>
```

<color-ID>

…

Where,

- `layer_name` is the LEF layer name in the design
- `layer_number` is the layer ID in the GDS/OASIS file
- `data_type` is the datatype of the layer in the GDS/OASIS file
- `use_type` is the keyword for identifying the use of the data. The valid values are:
  -
    - `gray`: library data in the GDS/OASIS file
    - `fill`: regular metal fill data in the GDS/OASIS file
    - `active`: active metal fill data in the GDS/OASIS file
- `color-ID` is the color ID. Use "`1`" or "`2`" if the layer is colored, or use hyphen "`-`" if the layer is not colored

For example,

```
M1   31   0 gray -

M1   31   1 fill -

M2   32   0 gray -

M2   32   1 fill -
```

Note: This parameter only affects signoff Quantus extraction.

Default: `""`

`-total_c_th` *`value_in_fF`* Specifies the threshold value (femtoFarads) that determines when the extractor lumps a net's coupling capacitance to ground. This parameter is applicable only when the `-coupled` parameter is set to `true`.

The software grounds the coupling capacitances for the nets which have a total capacitance value less than the value specified with this parameter.

Default: `0 fF`. If the `setDesignMode -process` *`process_node`* command is used, the software assigns the recommended default value for this parameter based on the process node specified.

Note: See the `setDesignMode` command for the list of default values that are applied based on the process node settings.

Note: The TQuantus extraction engine supports this parameter.

`-tQuantusModelFile` *`fileName`*

Specifies the name of the tQuantus model file to be used for postRoute effort-level medium extraction.

You can specify the model file as follows:

```
setExtractRCMode -engine postRoute -effortLevel medium -
tQuantusModelFile tQuantus_model.bin
```

For details of the use model, see Creating RC Model Data in TQuantus Model File in the *Innovus User Guide*.

Default: `""`

Note: If this file is not specified, it will be generated automatically by the software.

`-tsvSubcktFile` *`file_name`*

Specifies the name of the Through Silicon Via (TSV) subckt file to be used. The file contains the .subckt model information for TSV. The file has a Spice format. A sample file is shown below.

```
.subckt TSV top bottom
R1 top n2 0.03075
R2 n2 bottom 0.03075
C1 n2 0 28f
.ends
```

Where,

`top` and `bottom` refer to the terminal nodes on the top and bottom layers of the TSV

`n2` is the internal node between the top and bottom layers

`R1 top n2 0.03075` means that a resistor with a resistance of `0.03075 Ohms` is added between the `top` and `n2` nodes

`R2 n2 bottom 0.03075` means that a resistor with a resistance of `0.03075 Ohms` is added between the `bottom` and `n2` nodes

`C1 n2 0 28f` means that a ground capacitance of `28 fF` is added on the `n2` node

Note: The IQuantus engine might not keep the subckt connectivity as it is. The optimization of RES network will remove node n2 (if there is no other connections) and keep a single resistor between the top and bottom nodes with an accumulated resistance. For example, in the above example, the accumulated resistance will be `0.0615 Ohms`. Capacitance will also be moved to a “higher layered node”. In the above example, this layer is `top`.

Default: `""` (empty string)

`-turboReduce {true|false|auto}`

Controls the turbo reduction feature for tQuantus/IQuantus, and Standalone Quantus extraction. It has no effect on tQuantus.

This is a cell-level reduction that provides a complimentary reduction

capability and only affects the coupling capacitors. It has no effect on lumped capacitor extraction.

Default: `auto`

When set to `auto`, the software automatically turns `on` turbo reduction and proceeds when the required Quantus license is available. However, if the required Quantus license is not installed in the license server, the software automatically turns off Turbo reduction and proceeds with a warning message. If the license is installed in the license server but temporarily not available, then two scenarios are possible. First, if the license wait mode is not turned on, then Quantus will look for the license right away and error out because the license is not available. Second, if the license wait mode is turned on, then IQuantus will wait and check for the license periodically until timeout. If the license is available before the timeout, turbo reduction will be performed but if the license is not available before timeout, IQuantus will error out.
When set to `true`, the turbo reduction engine is `on`. However, if the required Quantus license is not installed in the license server, then Standalone Quantus and IQuantus will error out with a message and the run will stop. It is because setting the option to true means the user really wants to perform turbo reduction.

When set to `false`, the turbo reduction engine is switched `off`.

For details, see the table titled, Behavior of Turbo Reduction, below. However, note that the licensing rules given in this table are not applicable to tQuantus because tQuantus does not require any Quantus license.

`-useQrcOAInterface {true | false}`

When set to `true`, specifies that OpenAccess (OA) interface be used for invoking Quantus extraction instead of using the DEF-based flow.

Default: `false`

`-useShieldingInDetailMode {true | false}`

Assumes shielding for any wire segment of a clock net that does not have any shielding yet, and the space between the wire segment and its neighbor is greater than or equal to the following:

`2s+w`

where,

s = minimum spacing or non-default spacing

w = minimum width or non-default width

This parameter is available in `postRoute -effortLevel low` mode only.

Default: `false`

`-viaCap {true | false}` Extracts capacitance values considering part of the via geometries that do not overlap with the wire geometries. Via geometries that do not overlap the wire geometries are often found if you are designing using newer design technologies such as 90nm or when using multiple cut via.

This parameter is only available in `postRoute -effortLevel low` mode.

Default: `true`

Παράρτημα Β.

Στο παράρτημα αυτό παρέχονται κάποιες βασικές και συχνά χρησιμοποιούμενες εντολές για τη διαδικασία της σύνθεσης. Οι εντολές αυτές μπορούν να χρησιμοποιηθούν απευθείας στη γραμμή εντολών του genus ή σε script κατά την εκτέλεση της σύνθεσης.

Βασικές εντολές αναζήτησης

Purpose	Command
To get all leaf instances of a design	`get_db insts` `inst:des1/core/d_reg[0] inst:des1/core/d_reg[1]. . .`
To query for all lib cells	`get_db lib_cells`
To get all base cells	`get_db base_cells`

Given an *inst*, to determine its *libcell* and *base_cell*	`get_db inst:core/out_reg[0] .lib_cell`
	`lib_cell:libset1/slow/SDFSNQD1BWP`
	`get_db inst:core/out_reg[0] .base_cell`
	`base_cell:SDFSNQD1BWP`

To find all CP leaf pins in a design	`get_db pins -if {.base_name==CP}` `get_db pins */CP`
All instances of a certain cell type	`get_db insts -if {.base_cell.name==DFFX1}`

Εύρεση objects με όνομα και τύπο

Purpose	Command
To get the root-level attribute	`get_db cpu_runtime`
To get the *state* attribute on a top-level design	`get_db design:<design_name> .state`
To get the *base_cell* attribute on a leaf instance	`get_db inst:<inst_name> .base_cell`
To get the name of the *base_cell* of a leaf instance (this is called chaining)	`get_db inst:<inst_name> .base_cell.name`
To return a unique list of all *base_cells* used in a design	`get_db insts .base_cell -uniq`
(Slightly more complicated chaining example) To return the set of unique *base_pins* (i.e., *lib_pins*) of all instances in a design	`get_db insts .pins.base_pin.name -uniq`

To find all designs	`get_db designs`
To find all leaf instances in all the designs	`get_db designs .insts` Or `get_db insts`
To find all sequential leaf instances in all the designs	`get_db designs .insts -if {.is_sequential}`
To find all hierarchical instances (including unresolved *ref* instances) in all the designs	`get_db designs .hinsts`
To find all combinational (and tristate) leaf instances in the current design	`get_db current_design .insts -if {.is_combinational}`
To find all combinational (and tristate) leaf instances under the current hierarchy	`get_db . .insts -if {.is_combinational}`
To find (outside) pins on hier-instances (recursively)	`get_db hpins`
To find (outside) input pins on hier-instances (recursively)	`get_db hpins -if {.direction == in}`
To find pins on a leaf-instance	`get_db <inst_name> .pins`
To return all leaf sequential instances locally (i.e., non-recursively) under top-design *m1*	`get_db design:m1 .local_insts -if {.is_sequential}`
To get all instances of a certain cell type	`get_db insts -if {.base_cell.name==DFFX1}`

Εύρεση object βασισμένα στις τιμές ιδιοτήτων (attributes)

Purpose	Command
To find all flop leaf instances (not latches or timing-models)	`get_db designs .insts -if {.is_flop == true}`
To return all preserved leaf-instances locally under *hinst m1/m2*	`get_db hinst:m1/m2 .local_insts -if {.preserve != false}`
To get the name of the *base_cell* of a leaf instance (this is called chaining)	`get_db inst:<inst_name> .base_cell.name`
To get all placed sequential cells	`get_db [get_db insts -if {.is_sequential==true \ && .place_status==placed}] .name`
Another example to get the number of inverters and buffers	`llength [get_db insts -if {.is_buffer||.is_inverter}]`

Εντολές για την παραγωγή αναφορών

Command	Description
`report_area`	Prints an exhaustive hierarchical area report
`report_dp`	Prints a datapath resources report (to be done before syn_map)
`report_design_rules`	Prints design rule violations
`report_gates`	Reports libcells used, total area, and instance count summary
`report_hierarchy`	Prints a hierarchy report
`report_instance`	Generates a report on the specified instance
`report_memory`	Prints a memory usage report
`report_messages`	Prints a summary of the error messages that have been issued
`report_power`	Prints a power report
`report_qor`	Prints a quality-of-results report
`report_timing`	Prints a timing report
`report_summary`	Prints an area, timing, and design rules report

Συντομεύσεις πλήκτρων στη γραμμή εντολών

CTRL-a	Goes to the beginning of the line
CTRL-c	Stops the current Genus process and, if pressed twice, exits Genus
CTRL-e	Goes to the end of the line
CTRL-k	Deletes all text to the end of line
CTRL-n	Goes to the next command in the history
CTRL-p	Goes to the previous command in the history
CTRL-z	Instructs Genus to go to sleep
Up arrow	Displays the previous command in history
Down arrow	Displays the next command in history
Left arrow	Moves the cursor to the right
Right arrow	Moves the cursor to the left
BACKSPACE	Deletes the character to the left of the cursor
DELETE	Deletes the character to the right of the cursor