f

2020

Ψηφιακά Ολοκληρωμένα Κυκλώματα VLSI-ASIC Μεγάλης Κλίμακας

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

ΑΝ. ΚΑΘΗΓΗΤΗΣ ΒΑΣΙΛΗΣ ΠΑΥΛΙΔΗΣ

Περιεχόμενα

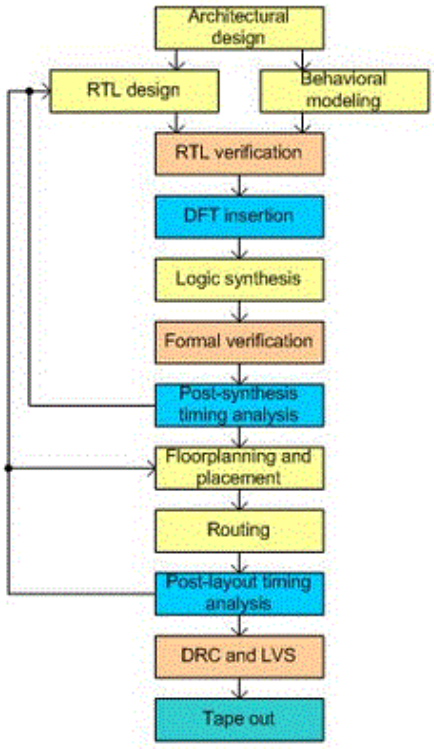
1. Εισαγωγή

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 v2020 και καταβάλλεται κάθε προσπάθεια να ενημερωνέται 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 ετερεία 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 προεραιτικά 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 (αυτήν 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 να υπάρχει 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.

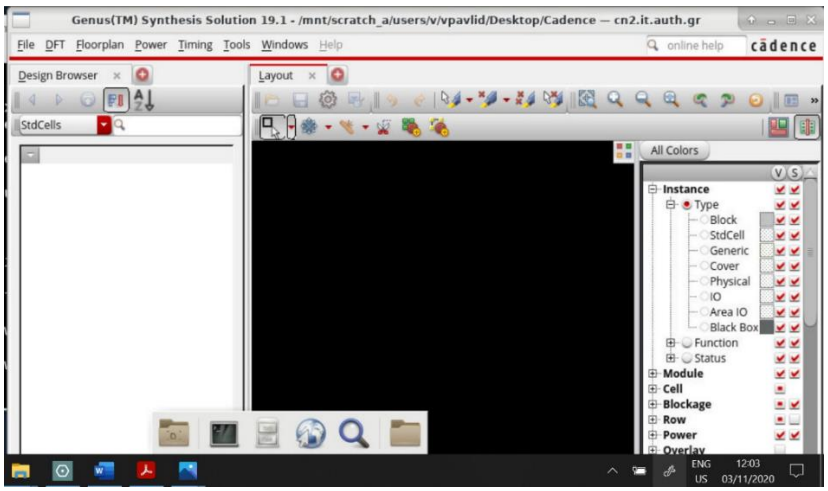
Το εργαλείο τυπώνει κάποια πληροφορία σχετικά π.χ. με τον τύπο άδειας που καλείται και την έκδοσή του κλπ και εν συνεχεία εμφανίζεται η γραμμή εργασίας:

```
@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 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:

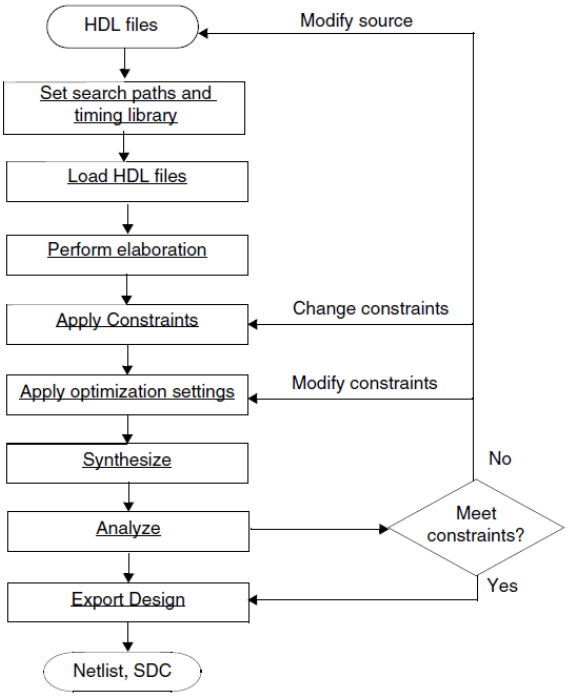
```
genus:root: 41> path_<Tab>
```

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 που 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 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:).

Έχοντας προσδιορίσει τη βιβλιοθήκη που θα χρησιμοποιηθεί, 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.

Ένα attribute μπορεί να τεθεί σε διάφορα επίπεδα της ιεραρχίας/σχεδίασης, δηλαδή να εφαρμοστεί σε όλα τα modules κυκλώματος ή σε συγκεκριμένα μόνο module ή object. Για να θέσετε ένα argument στο root (δηλ. κορυφή της ιεραρχίας) εκτελείτε:

set_db *attribute_name value* ή **set_db / .***attribute_name value*

Αν επιθυμείτε να δώσετε μία τιμή ενός attribute σε ένα άλλο object τότε πρέπει να καθορίσετε το object αυτό και τον τύπο του ως εξής:

set_db *object_type:name attribute_name value*

Παράδειγμα: **set_db** `design:top_mod .lp_clock_gating_exclude true`

Για να δείτε τί τιμή έχει ένα attribute για ένα συγκεκριμένο object τότε εκτελείτε:

get_db *object .attribute_name*

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 }
```

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:

- Φτιάχονται δομές δεδομένων
- Εισάγει τα ακολουθιακά στοιχεία του κυκλώματος
- Πραγματοποιεί βελτιστοποιήσεις του HDL σε υψηλό επίπεδο, όπως αφαίρεση κώδικα HDL που δε χρησιμοποιείται (dead code)
- Ελέγχει τους χρησιμοποιούμενους συμβολισμούς (semantics) στον HDL κώδικα

Σημείωση: 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.

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 οποιοδήποτε 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 το αν 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
```

Μετα 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 το 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 χρησιμποιώντας δύο συνεχόμενες φορές τη συντόμευση <Control-c>.

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). ΄Οταν 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
#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 > dessign.v
write_sdc > constraints.sdc
write_script > constraints.g

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

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
```

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