

Τελική Αναφορά Συστήματος

ΜΑΘΗΜΑ: ΕΡΓΑΣΤΗΡΙΟ ΣΧΕΔΙΑΣΗΣ SOC ΜΕ ΕΡΓΑΛΕΙΑ CAD
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ | ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Περιεχόμενα

Εισαγωγή.....	2
Διαμοιρασμός της εργασίας	3
Προσδιορισμός του συστήματος.....	6
Προσομοίωση συμπεριφοράς συστήματος	12
Λογική Σύνθεση	22
Εισαγωγή δομών DFT (Design For Test)	36
Φυσική Σχεδίαση	42
Συμπέρασμα	65

Εισαγωγή

Διαχειριστές του Project: Παναγιώτης Αναστασιάδης (2134), Αναστάσιος Μεσσής (2266)

Το παρόν έγγραφο αποτελεί τη τελική αναφορά του συστήματος. Στην τελική αναφορά εμπεριέχεται ολόκληρη η ροή της σχεδίασης που ακολουθήθηκε για το System on Chip της ομάδας και περιγράφονται τα στάδια υλοποίησης από την επιλογή συστήματος μέχρι και την φυσική σχεδίαση.

Από την 1^η τεχνική έκθεση μέχρι και την τελευταία σε αριθμό 5^η, έχει αναλυτικά διατυπωθεί η ροή της σχεδίασης του συστήματος, με τυχόν αναθεωρήσεις και αλλαγές στη δομή του, να τονίζονται εκεί που χρειάζεται. Συνεπώς, το σύστημα για να φτάσει στη τελική μορφή του με την οποία πέρασε ολοκληρωμένα όλα τα στάδια της σχεδίασης ενός SoC, έπρεπε να περάσει από πολλές εκδόσεις.

Έτσι στην τελική αυτή αναφορά του συστήματος περιγράφεται η νεότερη και τελευταία έκδοση του, η οποία είναι και αυτή που από το θεωρητικό πλαίσιο πήρε μορφή σε RTL και πέρασε από τη προσομοίωση συμπεριφοράς στη λογική σύνθεση, από εκεί στην εισαγωγή των δομών DFT και τέλος στη φυσική σχεδίαση της.

Συνοπτικά, στις σελίδες που ακολουθούν αναλύεται η ροή της σχεδίασης ως εξής:

Προσδιορισμός συστήματος: Κατά τον προσδιορισμό του συστήματος περιγράφεται εκτενώς η θεωρητική υπόσταση του chip, εν ολίγοις τι αποτελεί και ποιες είναι οι εφαρμογές για τις οποίες υλοποιείται. Επιπλέον τονίζεται η δομή του, οι μονάδες που το απαρτίζουν, οι προδιαγραφές αυτών καθώς και η λειτουργία του ως σύστημα.

Προσομοίωση συμπεριφοράς: Στο στάδιο της προσομοίωσης αφού οριστεί το σενάριο λειτουργίας με το οποίο θα επαληθευτεί η συμπεριφορά του συστήματος, στη συνέχεια μέσω κυματομορφών ελέγχεται αν το αποτέλεσμα είναι το επιθυμητό. Για το σενάριο λειτουργίας κατασκευάζεται ένα behavioral Testbench σε κώδικα Verilog το όποιο και εξετάζει τη συμπεριφορά της RTL.

Λογική σύνθεση: Στο κεφάλαιο αυτό, η RTL αφού δέχεται τις όποιες αλλαγές για να υφίσταται συνθέσιμη, στη συνέχεια μεταβιβάζεται στο επίπεδο πυλών. Στο στάδιο αυτό και σε μορφή Netlist πλέον, δέχεται στατική χρονική ανάλυση και εξετάζονται οι περιπτώσεις παραβίασης των φαινομένων setup time και hold time. Αφού γίνουν οι απαιτούμενες ενέργειες ώστε οι παραβιάσεις να είναι μηδενικές, το σύστημα περνά ξανά από προσομοίωση συμπεριφοράς.

Εισαγωγή δομών DFT: Τα ολοκληρωμένα συστήματα πρέπει να μπορούν να επαληθεύονται όταν κατασκευαστούν. Την δυνατότητα της επαλήθευσης αναλαμβάνει ο αλγόριθμος ATPG, ο οποίος για να εκτελεστεί πάνω σε ένα σύστημα πρέπει αυτό να ενσωματώνει στο εσωτερικό του τις δομές DFT (Design For Test). Στο στάδιο αυτό αναλύεται η ενσωμάτωση των δομών αυτών στη Netlist και γίνεται εφαρμογή του ATPG για την επαλήθευση τους.

Φυσική σχεδίαση: Τελευταίο στάδιο της ροής είναι η φυσική σχεδίαση. Στο κομμάτι της φυσικής σχεδίασης γίνεται χωροθέτηση του chip, γίνεται διασύνδεση των συνδέσεων ρεύματος, τοποθετούνται τα standard cells στο πυρήνα του σχεδίου καθώς και το δέντρο ρολογιού, ενώ πραγματοποιείται και το Nano Routing. Με το τελείωμα του σχεδίου, το σύστημα επιδέχεται τελική στατική χρονική ανάλυση και ανάλυση ισχύος και τέλος περνά από μια τελική προσομοίωση συμπεριφοράς.

Διαμοιρασμός της εργασίας

Η διαχείριση του Project που περιγράφεται στο σύνολο του από την παρούσα τελική αναφορά, έγινε από τα μέλη της ομάδας, Παναγιώτη Αναστασιάδη (2134) και Αναστάσιο Μεσσή (2266), στη διάρκεια του εαρινού εξαμήνου 2017-2018. Ο καταμερισμός του φόρτου εργασίας έγινε ανά τα στάδια που ορίζει η ροή της σχεδίασης ενός SoC και συγκεκριμένα ως εξής:

Προσδιορισμός συστήματος

Παναγιώτης Αναστασιάδης:

- Έρευνα και επιλογή μονάδων από το www.opencores.org και αντίστοιχα sites για την υλοποίηση του συστήματος που επιλέχθηκε.
- Γραφή κώδικα Verilog για τη δημιουργία μονάδων (βλέπε παρακάτω: Control Unit), αλλαγές στον κώδικα των έτοιμων μονάδων του opencores.org και σύνθεση των επιμέρους μονάδων για την δημιουργία ενιαίου συστήματος.
- Περιγραφή των μονάδων που επιλέχθηκαν και εκτενή ανάλυση του τρόπου συνδεσμολογίας τους.
- Ανάλυση της λειτουργίας του συστήματος σε βήματα.

Αναστάσιος Μεσσής:

- Έρευνα και διατύπωση των εφαρμογών που απευθύνεται το σύστημα που υλοποιείται.
- Έρευνα για την τεκμηρίωση της θεωρητική σκοπιάς του συστήματος, δηλαδή της αναζήτησης των ορισμών των κυρίων μονάδων (βλέπε PID,PWM Ορισμός) και έλεγχος του συστήματος για την επαλήθευση των ορισμών σε κυκλωματικό επίπεδο.
- Κατασκευή του διαγράμματος Μπλοκ (Block Diagram) του συστήματος,
- Αναζήτηση και διατύπωση των πλεονεκτημάτων και μειονεκτημάτων της δομής του συστήματος.

Προσομοίωση συμπεριφοράς

Παναγιώτης Αναστασιάδης:

- Κατασκευή σε Verilog του Behavioral Testbench που θα επαληθεύσει το σύστημα.
- Ανάλυση των σεναρίων λειτουργίας (2 σενάρια) , δηλαδή του τρόπου με τον οποίον επιχειρείται να επαληθευτεί το σύστημα μέσω του Testbench και των κυματομορφών.
- Εκτέλεση και περιγραφή της προσομοίωσης για το σενάριο πρώτο (βλέπε παρακάτω Σενάριο Λειτουργίας: για Set-Point = 70%).

Αναστάσιος Μεσσής:

- Εκτέλεση και περιγραφή της προσομοίωσης για το σενάριο δεύτερο (βλέπε παρακάτω Σενάριο Λειτουργίας: για Set-Point = 10%).
- Κατασκευή 2 διαγραμμάτων Excel για τα 2 σενάρια λειτουργίας.
- Ανάλυση της λειτουργίας Coverage % του κώδικα από το INCISIVE.

Λογική σύνθεση

Παναγιώτης Αναστασιάδης:

- Διόρθωση της RTL στα σημεία που απαιτήθηκε ώστε να γίνει συνθέσιμη.
- Κατασκευή και οργάνωση ενός script που περιέχει τις εντολές εκείνες που είναι απαραίτητες για το πέρας του σχεδίου από τη λογική σύνθεση και το Design Compiler (π.χ. εντολές για compile , report timing, write Netlist/SDF κ.α.).
- Έρευνα και ορισμός μέσα από τις κατάλληλες εντολές, των false – paths του συστήματος, τα drive και load του inverter που απαιτούνται να ενσωματωθούν στα IO του συστήματος.
- Εκτέλεση και εξέταση της προσομοίωσης συμπεριφοράς με το αρχείο SDF από την ανάλυση για το setup time.

Αναστάσιος Μεσσής:

- Φόρτωση των κατάλληλων βιβλιοθηκών στο σύστημα (καλύτερης ή χειρότερης περίπτωσης) ανάλογα με την ανάλυση που πραγματοποιείται, γεγονός το οποίο σημαίνει κατάλληλη διαμόρφωση του αρχείου synopsys_dc.setup που απαιτεί ο Design Compiler.
- Εξαγωγή εικόνων του σχηματικού του σχεδίου για την τεχνική έκθεση, ώστε να φανεί η μορφή του στο επίπεδο πυλών και εξαγωγή αποτελεσμάτων της εντολής report_area.
- Εξαγωγή ιστογραμμάτων από το Design Compiler ως προς το Endpoint Slack και το Net Capacitance.
- Εκτέλεση και εξέταση της προσομοίωσης συμπεριφοράς με το αρχείο SDF από την ανάλυση για το hold time.

Από κοινού:

- Στατική χρονική ανάλυση ως προς setup time και εξαγωγή αποτελεσμάτων.
- Στατική χρονική ανάλυση ως προς hold time και εξαγωγή αποτελεσμάτων.
- Κατάλληλη επιλογή των περιόδων των ρολογιών του συστήματος ώστε να μην υπάρχουν παραβιάσεις.

Εισαγωγή δομών DFT

Παναγιώτης Αναστασιάδης:

- Εκτέλεση της προσομοίωσης του αυτόματου Testbench που παράγεται από το TetraMAX και έλεγχος για τη σωστή εξέταση όλων των patterns, δηλαδή με 0 mismatches.

Αναστάσιος Μεσσής:

- Εξαγωγή νέων ιστογραμμάτων από το Design Compiler (Endpoint Slack, Net Capacitance) μετά την εισαγωγή των αλυσίδων σάρωσης, ώστε να γίνει σύγκριση με τα προηγούμενα.

Από κοινού:

- Ενσωμάτωση δομών DFT στο σύστημα μέσω του Design Compiler.
- Διοχέτευση του συστήματος στο TetraMAX με ενέργειες όπως εκτέλεση του αλγορίθμου ATPG και δημιουργία των patterns που απαιτεί η ροή.

Φυσική σχεδίαση

Παναγιώτης Αναστασιάδης:

- Διαμόρφωση της Netlist και του αρχείου SDC, με σκοπό να περιέχουν τα I/O Pads για τα οποία επιλέχθηκαν οι κατάλληλοι IO buffers από τη βιβλιοθήκη.
- Δημιουργία IO File με τα Power Pads.
- Εξαγωγή του αρχείου SAIF και εκτέλεση του Report Power στο Design Compiler για να διαπιστωθεί η συνολική ισχύς του συστήματος.
- Στατικές χρονικές αναλύσεις ανά τα στάδια της φυσικής σχεδίασης και τελική στατική χρονική ανάλυση.

Αναστάσιος Μεσσής:

- Προετοιμασία INNOVUS. Συγκεκριμένα επιλογή κατάλληλων αρχείων και ρυθμίσεων κατά το Import Design.
- Εισαγωγή IO Fillers.
- Εισαγωγή Core Fillers.
- Ανάλυση Ισχύος.

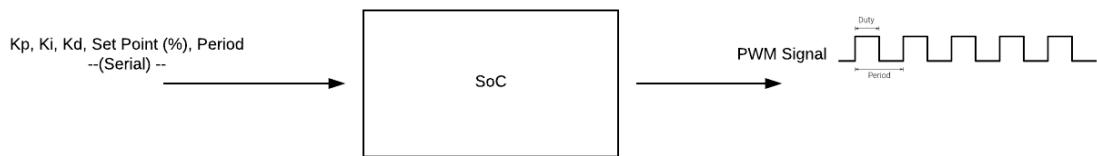
Από κοινού:

- Πραγματοποίηση χωροθέτησης, εισαγωγή διασυνδέσεων ρεύματος, Clock Tree Synthesis, Pre-CTS και Post-CTS Βελτιστοποίηση και Nano Routing.
- Επαληθεύσεις Verify Connectivity, Geometry, DRC, AC Limit.
- Τελική προσομοίωση συμπεριφοράς.

Προσδιορισμός του συστήματος

Τι είναι;

Το σύστημα SoC, στην τελική μορφή του που τέθηκε προς υλοποίηση, είναι ένα **επεξεργαστικό σχήμα ειδικής χρήσης**, το οποίο συνδέει έναν PID Ελεγκτή με μια γεννήτρια παραγωγής παλμών PWM (Pulse Width Modulation). Σε μια είσοδο **32-bit** δέχεται σειριακά ένα προς ένα τους συντελεστές K_p , K_i , K_d του PID Ελεγκτή, ένα επιθυμητό ποσοστό λειτουργίας για το duty cycle της γεννήτριας (π.χ. 20 %) και μια επιθυμητή περίοδο ορισμένη σε κύκλους ρολογιού. Σκοπός του συστήματος είναι, με βάση τη περίοδο που ορίζεται για τη γεννήτρια, να μετατρέψει το συγκεκριμένο ποσοστό στους κύκλους ρολογιού που αντιστοιχούν στο duty cycle της δοσμένης περιόδου και να βγάλει το προσαρμοσμένο παλμό στην έξοδο.



Εικόνα 1: Το σύστημα σε αφαιρετική μορφή.

Εφαρμογές που απευθύνεται:

Η χρήση τόσο του PID Controller όσο και των PWM παλμών συναντάται γενικά αρκετά συχνά στη βιομηχανία και την μηχανολογία. Το υλοποιημένο αυτό SoC δεν αποτελεί εξαίρεση και αποσκοπεί στη σύνδεση του με **κινητήρες DC**, όπου ο παλμός ελέγχει τις στροφές του, με **σύστημα ψύξης ή Θέρμανσης** με το σήμα να ελέγχει τη λειτουργία του ανεμιστήρα/θερμαντήρα, η και με ένα **απλό κύκλωμα σύνδεσης πηγής τάσης DC με μια λάμπα LED**, όπου ο παλμός ρυθμίζει την φωτεινότητα μεταξύ 0 και μέγιστης λειτουργίας.

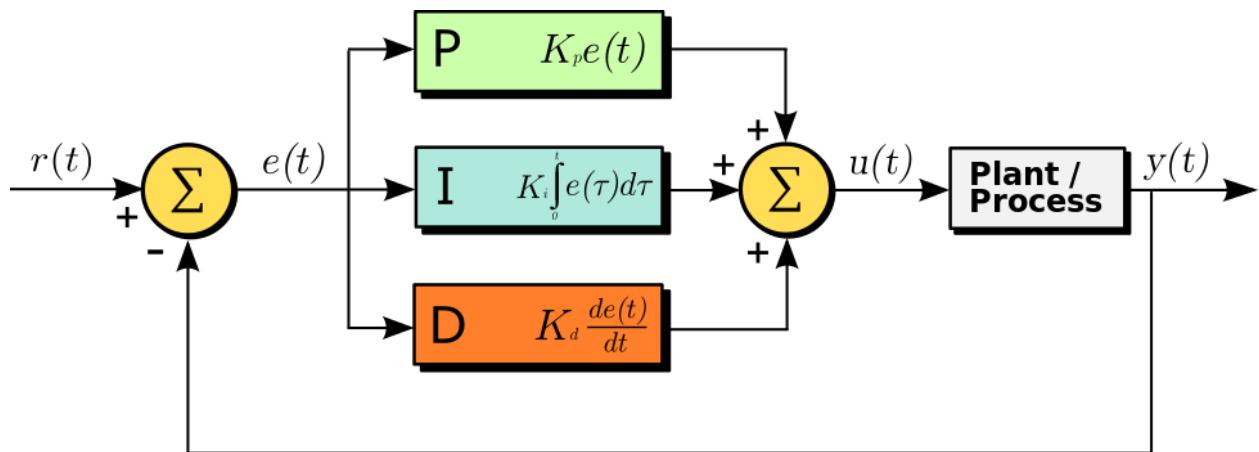
Στόχος είναι ο χρήστης να δίνει στο σύστημα το ποσοστό λειτουργίας που θέλει να δει στις διάφορες συσκευές που αναφέρθηκαν, και αυτό «μεταφράζοντας» το ψηφιακά σε κύκλους ρολογιού να το μεταβιβάζει στις συσκευές ως είσοδο.

**** Απαραίτητες θεωρητικές επισημάνσεις για βαθύτερη κατανόηση του συστήματος. ****

Τι είναι ένας PID Controller;

Ένας αναλογικός-ολοκληρωτικός-παραγωγικός ελεγκτής (ελεγκτής PID) είναι ένας γενικός μηχανισμός με ανατροφοδότηση βρόχων ελέγχου που χρησιμοποιείται ευρέως στα βιομηχανικά συστήματα ελέγχου. Ένας ελεγκτής PID προσπαθεί να διορθώσει το λάθος μεταξύ μιας μετρημένης μεταβλητής-διαδικασίας (Process Value) και ενός επιθυμητού σημείου λειτουργίας (setpoint) με τον υπολογισμό και έπειτα την έξοδο μιας διορθωτικής δράσης που μπορεί να ρυθμίσει τη διαδικασία αναλόγως.

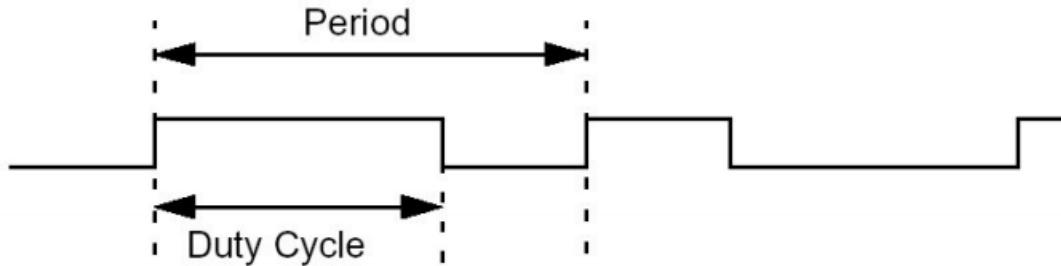
Ο υπολογισμός της εξόδου του ελεγκτή PID (αλγόριθμος) περιλαμβάνει τρεις ξεχωριστούς όρους. Τον αναλογικό, ολοκληρωτικό και παραγωγικό ορό. Το αναλογικό κέρδος καθορίζει την αντίδραση στο τρέχον λάθος, το ολοκλήρωμα καθορίζει την αντίδραση βασισμένη στο άθροισμα των λαθών και η παράγωγος καθορίζει την αντίδραση βάση του ποσοστού στο οποίο το λάθος έχει αλλάξει. Το σταθμισμένο ποσό αυτών των τριών ενεργειών χρησιμοποιείται για να ρυθμίσει τη διαδικασία μέσω ενός στοιχείου ελέγχου όπως η θέση μιας βαλβίδας ελέγχου ή η παροχή ηλεκτρικού ρεύματος ενός στοιχείου θέρμανσης, κινητήρα κτλ.



Εικόνα 2: Διάγραμμα συστήματος ενός PID ελεγκτή με ανάδραση.

PWM (Pulse Width Modulation) – Ορισμός:

Μία PWM κυματομορφή στην πραγματικότητα αποτελεί μία περιοδική κυματομορφή η οποία έχει δύο τμήματα. Το τμήμα ON στο οποίο η κυματομορφή έχει την μέγιστη τιμή της και το τμήμα OFF στο οποίο έχει την τιμή μηδέν. Το ON τμήμα ονομάζεται Duty Cycle και μετριέται είτε σε μονάδες χρόνου (ms, us κλπ) είτε σε ποσοστό (%) επί της περιόδου. Εφαρμόζοντας μία PWM κυματομορφή στην τροφοδοσία ενός φορτίου επιτυγχάνουμε να ελέγχουμε την το ποσοστό της ισχύος που πέφτει πάνω στο φορτίο. Για την περίπτωση που το φορτίο είναι ένας κινητήρας αυτό συνεπάγεται έλεγχος στροφών του κινητήρα.



Εικόνα 3: Αναπαράσταση ενός σήματος PWM.

**** Τέλος θεωρητικών επισημάνσεων****

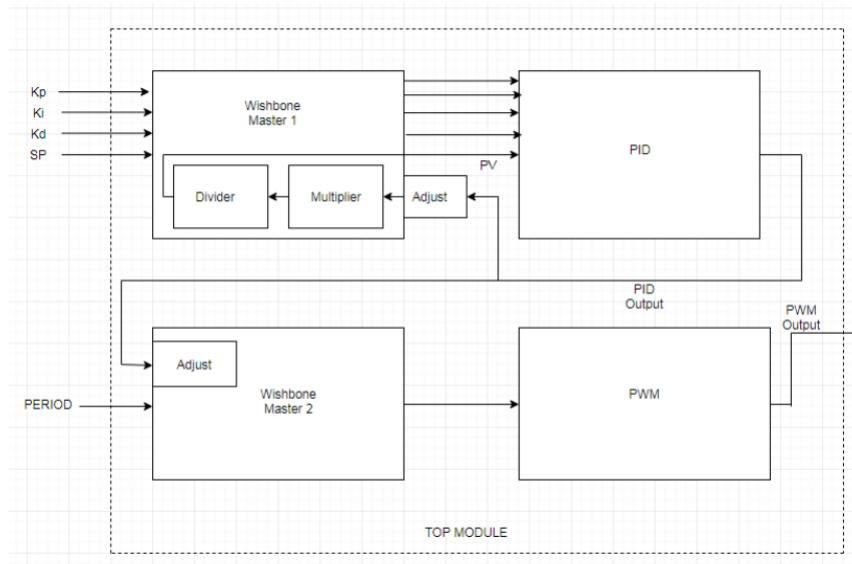
Συνοπτική λειτουργία του συστήματος σε βήματα:

- Θέτουμε τους συντελεστές K_p , K_i , K_d σε κάποια τιμή το καθένα (trial and error ή βάση κάποιας θεωρητικής μεθόδου π.χ. Ziegler - Nichols). Ανάλογα με το πόσο εύστοχη είναι η επιλογή των τιμών τόσο πιο γρήγορη θα είναι η σύγκλιση του αποτελέσματος.
- Θέτουμε τη περίοδο ίση με μια επιθυμητή τιμή P και την επιθυμητή κατάσταση Set – Point με μια τιμή ποσοστού S (%).
- Τα δεδομένα περνούν σειριακά στο σύστημα με τη σειρά K_p , K_i , K_d , Set – Point και Period.
- Ο PID υπολογίζει το σφάλμα μεταξύ set-point = S και process value = 0 (0 επειδή είναι η πρώτη φορά της επανάληψης).
- Η έξοδος του PID στέλνεται ως είσοδος του duty cycle στην PWM μονάδα και κατόπιν περνά πρώτα από έναν πολλαπλασιαστή (*100) και ύστερα από έναν διαιρέτη (δια τη περίοδο), ενώ τέλος επιστρέφει ως νέα process value στην είσοδο του PID.
- Αν αυτή η νέα τιμή process value είναι ίση με S σημαίνει ότι το σφάλμα είναι μηδενικό άρα το PID Output που είναι κατά επέκταση το duty cycle είναι σωστά ίσο με τον αριθμό των κύκλων ρολογιού που αντιστοιχούν στο ποσοστό S (%) της τιμής P της περιόδου.
- Αν όχι, η επανάληψη συνεχίζεται μέχρι το process value να γίνει ίσο με το set point και να φτάσουμε στο σημείο που περιγράφει το προηγούμενο βήμα.

Περιγραφή της δομής και των μονάδων του συστήματος:

1. **PID controller IP Core:** Έτοιμη μονάδα από τον ιστότοπο www.opencores.org, με πιστοποίηση για έτοιμο και «stable» κώδικα Verilog HDL. Διαθέτει Wishbone B4 Slave Interface και δέχτηκε κάποιες αλλαγές στο κώδικα, ώστε να μπορεί να συγχρονίζεται με τις υπόλοιπες μονάδες στην ανταλλαγή δεδομένων αλλά και στο «handshaking» με το Wishbone Master Control Unit.

2. **PWM IP Core:** Παρόμοια με το PID Unit, έτοιμη μονάδα από το www.opencores.org. Διέθετε πέρα από τις δυνατότητες παραγωγής PWM παλμού και δεύτερη λειτουργία που αφορούσε έναν μετρητή – Timer. Υπέστη και εδώ ο κώδικας διαφοροποιήσεις, όπως να απογυμνωθεί από τη λειτουργία του μετρητή καθώς κρίθηκε περιττή για το σύστημα, αλλά και αντίστοιχες αλλαγές με το PID, ως προς την συγχρονισμένη ανταλλαγή δεδομένων με τις υπόλοιπες μονάδες.
3. **Control unit:** Το control unit διαθέτει κώδικα που υλοποιήθηκε εξ ολοκλήρου από την αρχή και αποτελείται από 2 Wishbone Master μονάδες, μια να επικοινωνεί και να διαχειρίζεται τα δεδομένα του PID Core και μια να πράπτει τις αντίστοιχες λειτουργίες με το PWM.
4. **Top Module:** Αφορά τον κώδικα του συνολικού συστήματος το οποίο ουσιαστικά συνδέει τις υπόλοιπες μονάδες μεταξύ τους και γράφτηκε από την αρχή. Διαθέτει καταχωρητές έναν για κάθε είσοδο του συστήματος και αφού δεχτεί σειριακά όλες τις τιμές των εισόδων τις αποθηκεύει σε αυτούς και τις μεταβιβάζει στο Control Unit των Master Units.
5. **Multiplier:** Βρίσκεται μέσα στη μονάδα του Wishbone Master που επικοινωνεί του PID και αποτελεί έτοιμο πολλαπλασιαστή από τον ιστότοπο www.ellab.physics.upatras.gr (Online Arithmetic Module Generator Site).
6. **Divider:** Παρόμοια με τον πολλαπλασιαστή βρίσκεται στο Wishbone Master του PID Unit και αποτελεί έτοιμο σειριακό διαιρέτη από τον ιστότοπο www.fpga4student.com.



Εικόνα 4: Block Diagram του συστήματος. Παρόλο που η είσοδος είναι μια για τους συντελεστές, το Set – Point και τη περίοδο με τα δεδομένα να εισέρχονται σειριακά, στο σχήμα φαίνονται σαν 5 ξεχωριστές, ώστε να τονιστεί πως αυτές μοιράζονται στα Wishbone Master Units μέσα από το Topmodule unit.

Συνδεσμολογία της δομής και περιγραφή της ως σύστημα:

Η συνδεσμολογία του συστήματος φαίνεται αναλυτικά στο Block Diagram της εικόνας 4 και περιγράφεται ως εξής: Το topmodule παίρνει στην είσοδο το K_p , K_i , K_d , την περίοδο και την επιθυμητή κατάσταση-λειτουργία SP(Set Point) σειριακά. Στη συνέχεια τα αποθηκεύει σε καταχωρητές και τα μεταβιβάζει καταλλήλως στο Control Unit (K_p , K_i , K_d , Set-Point στο Wishbone Master του PID και τη περίοδο στο Wishbone Master του PWM). Τα 2 Master Units καταχωρούν τις τιμές αυτές στους καταχωρητές τους και σειριακά μέσω Hand -Shaking στην κάθε μετάδοση (Write-Acknowledgment), μεταβιβάζουν το πρώτο τους P,I,D συντελεστές και το Set-Point στον PID και το δεύτερο την περίοδο στον PWM. Αρχικά και μέχρι να υπολογιστεί το πρώτο Output του PID, το Process Value και το Duty Cycle παραμένουν στο μηδέν. Αυτό σημαίνει ότι ο PID σαν πρώτο σφάλμα υπολογίζει {Error = SP – 0}, ενώ το duty cycle του PWM και κατά επέκταση το ίδιο το σήμα που βγαίνει στην έξοδο του SoC είναι ίσα με το μηδέν. Με τον πρώτο υπολογισμό της εξόδου του ελεγκτή αλλά και σε κάθε νέο υπολογισμό, η τιμή αυτή επιστρέφει πίσω και στα 2 Master Units, όπου αυτός που διαχειρίζεται τις εισόδους της γεννήτριας PWM, προσαρμόζει εφόσον χρειαστεί την τιμή στο διάστημα μεταξύ 0 και περιόδου (βλέπε κουτί Adjust στο σχήμα) και το στέλνει ως duty cycle στη μονάδα. Ο άλλος Master εφόσον πραγματοποιήσει πρώτα την ίδια προσαρμογή – Adjust στην τιμή, κατόπιν τη μετατρέπει σε ποσοστό περνώντας την μέσα από τον Multiplier και τον Divider, αυτή ως νέα πια τιμή Process Value οδηγεί ξανά προς τον ελεγκτή. Η επανάληψη συνεχίζεται έως ότου το Process Value γίνει ίσο με το Set- Point και το σφάλμα μηδέν.

Αιτιολόγηση της δομής και ενναλακτική:

Με κύριο περιορισμό πρώτα τα χρονικά περιθώρια του μαθήματος και δεύτερον την διαχείριση έτοιμου κώδικα Verilog δεδομένης πολυπλοκότητας και μεγέθους, η δομή αυτή είναι η εφικτή δυνατή. Τα κομμάτια όπως το control unit και το topmodule ήταν απαραίτητο να γραφούν από την αρχή ώστε να μπορούν όλες οι επιμέρους μονάδες να επικοινωνούν ως σύστημα κάτω από το σκεπτικό του project και για αυτό η προσθήκη μεγαλύτερης πολυπλοκότητας στο σύστημα κρίθηκε αδύνατη. Ως προς πιθανές άλλες αλλαγές, ενναλακτική αποτελεί μόνο η πιθανότητα να βρεθεί γρηγορότερος Divider για να αντικαταστήσει τον νυν, καθώς ο υπάρχων καθυστερεί να υπολογίσει το αποτέλεσμα της διαίρεσης αν δουλεύει στο κοινό ρολόι των μονάδων και χάνει τους υπολογισμούς μερικών PID Outputs. Συμπερασματικά, ο divider από το www.fpga4student.com υλοποιήθηκε ώστε να δουλεύει κάτω από δικό του γρηγορότερο ρολόι από το κοινό του συστήματος.

Πλεονεκτήματα της δομής:

- Τα wishbone interfaces των μονάδων (Masters και Slaves) μειώνουν την πολυπλοκότητα στην επικοινωνία τους, κάνουν άμεση τη διοχέτευση των δεδομένων και συμβάλλουν ώστε το σύστημα να μπορεί να υποστηρίξει και άλλες μονάδες στο εσωτερικό του με παρόμοιο interface Wishbone.
- PID ελεγκτής υπολογίζει σε μόλις 15 κύκλους κάθε νέο αποτέλεσμα, οπότε το συνολικό σφάλμα μεταξύ SP και PV συγκλίνει στο μηδέν σε πολύ σύντομο χρονικό διάστημα.

- Τόσο ο 16 bit x 16 bit pipelined multiplier που βρίσκεται μέσα στο PID Unit όσο και ο multiplier που πάρθηκε από το www.ellab.physics.upatras.gr υπολογίζουν άμεσα το αποτέλεσμα έτσι δεν υπάρχει καθυστέρηση κατά την πράξη του πολλαπλασιασμού όπου αυτή χρειάζεται.
- Η απουσία μνήμης είναι ακόμα ένα θετικό καθώς όσες τιμές χρειάζεται να κρατηθούν χρησιμοποιούνται απλοί registers και τίποτα παραπάνω.

Μειονεκτήματα της νέας δομής:

- Ο αργός υπάρχων divider της δομής ο οποίος επιβάλλει τη χρήση ξεχωριστού ρολογιού, γρηγορότερο από το κοινό wishbone clock των υπόλοιπων μονάδων.
- Δεν υφίσταται μονάδα που να εκτελεί κάποιον αλγόριθμο καλής αυτόματης ρύθμισης των συντελεστών P,I και D, οπότε η ρύθμιση τους αφήνεται στην εμπειρία και τις ικανότητες του χρήστη.

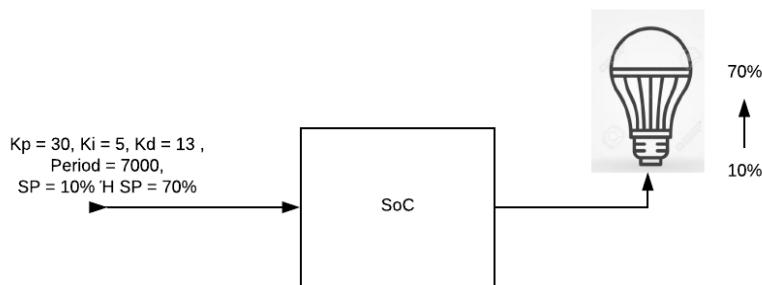
Προσομοίωση συμπεριφοράς συστήματος

Δημιουργία Behavioral Testbench:

Για την ανάγκη της προσομοίωσης της συμπεριφοράς του συστήματος υλοποιήθηκε ένα αρχείο Testbench σε κώδικα Verilog HDL το οποίο τροφοδοτεί τις εισόδους του chip με τα δεδομένα K_p, K_i, K_d , Set Point και Period και παρακολουθεί τον παλμό PWM στην έξοδο. Ένας counter σε κάθε ακμή του ρολογιού μετρά πόσους κύκλους είναι high (δηλαδή 1) το σήμα της εξόδου για να αποδείξει ότι το duty cycle είναι όντως ο αριθμός των κύκλων που αντιστοιχεί στο ποσοστό Set – Point που τοποθετήσαμε κατά την είσοδο.

Σενάριο λειτουργίας και τοποθέτησης των δεδομένων:

Ως σενάριο λειτουργίας θεωρούμε πως συνδέουμε το σύστημα μας με μια συσκευή τροφοδότησης ρεύματος μιας λάμπας LED, η οποία χρησιμοποιεί το σήμα της εξόδου του chip για να ορίσει τη φωτεινότητα-λειτουργία της λάμπας μεταξύ ποσοστού 0-100%. Εμείς επιλέγουμε να ορίσουμε τους συντελεστές $K_p = 30$, $K_i = 5$, $K_d = 13$ σαν το tuning του PID και τη περίοδο $period = 7000$. Επιθυμούμε να δούμε τη λάμπα να λειτουργεί μια φορά σε χαμηλή λειτουργία και μια φορά σε υψηλή οπότε θα δοκιμάσουμε ως $Set-Point = 10\%$ για τη πρώτη και $Set-Point = 70\%$ για τη δεύτερη.



Εικόνα 5: Αναπαράσταση σεναρίου λειτουργίας

Εργαλείο INCISIVE:

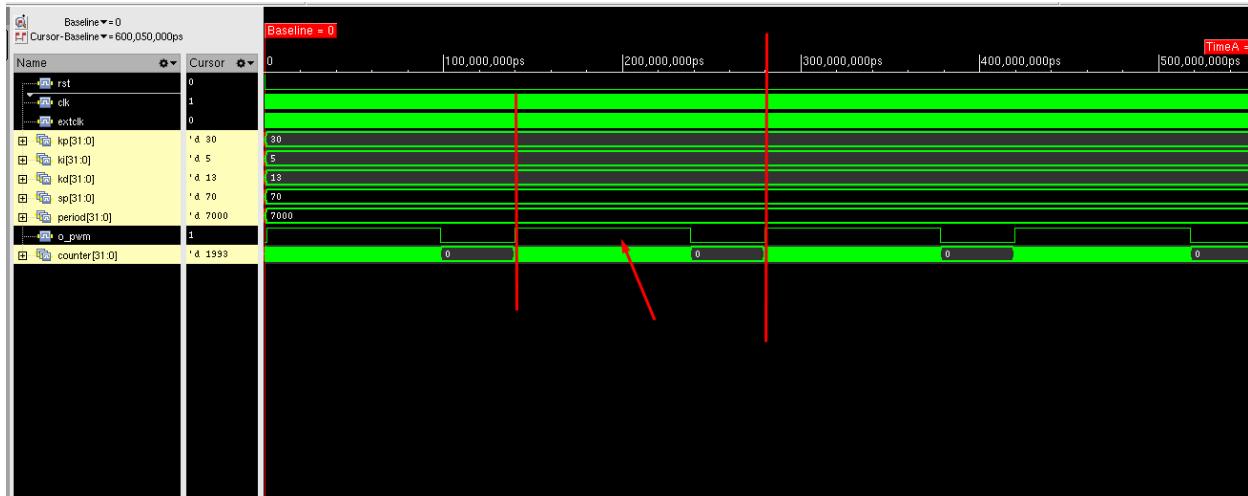
Για την παραγωγή κυματομορφών χρησιμοποιήθηκε το εργαλείο INCISIVE της Cadence. Όλα τα στάδια της ροής που απαιτούν προσομοίωση της συμπεριφοράς του συστήματος, έχουν πραγματοποιηθεί στο INCISIVE και όλες οι κυματομορφές των σημάτων προέρχονται από αυτό.

Στρατηγική επαλήθευσης:

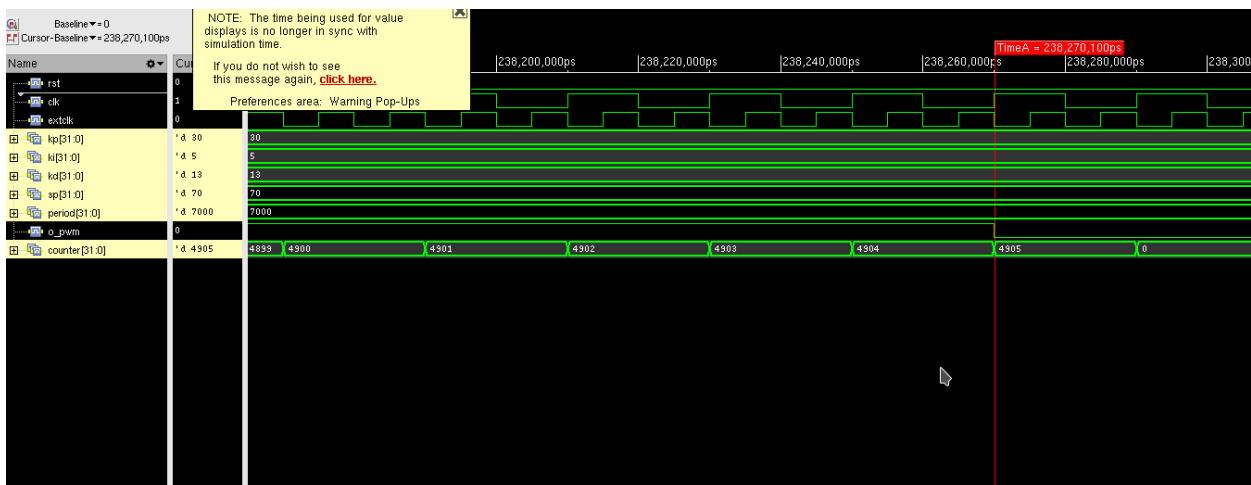
Θα ελεγχθεί η κάθε μονάδα του συστήματος ξεχωριστά με το top-level testbench αρχικά κάτω από το παραπάνω σενάριο λειτουργίας και στη συνέχεια οι υπόλοιπες μονάδες για να αποδειχθεί η ορθή συμπεριφορά και η ομαλή λειτουργία τους.

Κυματομορφές:

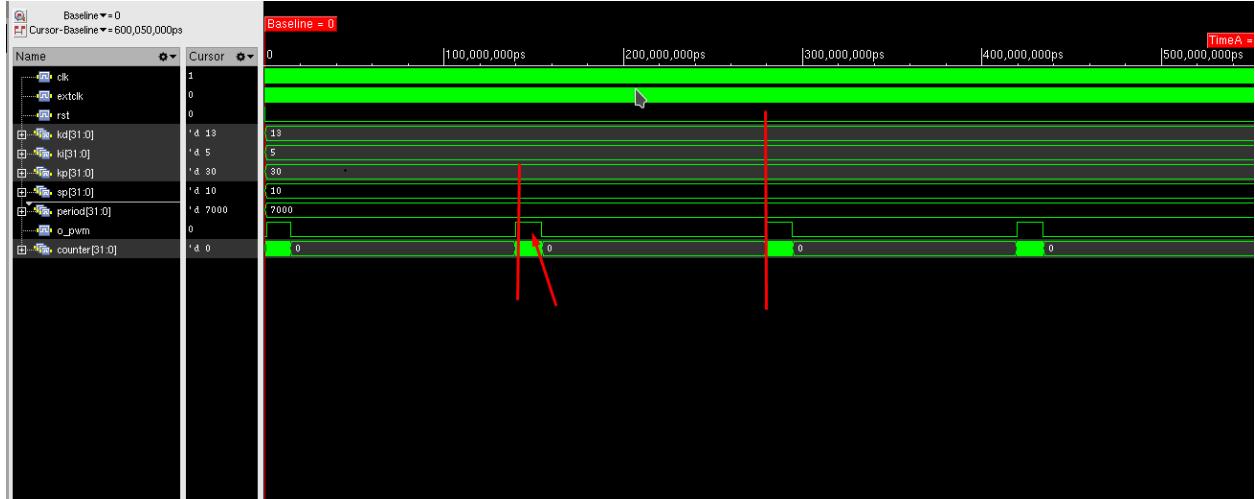
Top – level Testbench:



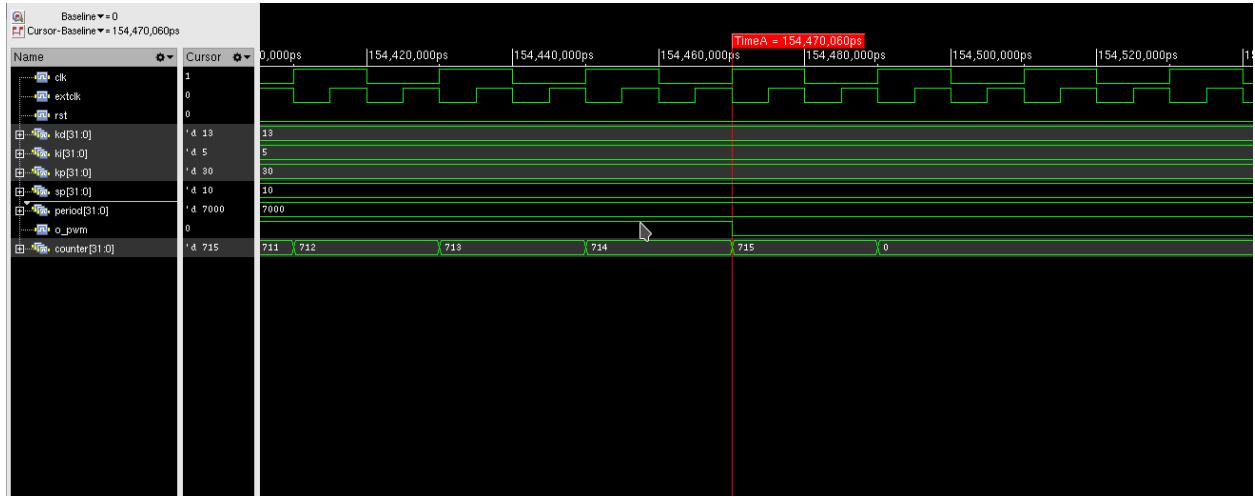
Κυματομορφή 1: Στη κυματομορφή αυτή διακρίνεται και αποδεικνύεται διαισθητικά πως με τις εισόδους που αναφέρθηκαν παραπάνω στο σενάριο συμπεριφοράς και ιδιαίτερα για Set–Point = 70 % το σήμα PWM στην έξοδο duty cycle στο 70% της περιόδου.



Κυματομορφή 2: Η δεύτερη εικόνα αποτελεί προιόν μεγάλης μεγέθυνσης της προηγούμενης και εστιάζεται στο σημείο όπου το duty cycle πέφτει από 1 στο 0 για να δειχθεί ότι ο counter έχει μετρήσει 4905 κύκλους «high» λειτουργίας του παλμού, δηλαδή ορθά το 70% της περιόδου ($4905 / 7000 = 0.7007$).

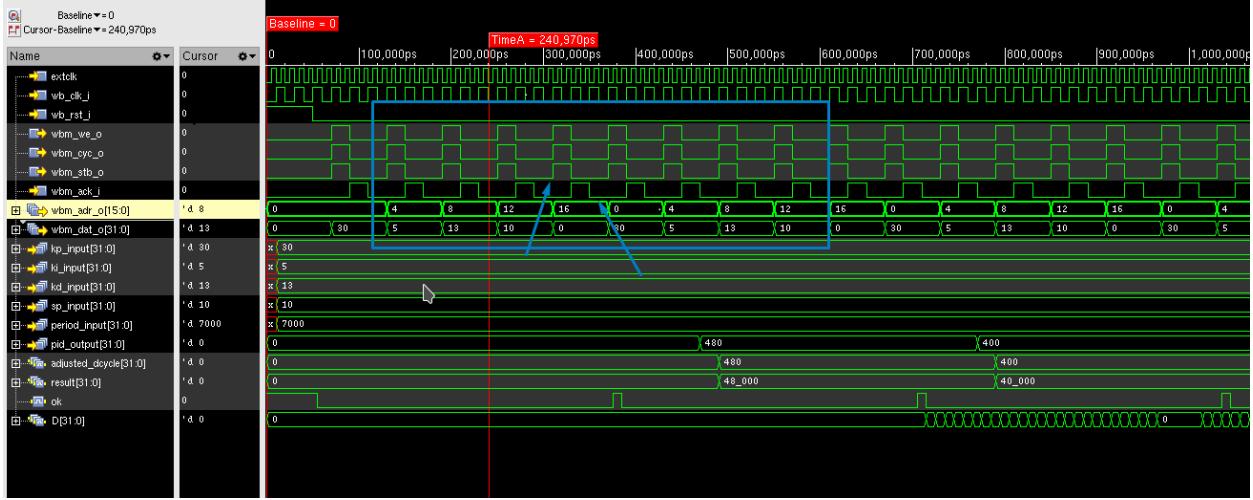


Κυματομορφή 3: Παρόμοια με τη κυματομορφή 1, για 10% λειτουργία της λάμπας ως επιθυμητή κατάσταση, διαισθητικά φαίνεται το αντιστοίχως σωστό PWM σήμα στην έξιδο.

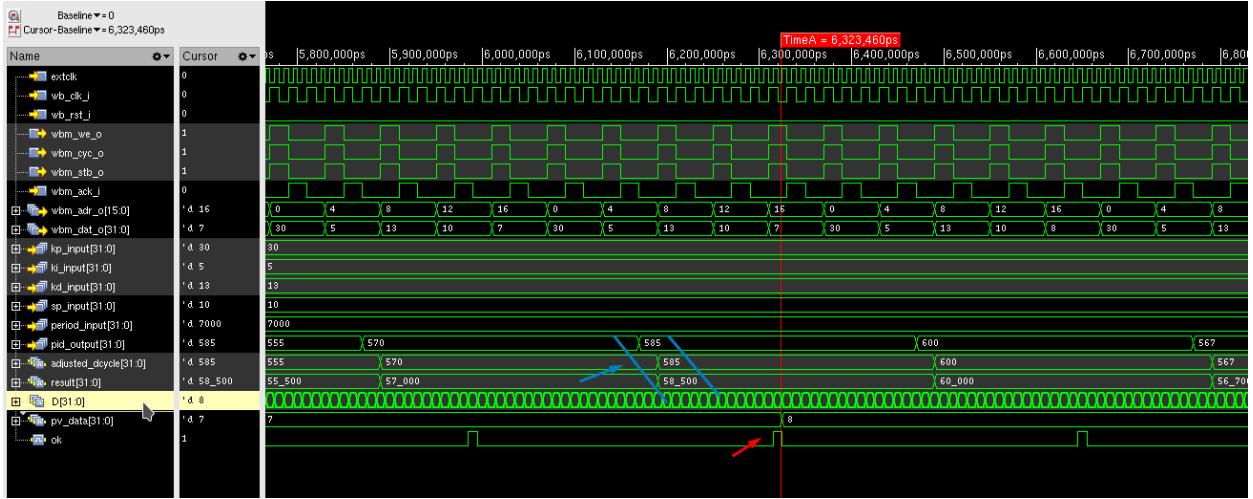


Κυματομορφή 4: Ομοίως με τη κυματομορφή 2 ο counter έχει μετρήσει 715 κύκλους high λειτουργίας του PWM που αντιστοιχούν στο 10% της περιόδου ($715 / 7000 = 0.102$).

Control Unit – Ιος Wishbone Master (το unit για διαχείριση δεδομένων του PID). -----

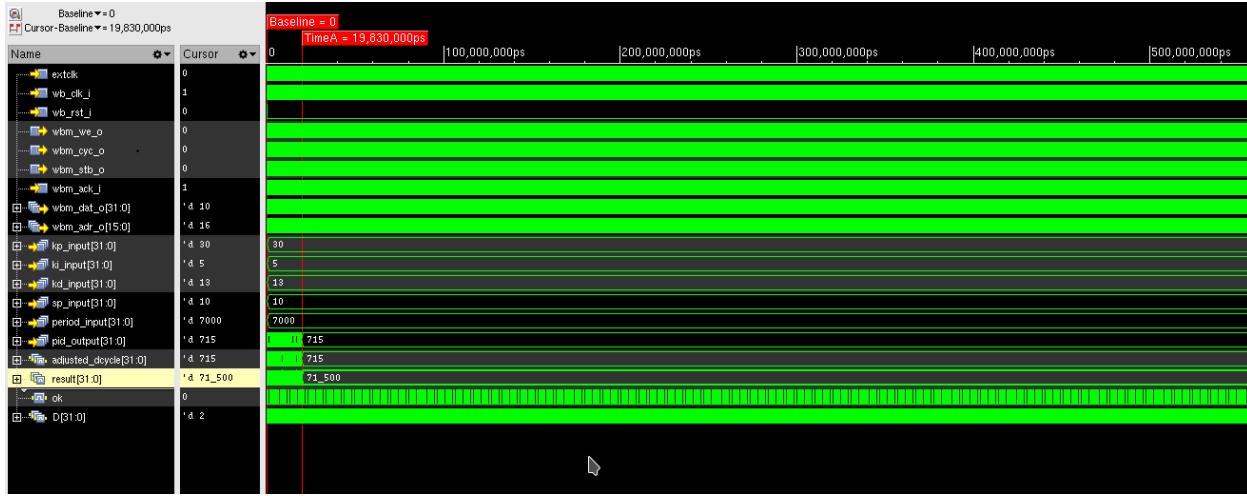


Κυματομορφή 5: Στη κυματομορφή αυτή, μέσα στη μονάδα του Master Unit που στέλνει τα δεδομένα στον PID, επιδεικνύεται ο τρόπος επικοινωνίας μεταξύ τους στο γαλάζιο πλαίσιο. Τα 3 σήματα we, cyc, stb αποτελούν το write enable κομμάτι που στέλνεται στον PID και ακριβώς από κάτω φαίνονται τα δεδομένα wbm_data_o μαζί με τη διεύθυνση wbm_adr_o που αντιστοιχεί σε κάθε ένα από αυτά. Μετά από κάθε ορθή μεταφορά στέλνεται το σήμα wbm_ack_i από τη μονάδα στο PID και δηλώνει ότι τα έλαβε επιτυχώς.



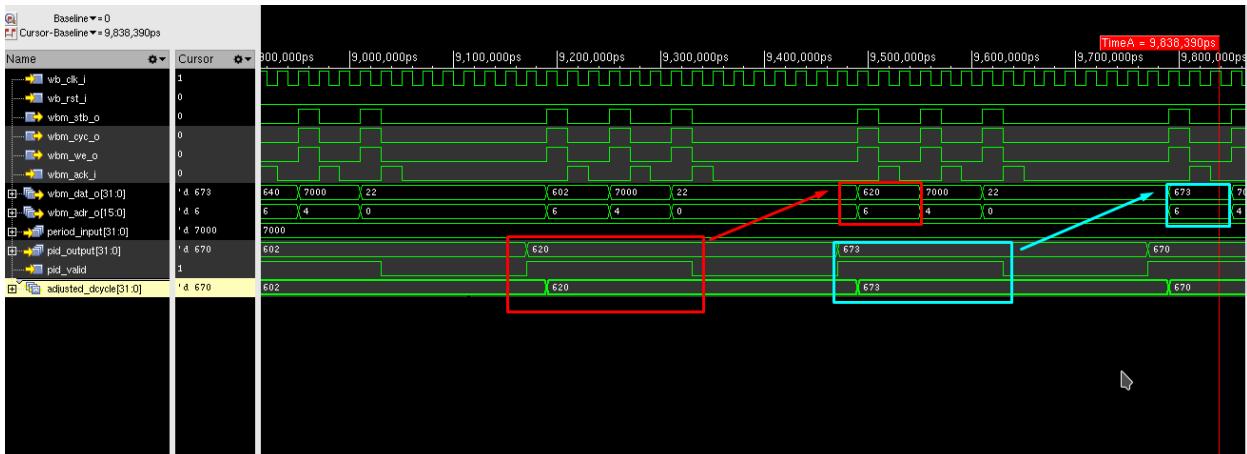
Κυματομορφή 6: Εδώ στη κυματομορφή αποδεικνύεται το γεγονός πως το κάθε PID Output που έρχεται πίσω στον Master προσαρμόζεται μεταξύ 0 και περιόδου (στη προκειμένη περίπτωση είναι μέσα στα όρια εξ αρχής αφού pid output = 585 και περίοδος = 7000). Στη συνέχεια πολλαπλασιάζεται *100 (βλέπε result) και διαιρείται με τη περίοδο (το αποτέλεσμα του divider είναι το σήμα D στην εικόνα και η valid τιμή του είναι αυτή που αντιστοιχεί στο ok όταν αυτό έχει την τιμή 1). Η διαιρεμένη

τιμή ($585 * 100\% / 7000 = 8\%$) μπαίνει τελικώς στον pv_data register και στέλνεται στον PID ως νέο process value.



Κυματομορφή 7: Στην τελευταία κυματομορφή για τον master unit του PID που αποτελεί πλήρη μεγέθυνση **30000** κύκλων προσομοίωσης, μπορούμε να διακρίνουμε ότι το τελικό αποτέλεσμα του υπολογισμού των 715 κύκλων για το σενάριο της 10% λειτουργίας, επιτυγχάνεται μόλις στους **1000** πρώτους (19,830,000 ps η στιγμή που υπολογίστηκε, δια 20 ns που είναι η περίοδος του ρολογιού αντιστοιχεί σε 991 κύκλους).

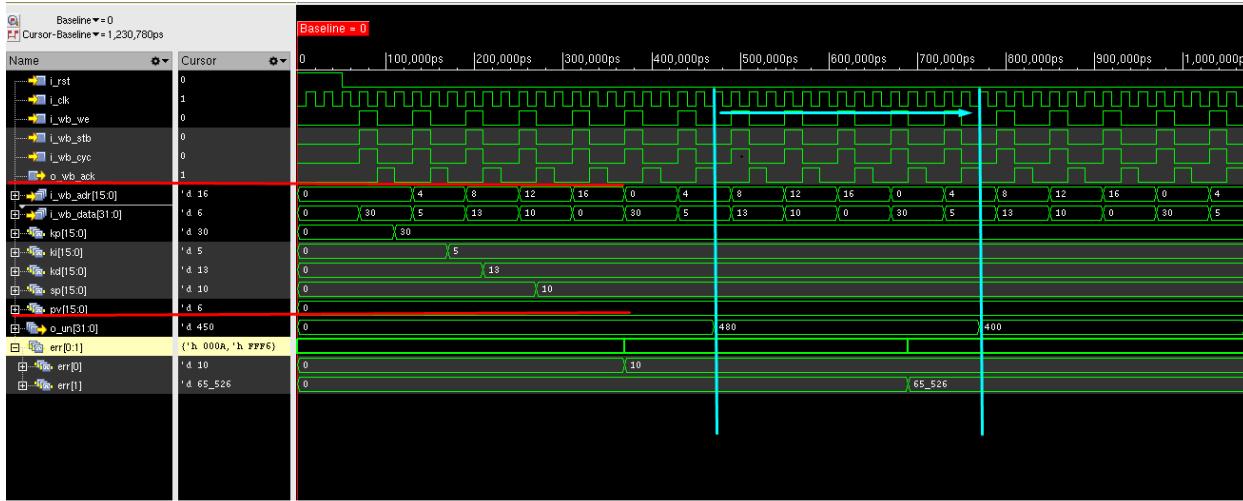
Control Unit – 2ος Wishbone Master (το unit για διαχείριση δεδομένων του PWM). -----



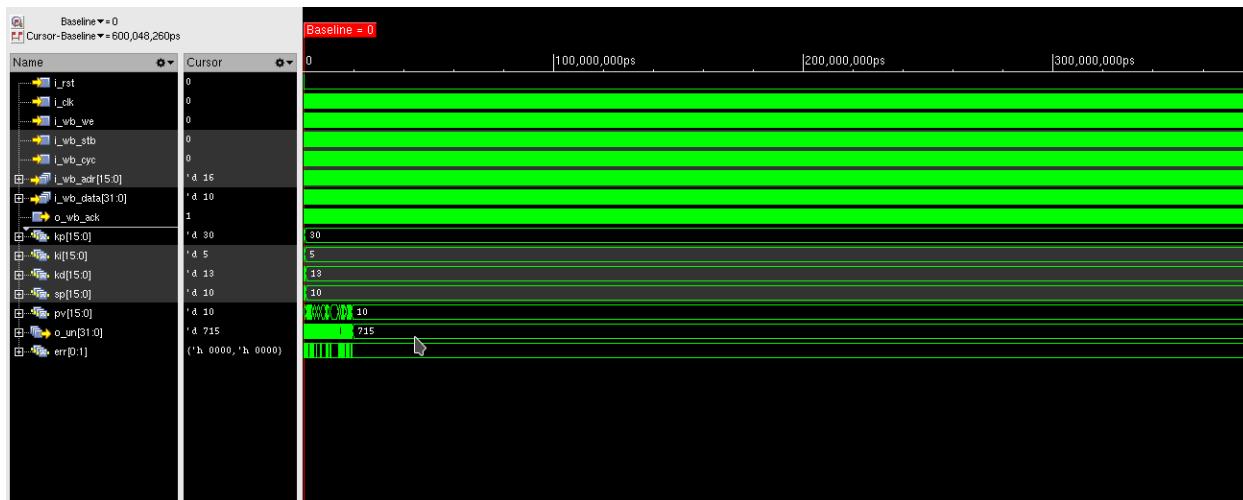
Κυματομορφή 8: Ο δεύτερος Master Unit που αφορά τη μετακίνηση δεδομένων στον PWM έχει πανομοιότυπη υλοποίηση με τον προηγούμενο με το handshaking και τα περισσότερα σήματα να είναι τα ίδια. Το σημείο που πρέπει να επισημανθεί στη δική του λειτουργία είναι εκείνο που το output του

PID προσαρμόζεται στο διάστημα 0 και περίοδο(υπενθυμίζουμε ότι η περίοδος στη προσομοίωση τοποθετήθηκε ίση με 7000) και στη συνέχεια διοχετεύεται στην έξοδο που οδηγεί στη μονάδα PWM ως wbm_data_0 με τη κατάλληλη διεύθυνση wbm_adr_o.

PID Unit. -

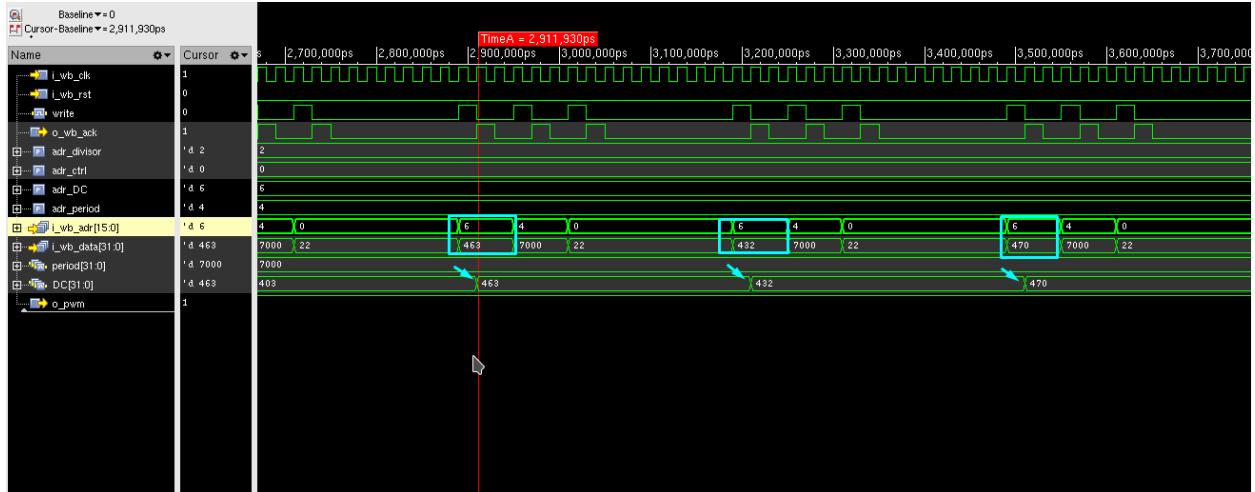


Κυματομορφή 9: Για τη λειτουργία του PID, στην πρώτη αυτή κυματομορφή παρουσιάζεται η πορεία της συμπεριφοράς του κατά τους πρώτους κύκλους της προσομοίωσης, όπου ανάμεσα από τις κόκκινες γραμμές διακρίνεται πως τα δεδομένα με την ανάλογη διεύθυνση εισέρχονται στη μονάδα (βλέπε `i_wb_adr` και `i_wb_data`) και στη σύνεχεια σειριακά αποθηκεύονται στους registers. Με τις γαλάζιες γραμμές αποδεικνύεται αυτό που ειπώθηκε παραπάνω, ότι από τον υπολογισμό του `o_un`(PID Output) στον επόμενο, μεσολαβούν 15 κύκλοι.

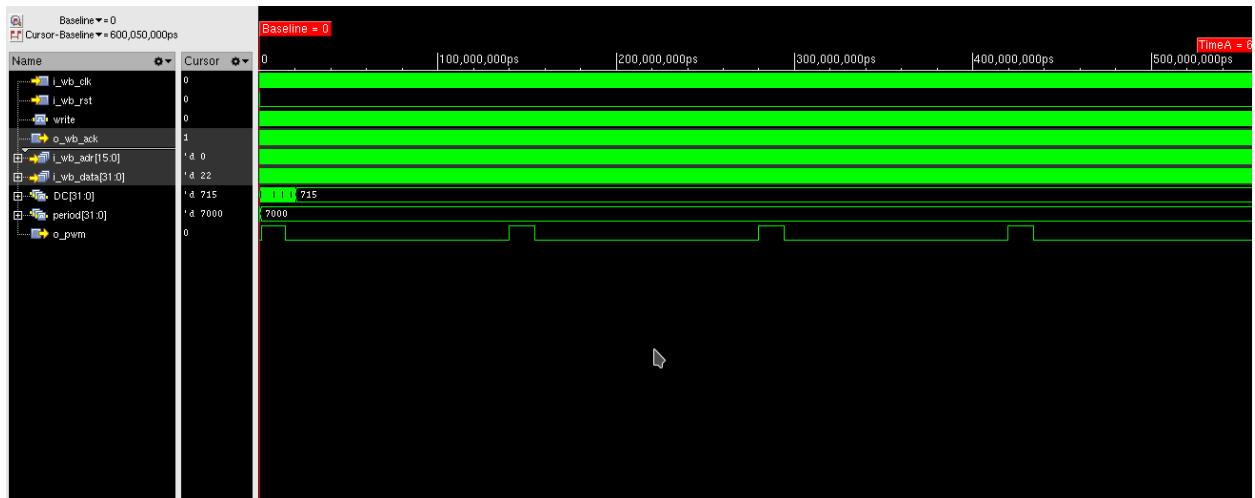


Κυματομορφή 10: Για τη προσομοίωση με set-point = 10 φαίνεται και εδώ στη μονάδα του PID ελεγκτή ότι το αποτέλεσμα που υπολογίζει είναι το σωστό που ειπώθηκε και παραπάνω, δηλαδή $o_un = 715$ κύκλοι.

PWM Unit.



Κυματομορφή 11: Για τη μονάδα του PWM παρουσιάζεται εδώ πως κάθε εισερχόμενη τιμή του προσαρμοσμένου PID Output εισχωρεί στον DC register που αφορά το duty cycle του σήματος.



Κυματομορφή 12: Αποδεικνύεται και εδώ σε πλήρη μεγέθυνση πως για τη προσομοίωση με επιθυμητή κατάσταση το 10%, το duty cycle στην μονάδα του PWM φτάνει ορθώς στους 715 κύκλους, ενώ ο παλμός `o_pwm` στην έξοδο φαίνεται διαισθητικά πως σε κάθε περίοδο είναι για το 10% ίσος με 1 και για το υπόλοιπο 90% ίσος με 0.

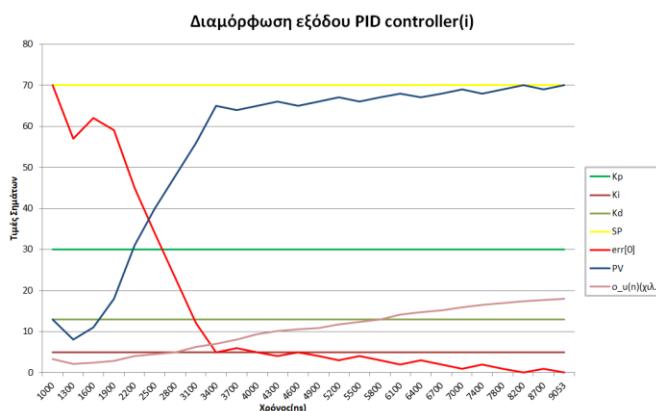
Topmodule.



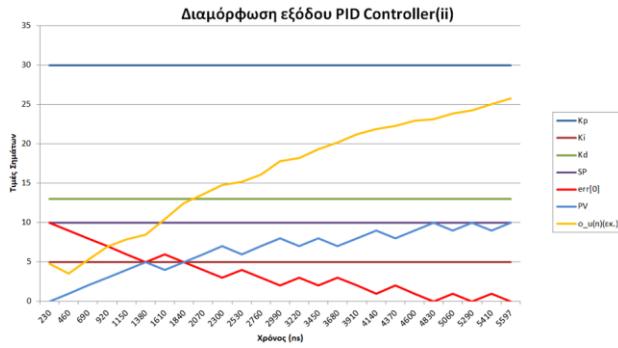
Κυματομορφή 13: Η τελευταία κυματομορφή αφορά το topmodule, το οποίο ρόλος είναι η σειριακή είσοδος των δεδομένων, οι καταχωρίσεις αυτών σε registers και κατόπιν οι διοχέτευση τους στις μονάδες των Master Units. Επιπρόσθετα οι διασυνδέσεις με καλωδιώσεις των επιμέρους μονάδων μεταξύ τους είναι ένα ακόμη χαρακτηριστικό τους. Στα αριστερά διακρίνονται τα καλώδια με πρόθεμα m2s από τους masters στους slaves και τα καλώδια s2m που ακολουθούν την αντίθετη διαδρομή. Επίσης φαίνονται τα σήματα εισόδου του chip, το σήμα o_pwm της εξόδου, τα 2 ρολόγια (το extclk αφορά το ρολόι του divider που προαναφέρθηκε) του συστήματος και το σήμα reset.

Διαγράμματα Excel:

Παρακάτω εμφανίζονται δυο σε αριθμό διαγράμματα Excel εκ των οποίων το πρώτο αφορά τη λειτουργία για επιθυμητό SP = 70 % και η άλλη για το SP = 10 %. Στα διαγράμματα φαίνεται η διέγερση και η συμπεριφορά των μεταβλητών Process Value (PV) και PID Output (o_un) σε σχέση με τη πάροδο του χρόνου.



Εικόνα 6: Διάγραμμα Excel των τιμών για το σενάριο της 70% λειτουργίας του duty cycle.



Εικόνα 7: Διάγραμμα Excel των τιμών για το σενάριο της 10% λειτουργίας του duty cycle.

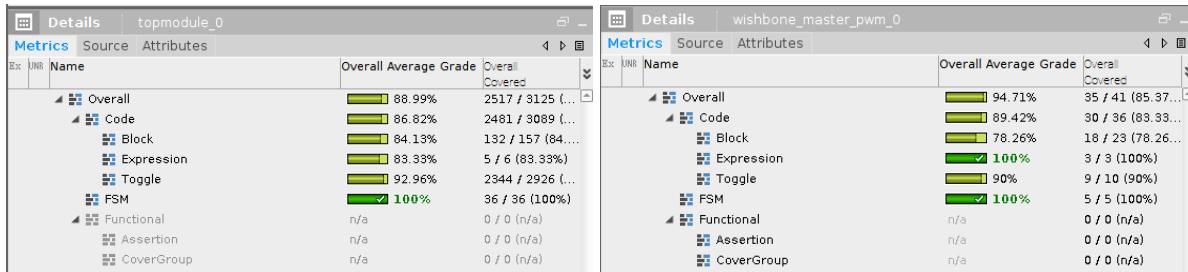
Κάλυψη % των μονάδων (Coverage):

Για να παρατηρήσουμε τη κάλυψη του κώδικα του συστήματος ενεργοποιήσαμε τη Coverage δυνατότητα για τη προσομοίωση από το INCISIVE. Με τη λειτουργία αυτή ελέγχεται κατά τι ποσοστό το behavioral testbench ελέγχει όλο τον κώδικα της RTL που έχει υλοποιηθεί. Στη περίπτωση μας η κάλυψη φτάνει στο **89.55%** του συνόλου των ελέγχων που πραγματοποιήθηκαν, δηλαδή στο σύνολο των Block, Expression, Toggle και FSM Coverages.

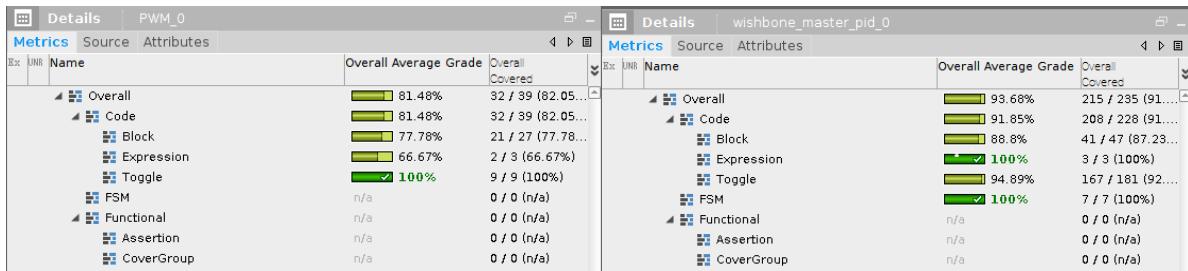
Ex	UNR	Name	Overall Average Grade	Overall Covered	Assertion Status Grade
		(no filter)	(no filter)	(no filter)	(no filter)
		Verification Metrics			
		Types	89.55%	4055 / 6312 (64.24%)	n/a
		Instances	86%	1507 / 3155 (47.77%)	n/a
		topmodule_tb	93.11%	2548 / 3157 (80.71%)	n/a
		topmodule_0	93.11%	2548 / 3125 (80.71%)	n/a
		wishbone_master_pid_0	88.99%	2517 / 3125 (80.54%)	n/a
		PIDO	93.68%	215 / 235 (91.49%)	n/a
		wishbone_master_pwm_0	86.09%	2235 / 2810 (79.54%)	n/a
		PWM_0	94.71%	35 / 41 (85.37%)	n/a
			81.48%	32 / 39 (82.05%)	n/a

Εικόνα 8: Αναλυτικά τα ποσοστά κάλυψης των επιμέρους μονάδων του συστήματος.

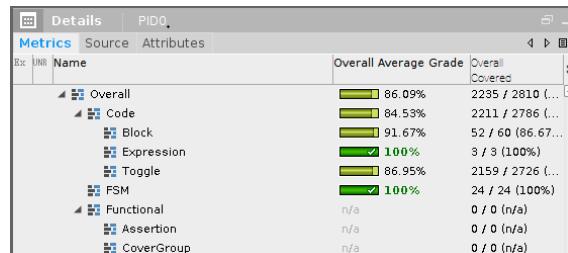
Ακολουθούν η αναλυτική % κάλυψη σε Block, Expression, Toggle και FSM Coverage όλων των μονάδων:



Εικόνα 9: Αναλυτική κάλυψη του topmodule (αριστερά) και του Master Unit για το PWM(δεξιά).



Εικόνα 10: Αναλυτική κάλυψη του PWM Unit (αριστερά) και του Master Unit για το PID(δεξιά).



Εικόνα 11: Αναλυτική κάλυψη του PID Core.

Λογική Σύνθεση

Σκοπός:

Μετά και τη προσομοίωση συμπεριφοράς της RTL, το επόμενο βήμα στη ροή αφορά τη μεταβίβαση του σχεδίου στο στάδιο της λογικής σύνθεσης. Στο σημείο αυτό το σχέδιο εισήχθη στο εργαλείο **Design Compiler** της **Synopsys** με σκοπό ο κώδικας Verilog της RTL να μεταβιβαστεί στο επίπεδο λογικών πυλών. Το κομμάτι των πυλών στηρίζεται και υλοποιείται πάνω στη βιβλιοθήκη της **Motorola** στα **90nm**. Στο στάδιο της λογικής σύνθεσης πραγματοποιήθηκαν συνοπτικά οι εξής ενέργειες:

- Εξήχθη το αρχείο της Netlist με το σχέδιο σε επίπεδο πυλών.
- Πραγματοποιήθηκε στατική χρονική ανάλυση για την εξέταση φαινόμενων setup και hold time.
- Πραγματοποιήθηκε εξαγωγή των αρχείων SDF με τις χρονικές καθυστερήσεις για να ενσωματωθούν στη δεύτερη προσομοίωσή συμπεριφοράς στο INCISIVE.
- Έγινε η προσομοίωση συμπεριφοράς του συστήματος σε επίπεδο πυλών στο INCISIVE.

Συνθέσιμη Verilog:

Σε πρώτη φάση προτού ασχοληθούμε με τις βιβλιοθήκες που πρέπει να φορτωθούν στον compiler, χρησιμοποιήθηκε το εργαλείο αρχικά και μόνο για να εξετασθεί αν ο κώδικας Verilog HDL της δομής είναι συνθέσιμος. Κατά το διάβασμα των αρχείων του συστήματος, βρέθηκαν μερικά απλά σφάλματα στο σύνολο της δομής που το καθιστούσαν μη συνθέσιμο και αυτά αφορούσαν κυρίως παραβιάσεις όπως ανάθεση τιμών σε registers κατά τη δήλωση και χρησιμοποίηση της μεταβλητής του ρολογιού μέσα σε always block και μέσα συνθήκη τύπου if else (π.χ. else (clk && ack) begin end). Τα λάθη αυτά διορθώθηκαν και ο κώδικας έγινε συνθέσιμος.

Φόρτωση Βιβλιοθηκών για την εξέταση παραβίασης Setup - Time:

Προτού εκτελεστεί ο design compiler της Synopsys για το συνθέσιμο κώδικα, πρώτα υλοποιείται το script των εντολών για τη φόρτωση των βιβλιοθηκών στη χειρότερη περίπτωση, καθώς θα εξετασθεί το φαινόμενο των πιθανών παραβιάσεων setup.

Setup Time: Ορίζεται η χρονική διάρκεια πριν το χρονικό σημείο μετάβασης ενός ακμοπυροδότητου *Flip-Flop*, όπου όλα τα δεδομένα της εισόδου πρέπει να παραμείνουν σταθερά στις λογικές τιμές τους για να πραγματοποιηθεί σωστά η μετάβαση του *Flip-Flop*.

Γιατί χειρότερη περίπτωση: Φορτώνοντας τη χειρότερη περίπτωση στην βιβλιοθήκη τεχνολογίας της Motorola (Χαμηλή τάση τροφοδοσίας, υψηλή θερμοκρασία κ.τ.λ.) εξετάζουμε το σύστημα με τις χειρότερες δυνατές καθυστερήσεις, ώστε όσα φαινόμενα παραβίασης setup-time υπάρχουν να εμφανιστούν και να προσδιορίσουν κατά πόσο πρέπει να αυξηθεί η περίοδος του ρολογιού για να υφίσταται η εξάλειψη τους.

```

#
# Written by : DC-Transcript, Version 2003.06 -- May 20, 2003
# Date       : Mon Aug  4 19:23:48 2003
#



set company {IHP Frankfurt (Oder) GmbH}
set designer "Panagiotis Anastasiadis"

set cache_read /tmp
set cache_write /tmp

set text_print_command {lpr -P }

set echo_include_commands false

set view_script_submenu_items [concat $view_script_submenu_items [list {remove all designs} {remove_design -designs}]]


set search_path [list "/home/faculty/chsotiriou/IHP-Lib/IHP-0.25um/digital/synopsys"]

set target_library sgc25a.max.db
set symbol_library sgc25a.sdb
set link_library [concat "*" $target_library]

# /* add I/O library: */
set target_library [concat $target_library sgc25a_io.max.db]
set symbol_library [format "%s" $symbol_library sgc25a_io.sdb]
set link_library [concat "*" $target_library]

```

Εικόνα 12: Screenshot από το script “synopsys_dc.setup” για την φόρτωση των max βιβλιοθηκών (χειρότερη περίπτωση) που απαιτεί ο Design Compiler.

Φόρτωση του σχεδίου – project στον Design Compiler:

Για να φορτωθεί το σύστημα στον compiler γράφτηκε ακόμα ένα script, κατά το οποίο εκτελούνται κατά σειρά οι εντολές **read** , **elaborate** των αρχείων της RTL του σχεδίου, ορίζεται η περίοδος του ρολογιού, τα **false-paths** , τα **drive** και **load** στις εισόδους και την έξοδο του συστήματος αντίστοιχα, το **compile** και το **report timing** για το **setup** αλλά και η εισαγωγή των αρχείων **netlist**, **SDF**. Ενδιάμεσα βρίσκεται και η ροή εντολών για την εισαγωγή του DFT στη δομή, το οποίο αναλύεται εκτενώς και αναφέρεται λίγες σελίδες παρακάτω.

```

#####
##### read RTL and elaborate topmodule
analyze -library WORK -format verilog {/home/inf2015/paanastasiadis/soc7/boot.v /home/inf2015/paanastasiadis/
inf2015/paanastasiadis/soc7/dok_pwm.v /home/inf2015/paanastasiadis/soc7/eudoxus_multiplier.v /home/inf2015/paa
paanastasiadis/soc7/wb_master_pwm.v}

elaborate topmodule -architecture verilog -library WORK

#####
##### clocks, false path , drive & load
create_clock clk -period 50
create_clock extclk -period 25
set_false_path -from rst
set_drive 3.1117 [all_inputs]
set_load 0.006109 [all_outputs]

#####
##### create_test_protocol for dft
create_test_protocol -infer_asynch -infer_clock

#####
##### compile for
compile -map_effort high -scan
check_design
report_timing

#####
##### compile for hold-time analysis
#compile - map_effort high -scan -only_hold_time
#report_timing -delay_type

#####
##### dft_drc + scan chain insertion + dft coverage + report_timing
dft_drc
insert_dft
dft_drc -coverage estimate
compile -incremental_mapping
report_timing

#####
##### files needed for TetraMAX and Incisive
write_h -f verilog -output netlist.v
write_sdf setup.sdf
#write_sdf hold.sdf

#####
##### file needed for TetraMAX
change_names -hierarchy -rule verilog
write_test_protocol -output scan_protocol.stil

```

Εικόνα 13: Το script της ροής των εντολών κατά τη φόρτωση του project στον Design Compiler.

Περίοδος ρολογιών:

Όπως έχει τονιστεί το σύστημα λόγω της χρήσης ενός αρκετά αργού divider στο σύστημα διαθέτει ένα ακόμα ρολό “extclk”. Το ρολό αυτό χρησιμοποιείται καθαρά και μόνο για τη λειτουργία του divider όπου έχει κατά το ήμισυ μικρότερη περίοδο, ώστε να εξάγει το αποτέλεσμα της διαίρεσης στο χρόνο που απαιτείται. Στον αντίποδα το υπόλοιπο σύστημα εκτός του divider χρησιμοποιεί ένα κοινό ρολό για όλες τις μονάδες του, το “clk” στην εικόνα. Όπως θα δειχθεί παρακάτω αναλυτικότερα (βλέπε παράγραφο report timing) το σύστημα για να υφίσταται λειτουργικό χωρίς τα setup violations αλλά και αργότερα τα hold violations, χρειάζεται μέγιστη γρηγορότερη περίοδο 25 (40 MHz) για το extclk και τη διπλάσια 50 (20 MHz) για το clk με το slack του χρόνου τόσο στην εξέταση για setup όσο και για hold να προκύπτει μεγαλύτερο του μηδενός στις δεδομένες αυτές τιμές.

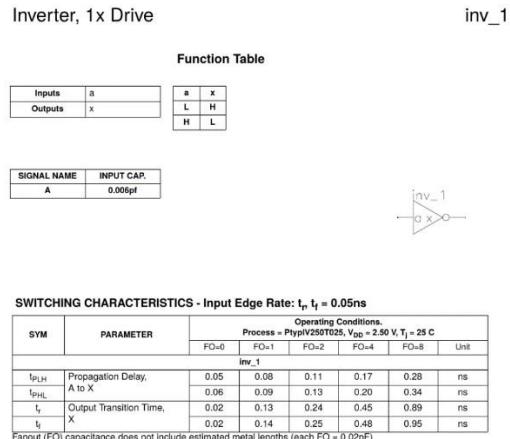
“Συχνότητα 20 MHz - clk (κοινό ρολόι)”

“Συχνότητα 40 MHz – extclk (ρολόι Divider)”

Ψευδή μονοπάτια και drive/capacity στις εισόδους/εξόδους του συστήματος:

Για να προσαρμοστεί το compile του συστήματος με τις ορθές παραμέτρους ορίστηκε ψευδές μονοπάτι το path από το reset, ενώ από την RTL του σχεδίου δεν βρέθηκε άλλο σήμα να οριστεί αναλόγως. Επιπλέον, από το documentation της βιβλιοθήκης της Motorola και εν συνέχεια με τις εντολές **drive of/load of**, πάρθηκε η τιμή του capacity και drive του inverter (inv_1). Με τις εντολές **set_drive/set_load** τοποθετείται η τιμή φορτίου **0.006109** στην έξοδο του συστήματος και το drive = **3.11117** στις εισόδους ώστε να αποδοθεί ρεαλισμός κατά το compile του σχεδίου.

Εικόνα 14: Προδιαγραφές του αντιστροφέα inv_1 από το documentation της βιβλιοθήκης πυλών της Motorola.



Compile και Write SDF / netlist:

Αφού ορίστηκαν όλες οι ρυθμίσεις που ήταν απαραίτητες στο σχέδιο γίνεται το compile και εκτελείται η εντολή **report_timing** τα αποτελέσματα της οποίας φαίνονται εκτενώς παρακάτω. Με το πέρασμα και αυτού του σταδίου εκτελούνται όπως φαίνεται και στην εικόνα 3, οι εντολές εξαγωγής για τη netlist και το SDF. Η netlist μαζί με το SDF (το συγκεκριμένο αφορά το worst case και το setup, παρακάτω θα εξαχθεί αντίστοιχο SDF για το hold) θα χρησιμοποιηθούν κατά την εκτέλεση του INCISIVE της Cadence για να προσομοιωθεί και να εξετασθεί η σωστή συμπεριφορά της netlist μέσα από τις κυματομορφές.

Στατική χρονική ανάλυση ως προς setup – time:

Report Timing: Κατά την στατική χρονική ανάλυση του συστήματος και του συγχρονισμού του, το **report_timing** είναι η εντολή που σε συνδυασμό με τη φόρτωση των worst-case βιβλιοθηκών, έδειχνε αναλόγως με τη τιμή της περιόδου του ρολογιού κατά πόσο υπήρχε παραβίαση στο setup-time. Όσο το slack ήταν αρνητικό χρειαζόταν να

αυξάνουμε τη περίοδο κάθε φορά στη διαδικασία μέχρι να εμφανιστεί σταθεροποίηση, δηλαδή τιμή θετική κοντά στο μηδέν ή μεγαλύτερη. Με την παράμετρο **-delay_type <min ή max>** στην εντολή report_timing ορίζεται αν είναι επιθυμητό να γίνει ανάλυση για setup (με το max) ή για το hold (με το min). Μη τοποθετώντας κάτι διπλά στην εντολή report_timing υπονοείται η παράμετρος max.

Κοινό ρολόι:

Report : timing			
-path full			
-delay max			
-max_paths 1			
Design : topmodule			
Version: M-2016.12			
Date : Sat Mar 31 14:45:38 2018			

Operating Conditions: nom_pvt Library: cdr3synPwclsLV225T125			
Wire Load Model Mode: enclosed			
Startpoint: PWM_0/period_reg[0]			
(rising edge-triggered flip-flop clocked by clk)			
Endpoint: PWM_0/ct_reg[1]			
(rising edge-triggered flip-flop clocked by clk)			
Path Group: clk			
Path Type: max			
Des/Clust/Port	Wire Load Model	Library	

topmodule	12K	cdr3synPwclsLV225T125	
PWM	2K	cdr3synPwclsLV225T125	
PWM_DW01_dec_0	1K	cdr3synPwclsLV225T125	
Point	Incr	Path	

clock clk (rise edge)	0.00	0.00	
clock network delay (ideal)	0.00	0.00	
PWM_0/period_reg[0]/ck (dffpr_2)	0.00	0.00 r	
PWM_0/period_reg[0]/q (dffpr_2)	0.76	0.76 f	
PWM_0/sub_84/A[0] (PWM_DW01_dec_0)	0.00	0.76 f	
PWM_0/sub_84/U76/x (nor2_0)	0.57	1.34 r	
PWM_0/sub_84/U75/x (nand2_0)	0.89	2.23 f	
PWM_0/sub_84/U74/x (nor2_0)	0.84	3.07 r	
PWM_0/sub_84/U73/x (nand2_0)	0.90	3.97 f	
PWM_0/sub_84/U72/x (nor2_0)	0.85	4.82 r	
PWM_0/sub_84/U71/x (nand2_0)	0.90	5.72 f	
PWM_0/sub_84/U70/x (nor2_0)	0.85	6.57 r	
PWM_0/sub_84/U69/x (nand2_0)	0.90	7.47 f	
PWM_0/sub_84/U68/x (nor2_0)	0.85	8.32 r	
PWM_0/sub_84/U67/x (nand2_0)	0.90	9.22 f	
PWM_0/sub_84/U65/x (nor2_0)	0.85	10.06 r	
PWM_0/sub_84/U63/x (nand2_0)	0.90	10.97 f	
PWM_0/sub_84/U61/x (nor2_0)	0.85	11.81 r	

PWM_0/sub_84/U61/x (nor2_0)	0.85	11.81 r	
PWM_0/sub_84/U59/x (nand2_0)	0.90	12.72 f	
PWM_0/sub_84/U57/x (nor2_0)	0.85	13.56 r	
PWM_0/sub_84/U55/x (nand2_0)	0.90	14.46 f	
PWM_0/sub_84/U53/x (nor2_0)	0.85	15.31 r	
PWM_0/sub_84/U51/x (nor2_0)	0.90	16.21 f	
PWM_0/sub_84/U49/x (nor2_0)	0.85	17.06 r	
PWM_0/sub_84/U46/x (nand2_0)	0.90	17.96 f	
PWM_0/sub_84/U44/x (nor2_0)	0.85	18.81 r	
PWM_0/sub_84/U42/x (nand2_0)	0.90	19.71 f	
PWM_0/sub_84/U40/x (nor2_0)	0.85	20.56 r	
PWM_0/sub_84/U38/x (nand2_0)	0.90	21.46 f	
PWM_0/sub_84/U36/x (nor2_0)	0.85	22.30 r	
PWM_0/sub_84/U34/x (nand2_0)	0.90	23.21 f	
PWM_0/sub_84/U32/x (nor2_0)	0.85	24.05 r	
PWM_0/sub_84/U30/x (nand2_0)	0.90	24.96 f	
PWM_0/sub_84/U28/x (nor2_0)	0.85	25.81 r	
PWM_0/sub_84/U27/x (ao21_1)	1.01	26.82 r	
PWM_0/sub_84/SUM[29] (PWM_DW01_dec_0)	0.00	26.82 r	
PWM_0/U41/(nand2_2)	0.31	27.13 f	
PWM_0/U42/(inv_2)	0.21	27.34 r	
PWM_0/U228/x (ao221_1)	0.28	27.62 f	
PWM_0/U229/x (ao221_1)	0.83	28.45 f	
PWM_0/U230/x (ao21_1)	0.37	28.82 r	
PWM_0/U231/x (ao22_1)	0.37	29.19 f	
PWM_0/U40/x (inv_2)	0.28	29.46 r	
PWM_0/U39/x (ao21_1)	0.28	29.74 f	
PWM_0/U26/(buf_3)	0.62	30.36 f	
PWM_0/U27/(nor2_1)	0.71	31.08 r	
PWM_0/U11/x (buf_3)	0.77	31.85 r	
PWM_0/U72/x (ao22_1)	0.71	32.56 r	
PWM_0/ct_reg[1]/d (dffpr_2)	0.00	32.56 r	
data arrival time		32.56	

clock clk (rise edge)	50.00	50.00	
clock network delay (ideal)	0.00	50.00	
PWM_0/ct_reg[1]/ck (dffpr_2)	0.00	50.00 r	
library setup time	-0.57	49.43	
data required time		49.43	

data required time		49.43	
data arrival time		-32.56	

slack (MET)		16.87	

Εικόνα 15: Εκτέλεση του report_timing για setup. Στην εικόνα φαίνονται τα αποτελέσματα στο γραφικό περιβάλλον του εργαλείου και αφορά το path για το κοινό ρολόι, ενώ δεξιά φαίνεται κάθε φορά το increment δίπλα στο path, ενώ το r ή f επιδεικνύει αν αφορά rise ή fall.

Συμπέρασμα: Το slack βγαίνει ίσο με +16.87. Αν και αρκετά θετικά μεγάλο, γεγονός που δείχνει πως τα δεδομένα φτάνουν αρκετά νωρίς, δεν επηρεάζεται η σωστή λειτουργία του συστήματος, όπως θα δούμε, και πρέπει να επισημανθεί ότι δε μπορεί να μειωθεί περαιτέρω η περίοδος καθώς πρέπει να παραμένει διπλάσια του δεύτερου ρολογιού extclk.

Ρολόι Divider:

Des/Clust/Port	Wire Load Model	Library		
topmodule	12K	cdr3synPwcslv225T125		
wishbone_master_pid	5K	cdr3synPwcslv225T125		
multiplier	2K	cdr3synPwcslv225T125		
FA_220	1K	cdr3synPwcslv225T125		
FA_207	1K	cdr3synPwcslv225T125		
FA_194	1K	cdr3synPwcslv225T125		
FA_180	1K	cdr3synPwcslv225T125		
FA_164	1K	cdr3synPwcslv225T125		
FA_151	1K	cdr3synPwcslv225T125		
FA_136	1K	cdr3synPwcslv225T125		
FA_122	1K	cdr3synPwcslv225T125		
FA_108	1K	cdr3synPwcslv225T125		
FA_93	1K	cdr3synPwcslv225T125		
FA_80	1K	cdr3synPwcslv225T125		
FA_67	1K	cdr3synPwcslv225T125		
FA_52	1K	cdr3synPwcslv225T125		
FA_38	1K	cdr3synPwcslv225T125		
FA_25	1K	cdr3synPwcslv225T125		
FA_24	1K	cdr3synPwcslv225T125		
FA_10	1K	cdr3synPwcslv225T125		
FA_9	1K	cdr3synPwcslv225T125		
Divide	1K	cdr3synPwcslv225T125		
Point	Incr	Path		
clock clk (rise edge)	0.00	0.00		
clock network delay (ideal)	0.00	0.00		
wishbone_master_pid_0/adjusted_dcycle_reg[1]/ck (dffpr_3)	0.00	0.00 r		
wishbone_master_pid_0/adjusted_dcycle_reg[1]/q (dffpr_3)	0.82	0.82 r		
wishbone_master_pid_0/multiplier_0/a[1] (multiplier)	0.00	0.82 r		
wishbone_master_pid_0/multiplier_0/inst487/U1/x (inv_4)	0.24	22.78 r		
wishbone_master_pid_0/multiplier_0/inst487/U3/x (nand2_5)	0.15	22.93 f		
wishbone_master_pid_0/multiplier_0/inst487/U4/x (nand2_5)	0.22	23.15 r		
wishbone_master_pid_0/multiplier_0/inst487/U6/x (ao22_3)	0.36	23.51 r		
wishbone_master_pid_0/multiplier_0/inst487/carry (FA_10)	0.00	23.51 r		
wishbone_master_pid_0/multiplier_0/inst488/I3 (FA_9)	0.00	23.51 r		
wishbone_master_pid_0/multiplier_0/inst488/U2/x (nand2_3)	0.21	23.72 f		
wishbone_master_pid_0/multiplier_0/inst488/U1/x (nand2_4)	0.19	23.91 r		
wishbone_master_pid_0/multiplier_0/inst488/carry (FA_9)	0.00	23.91 r		
wishbone_master_pid_0/multiplier_0/inst489/I3 (FA_8)	0.00	23.91 r		
wishbone_master_pid_0/multiplier_0/inst489/U2/x (exor2_2)	0.21	24.12 r		
wishbone_master_pid_0/multiplier_0/inst489/sum (FA_8)	0.00	24.12 r		
wishbone_master_pid_0/multiplier_0/m[23] (multiplier)	0.00	24.12 r		
wishbone_master_pid_0/Divide_0/A[23] (Divide)	0.00	24.12 r		
wishbone_master_pid_0/Divide_0/U3/x (ao222_4)	0.43	24.55 r		
wishbone_master_pid_0/Divide_0/result_reg[23]/d (dffpr_4)	0.00	24.55 r		
data arrival time			24.55	
clock extclk (rise edge)	25.00	25.00		
clock network delay (ideal)	0.00	25.00		
wishbone_master_pid_0/Divide_0/result_reg[23]/ck (dffpr_4)	0.00	25.00 r		
library setup time			24.55	
data required time			24.55	
data arrival time			24.55	
slack (MET)			0.01	

Εικόνα 16: Η συνέχεια των αποτελεσμάτων της εντολής report_timing και η ανάλυση με path group το extclk (ρολόι Divider).

Συμπέρασμα: Βλέπουμε πως για τη περίοδο ορισμένη στα 25, το slack βγαίνει οριακά θετικό και ίσο με 0.01.

Εστίαση στο hold – time και αλλαγή βιβλιοθηκών:

Για να ελεγχθούν τυχόν παραβιάσεις στο hold time πρέπει να αλλάξουμε τις βιβλιοθήκες από τη χειρότερη περίπτωση σε αυτή της καλύτερη. Έτσι στο script “synopsys_dc.setup” διαμορφώνουμε αναλόγως τις παραμέτρους ώστε να φορτωθούν οι min βιβλιοθήκες.

Hold Time: Ορίζεται η χρονική διάρκεια αμέσως μετά το σημείο μετάβασης κατά την οποία τα δεδομένα εισόδου πρέπει να παραμείνουν σταθερά στις λογικές τιμές τους για να εξασφαλιστεί ότι επιθυμητή μετάβαση του Flip-Flop πραγματοποιήθηκε σωστά.

Γιατί καλύτερη περίπτωση: Φορτώνοντας την καλύτερη περίπτωση (best-case) στη τεχνολογία της Motorola, δηλαδή υψηλή τάση τροφοδοσίας, χαμηλή θερμοκρασία κ.τ.λ., επιτυγχάνουμε τις λιγότερες δυνατές καθυστερήσεις. Έτσι τα δεδομένα φτάνουν αρκετά γρήγορα και όσα φαινόμενα παραβίασης hold-time μπορούν να προκύψουν θα εμφανιστούν σε αυτή την περίπτωση, δηλαδή την καλύτερη.

```

#
# Written by : DC-Transcript, Version 2003.06 -- May 20, 2003
# Date       : Mon Aug  4 19:23:48 2003
#



set company {IHP Frankfurt (Oder) GmbH}
set designer "Panagiotis_Anastasiadis"

set cache_read /tmp
set cache_write /tmp

set text_print_command {lpr -P }

set echo_include_commands false

set view_script_submenu_items [concat $view_script_submenu_items [list {remove all designs} {remove_design -designs}]]



set search_path [list "/home/faculty/chsotiriou/IHP-Lib/IHP-0.25um/digital/synopsys"]

set target_library sgc25a_min.db ← ΑΛΛΑΓΗ ΣΤΙΣ ΒΙΒΛΙΟΘΗΚΕΣ
set symbol_library sgc25a.sdb
set link_library [concat "*" $target_library]

# /* add I/O library: */
set target_library [concat $target_library sgc25a_io_min.db]
set symbol_library [format "%s%s" $symbol_library sgc25a_io.sdb]
set link_library [concat "*" $target_library]

```

Εικόνα 17: Η αλλαγή στο script του setup από max σε min των βιβλιοθηκών.

Ροή εντολών για εξέταση hold-time violation:

Ξεκινώντας τη διαδικασία για να πραγματοποιήσουμε στατική χρονική ανάλυση με εστίαση το hold time φορτώνουμε τα αποθηκευμένα .ddc αρχεία του σχεδίου ώστε να γίνουν compile με τις min βιβλιοθήκες. Στην εντολή compile τοποθετείται η παράμετρος -only_hold_time και εστιάζει τη διαδικασία και τη βελτιστοποίηση της σύνθεσης ως προς τις hold time παραβιάσεις. Το script των εντολών για τις περιόδους των ρολογιών, τα false paths, τα drive-loads στα inputs/outputs παραμένει το ίδιο με τη διαφορά ότι γράφεται νέο SDF που αφορά το νέο compile για το hold.

```

1 gui_start
2 read_file -format ddc {/home/inf2015/paanastasiadis/teliko/adder_32bit_0.ddc /home/inf2015/paanastasiadis/teliko/adder_32bit_1.ddc /home/inf2015/paanastasiadis/teliko/boot_array.ddc /home/inf2015/
3 current_design topmodule
4 compile -map_effort high -only_hold_time

```

Εικόνα 18: Η ροή εκτέλεσης των εντολών μετά τη φόρτωση των min βιβλιοθηκών και το compile του σχεδίου.

```

design_vision> write_sdf -version 1.0 hold.sdf
Information: Annotated 'cell' delays are assumed to include load delay. (UID-282)
Information: Writing timing information to file '/home/inf2015/paanastasiadis/teliko/hold.sdf'. (WT-3)
1
design_vision>

```

Εικόνα 19: Εξαγωγή SDF μετά το compile για το hold.

Στατική χρονική ανάλυση ως προς hold – time:

Report Timing: Όπως προαναφέρθηκε παραπάνω η εντολή για την χρονική ανάλυση στον design compiler είναι η report_timing. Για να εξετάσουμε τα τυχόν hold violations στο σχέδιο τοποθετείται η παράμετρος -delay_type min. Στις παρακάτω εικόνες αποδεικνύεται πως για τις περιόδους του ρολογιού που προαναφέρθηκαν, το slack στη hold ανάλυση βγαίνει αντιστοίχως με το setup θετικό έτσι οι τιμές κρίνονται έγκυρες.

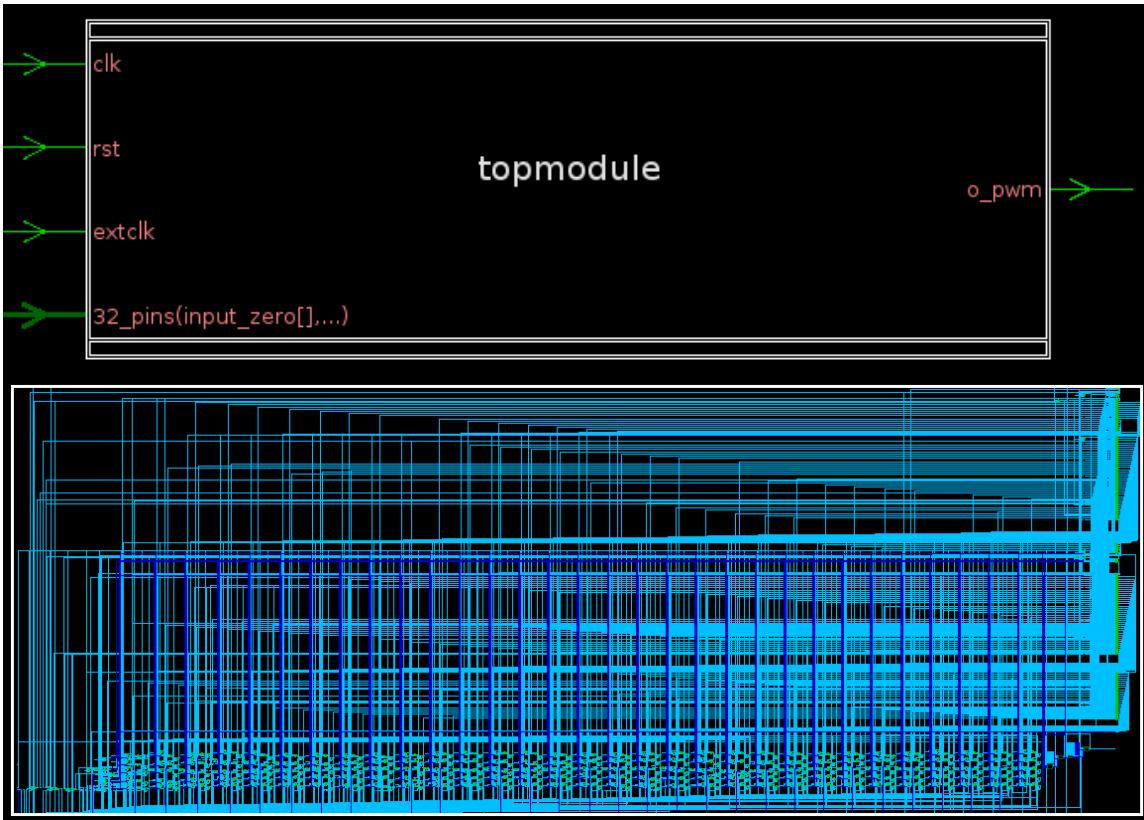
Design : topmodule Version: M-2016.12 Date : Sat Mar 31 15:06:24 2018 *****					
Operating Conditions: nom_pvt Library: cdr3synPbcslv270Tm040 Wire Load Model Mode: enclosed					
Startpoint: P1D0/multiplier_16x16bit_pipelined/reg_layer_2_w31_reg (rising edge-triggered flip-flop clocked by clk)			Startpoint: wishbone_master_pid_0/Divide_0/denom_reg[0] (rising edge-triggered flip-flop clocked by extclk)		
Endpoint: P1D0/multiplier_16x16bit_pipelined/reg_layer_2_w31_reg (rising edge-triggered flip-flop clocked by clk)			Endpoint: wishbone_master_pid_0/Divide_0/denom_reg[0] (rising edge-triggered flip-flop clocked by extclk)		
Path Group: clk		Path Group: extclk	Path Type: min		
Path Type: min					
Des/Clust/Port	Wire Load Model	Library	Des/Clust/Port		
topmodule	12K	cdr3synPbcslv270Tm040	topmodule	12K	cdr3synPbcslv270Tm040
multiplier_16x16bit_pipelined	3K	cdr3synPbcslv270Tm040	Divide	1K	cdr3synPbcslv270Tm040
Point	Incr	Path	Point		
clock clk (rise edge)	0.00	0.00	clock extclk (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00	clock network delay (ideal)	0.00	0.00
P1D0/multiplier_16x16bit_pipelined/reg_layer_2_w31_reg/ck (dffpr_2)	0.00	0.00 r	wishbone_master_pid_0/Divide_0/denom_reg[0]/ck (dffpr_2)	0.00	0.00 r
P1D0/multiplier_16x16bit_pipelined/reg_layer_2_w31_reg/qb (dffpr_2)	0.22	0.22 r	wishbone_master_pid_0/Divide_0/denom_reg[0]/qb (dffpr_2)	0.22	0.22 r
P1D0/multiplier_16x16bit_pipelined/U145/x (nand2_2)	0.04	0.26 f	wishbone_master_pid_0/Divide_0/U177/x (oai22_1)	0.05	0.27 f
P1D0/multiplier_16x16bit_pipelined/reg_layer_2_w31_reg/d (dffpr_2)	0.00	0.26 f	wishbone_master_pid_0/Divide_0/denom_reg[0]/d (dffpr_2)	0.00	0.27 f
data arrival time	0.26		data arrival time	0.27	
clock clk (rise edge)	0.00	0.00	clock extclk (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00	clock network delay (ideal)	0.00	0.00
P1D0/multiplier_16x16bit_pipelined/reg_layer_2_w31_reg/ck (dffpr_2)	0.00	0.00 r	wishbone_master_pid_0/Divide_0/denom_reg[0]/ck (dffpr_2)	0.00	0.00 r
library hold time	-0.04	-0.04	library hold time	-0.03	-0.03
data required time	-0.04		data required time	-0.03	
data required time	-0.04		data required time	-0.03	
data arrival time	-0.26		data arrival time	-0.27	
slack (MET)	0.30		slack (MET)	0.30	

Εικόνα 20: Αριστερά φαίνεται τα αποτελέσματα για το κοινό ρολόι clk και δεξιά για το ρολόι του Divider extclk.

Συμπέρασμα: Το slack και στις δυο περιπτώσεις βγαίνει θετικό και ίσο με 0.30, οπότε δεν υπάρχει καμία παραβίαση.

Σχηματικό:

Χρήσιμη να τονιστεί είναι η δυνατότητα που προσφέρει ο Design Compiler της Synopsys να παράγει σχηματικό του σχεδίου μέσα από το γραφικό περιβάλλον. Στις παρακάτω εικόνες φαίνεται το σχηματικό του συστήματος έτσι όπως κατασκευάστηκε από το εργαλείο και παράχθηκε στην οθόνη. (Σημείωση: τα σχήματα παρακάτω αφορούν την απεικόνιση του σχεδίου πριν να εισαχθούν σε αυτό τα scan – chains από το DFT).



Εικόνα 21: Το σχηματικό του σχεδίου έτσι όπως εμφανίζεται από το γραφικό περιβάλλον του Design Compiler.

Εντολή report_area:

Χρήσιμη εντολή είναι και η report_area με την οποία παρουσιάζεται ο αριθμός των cells που καταλαμβάνει το σύστημα, καθώς επίσης παρουσιάζονται σε κατηγορίες ο αριθμός των ports, nets, το εμβαδόν της combinational area, της sequential area κ.τ.λ. Τονίζεται εδώ πως η εικόνα αφορά την εντολή όταν αυτή εκτελέστηκε με τις min βιβλιοθήκες.

```
design_vision> report_area
*****
Report : area
Design : topmodule
Version: M-2016.12
Date  : Sat Mar 31 15:08:38 2018
*****
Library(s) Used:
  cdr3synPbcsLV270Tm040 (File: /home/faculty/chsotiriou/IHP-Lib/IHP-0.25um/digital/synopsys/scg25a.min.db)

Number of ports:          4516
Number of nets:           10166
Number of cells:          5870
Number of combinational cells: 4311
Number of sequential cells: 977
Number of macros/black boxes: 0
Number of buf/inv:         629
Number of references:      5

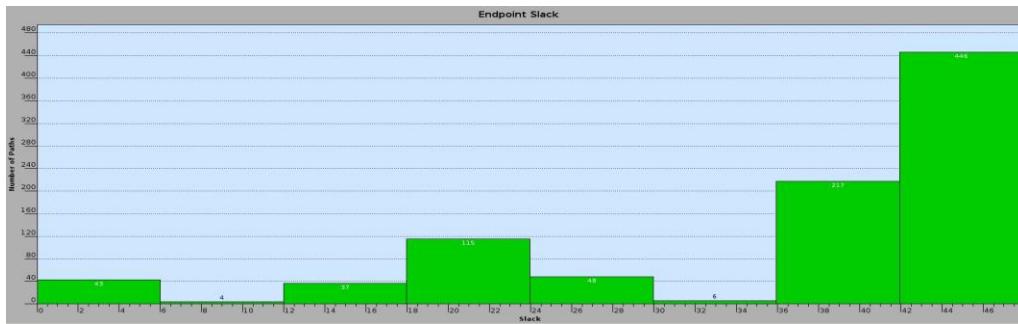
Combinational area:      598243.750000
Buf/Inv area:            43204.687500
Noncombinational area:   316692.187500
Macro/Black Box area:    0.000000
Net Interconnect area:  undefined (Wire load has zero net area)

Total cell area:          914935.937500
Total area:                undefined
1
```

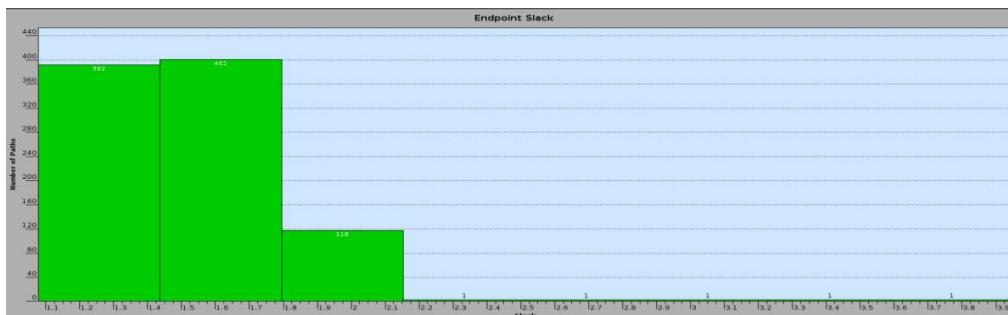
Εικόνα 22: Εκτέλεση της εντολής report_area και η εμφάνιση των αποτελεσμάτων (Total Cell Area 914935 um).

Ιστογράμματα:

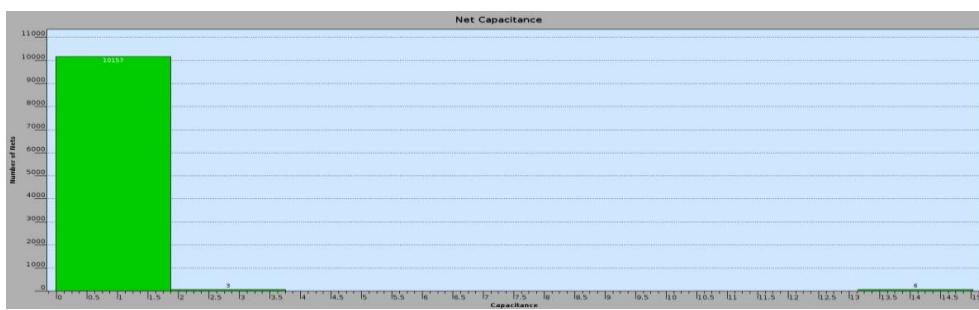
Ο Design Compiler περιέχει την παραγωγή χρήσιμων ιστογραμμάτων που διευκολύνουν την απεικόνιση πληροφορίων του σχεδίου για την εύκολη εξαγωγή συμπερασμάτων. Παρακάτω φαίνονται 3 ιστογράμματα, 2 που αφορούν το **Endpoint Slack** κατά την setup ανάλυση το πρώτο και τη hold ανάλυση το δεύτερο, ενώ το τρίτο αφορά το **Net Capacitance**.



Εικόνα 23: Το ιστόγραμμα για το endpoint slack κατά τη setup ανάλυση. Επειδή όλα τα endpoints βρίσκονται δεξιά στον θετικό άξονα, σημαίνει πως έχουν περάσει επιτυχώς τα constraints τους και το σύστημα λειτουργεί ορθά. Worst slack = 0.008, Best Slack 47.78.



Εικόνα 24: Το ιστόγραμμα για το endpoint slack κατά τη hold ανάλυση. Worst Slack = 1.234, Best Slack = 3.944



Εικόνα 25: Το ιστόγραμμα για το Net Capacitance. Φαίνεται ότι τα περισσότερα Nets (σύνολο 10.157) βρίσκονται κοντά στο 0 που είναι και το επιθυμητό, με εξαίρεση μόνο 6 Nets που φτάνουν τη χωρητικότητα στο 15.019.

Προσομοίωση συμπεριφοράς της netlist μαζί με το αρχείο SDF από την setup ανάλυση:

Εφόσον εξάχθηκαν τα αρχεία της netlist και αυτά των δύο SDF, ένα από την setup ανάλυση και ένα από τη hold, επόμενο βήμα είναι η προσομοίωση της netlist στο INCISIVE της Cadence, ώστε να διαπιστωθεί ξανά αν η συμπεριφορά της είναι ορθή. Οι προσομοιώσεις που θα πραγματοποιηθούν είναι 2, στη μια η netlist θα φορτωθεί στο εργαλείο με το SDF του setup και στην άλλη με το SDF του hold. Στα αρχεία που επίσης θα απαιτηθούν, εντάσσεται η zxall.v βιβλιοθήκη Verilog της Motorola και το behavioral testbench που διαθέτει το κώδικα προσομοίωσης συμπεριφοράς του συστήματος. Το testbench αυτό υλοποιεί το ίδιο σενάριο λειτουργίας έτσι όπως αυτό αναλύθηκε παραπάνω.

```
reg clk = 0;
reg rst = 1;
reg extclk = 0;

initial begin
    #70 rst=0;
end

always #25 clk = ~clk;
always #12.5 extclk = ~extclk;
```

`$(sdf_annotate()` στον κώδικα του testbench, ώστε να γίνεται το αρχείο compile και elaborate από το INCISIVE.

Μοναδικές αλλαγές που επιδέχθηκαν το testbench της προσομοίωσης αφορούν αρχικά των ορισμό των περιόδων των ρολογιών στις τιμές που δείχθηκε ότι δουλεύει το σύστημα μέσα από το Design Compiler, δηλαδή τα 20 και τα 40 MHz. Εδώ να σημειωθεί ότι επειδή η αλλαγή κατάστασης στο reset σήμα διαδραματίζόταν αρκετά κοντά στις ακμές των ρολογιών και δημιούργησε προβλήματα στη προσομοίωση, τέθηκε αναλόγως όπως φαίνεται παρακάτω. Τέλος για την αξιοποίηση των SDF στην προσομοίωση, προστέθηκε η εντολή

```
initial begin
    $(sdf_annotate ("setup.sdf", topmodule_0, "maximum"));
end
```

Κυματομορφές ορθής λειτουργίας:

Στις πρώτες δύο κυματομορφές θα αποδειχθεί ότι η netlist λειτουργεί σωστά και τα αποτέλεσμα που προκύπτουν είναι τα αναμενόμενα. Το σενάριο λειτουργίας είναι αυτό της 10% λειτουργίας για το Set-Point (βλέπε σελίδα Σημαντικές υπενθυμίσεις και επεξηγήσεις-Σενάριο Λειτουργίας).



Εικόνα 26: Στη κυματομορφή αυτή που αφορά τη μονάδα του wishbone master pid, αποδεικνύεται πως το duty cycle επιτυχώς συγκλίνει στους 702 κύκλους και το pv (Process-Value) διαμορφώνεται όντως στο 10% που επιθυμούμε.

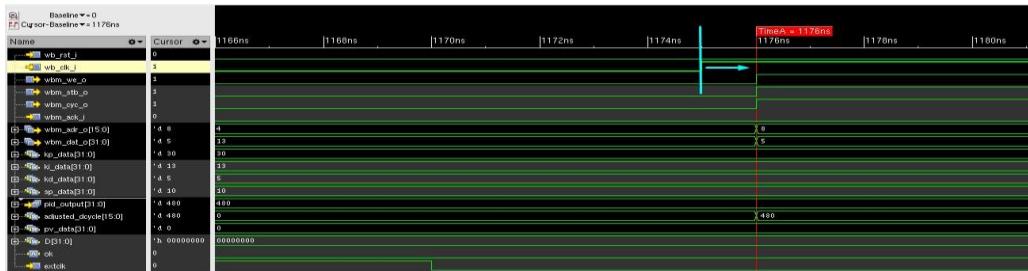


Εικόνα 27: Διαισθητικά φαίνεται και εδώ πως το σήμα στην έξοδο διαθέτει duty cycle 10%.

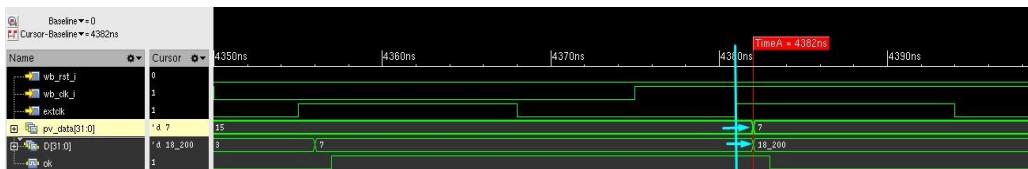
Φαίνεται λοιπόν, πως το σύστημα συμπεριφέρεται ορθά. Η netlist μαζί με το αρχείο SDF του Setup και τις όποιες καθυστερήσεις δημιουργούνται, δεν επηρεάζεται στη συμπεριφορά της και προσφέρει το ίδιο αποτέλεσμα με αυτά που έδειξε η RTL στη πρώτη προσομοίωση συμπεριφοράς.

Εμφάνιση Καθυστερήσεων:

Παρακάτω παρουσιάζονται οι καθυστερήσεις μεταξύ ρολογιού και των καταχωρητών flip-flops, έτσι όπως είναι λογικό να εμφανιστούν λόγω του SDF αρχείου που αφορά τη setup ανάλυση, όταν αυτή πραγματοποιήθηκε στη χειρότερη περίπτωση της τεχνολογίας της Motorola.



Εικόνα 27: Στην κυματομορφή αυτή, ξανά παρμένη από τη μονάδα του Wishbone Master PID, βλέπουμε σε σμίκρυνση τις καθυστερήσεις που δημιουργούνται τώρα μεταξύ του κοινού ρολογιού και των υπόλοιπων registers. Στη γαλάζια γραμμή φαίνεται η καθυστέρηση που εμφανίζεται μεταξύ της θετικής ακμής του κοινού ρολογιού και παραδείγματος χάρη των σημάτων we, stb, cyc που αφορούν τα write enable σήματα για να γράψει ο master στον slave.

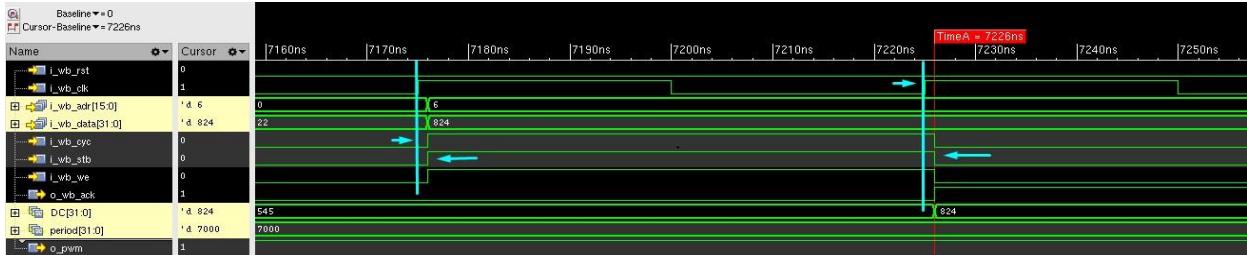


Εικόνα 28: Και πάλι η προσομοίωση από τη Wishbone Master PID μονάδα, αλλά με εστίαση στο extclk (το ρολόι του Divider). Φαίνεται και σε αυτή τη περίπτωση πως υπάρχει μικρή αλλά υπαρκτή καθυστέρηση από την ακμή του ρολογιού μέχρι την αλλαγή των δεδομένων στους καταχωρητές. Τα σήματα που εξαρτώνται από το ρολόι του Divider είναι το pv_data(Process Value) και το D(αποτέλεσμα της διαίρεσης).

Παρόμοιες καθυστερήσεις εμφανίζονται και στις άλλες μονάδες του συστήματος. Στις παρακάτω εικόνες φαίνονται οι κυματομορφές με έμφαση στις καθυστερήσεις από τα υπόλοιπα units.



Εικόνα 29: Καθυστερήσεις στη μονάδα του PID Unit.



Εικόνα 30: Καθυστερήσεις στη μονάδα του PWM Unit.



Εικόνα 31: Καθυστερήσεις στη μονάδα του Wishbone Master PWM Unit.

Προσομοίωση συμπεριφοράς της netlist μαζί με το αρχείο SDF από την hold ανάλυση:

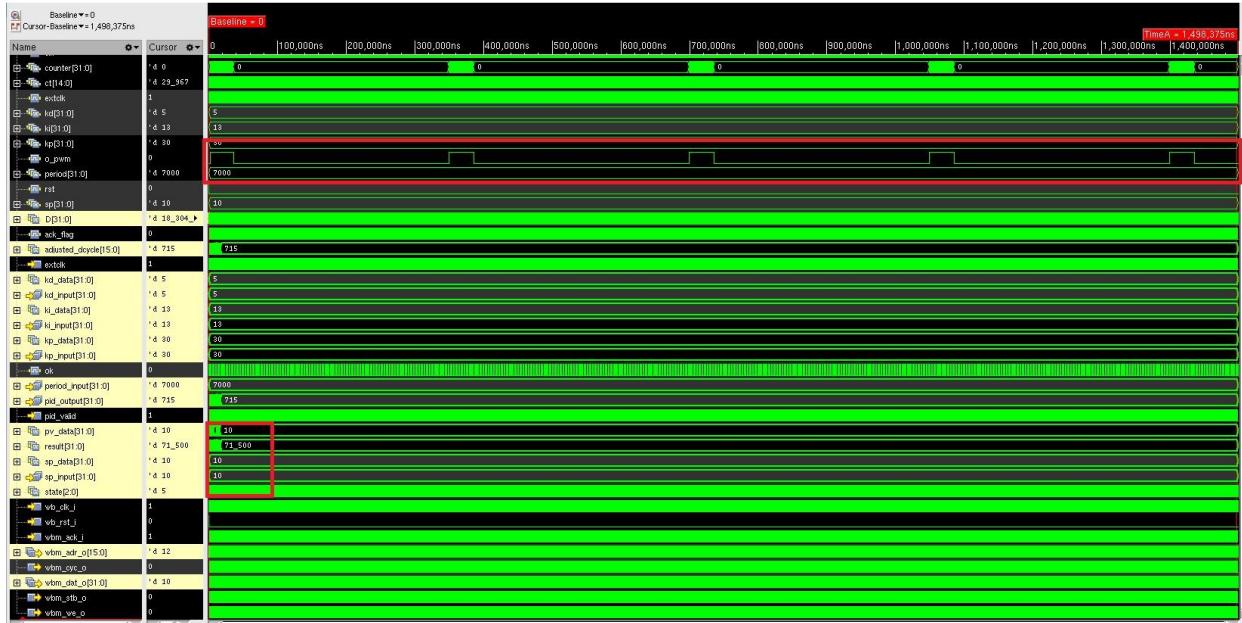
Αφού προσομοιώθηκε η netlist του συστήματος με τη φόρτωση του SDF από τη setup ανάλυση προχωράμε στην αντίστοιχη εκτέλεση της διαδικασίας με το hold SDF. Αποδείχθηκε με τις όποιες καθυστερήσεις εμφανίστηκαν κατά τη προσομοίωση, ότι η συμπεριφορά του συστήματος λειτουργεί ορθά και το ίδιο θα αποδειχθεί και τώρα με το αρχείο SDF από τη hold ανάλυση.

Σε πρώτο βήμα φορτώνουμε το διαφορετικό SDF αρχείο. Αφού γίνει η προσομοίωση, στις παρακάτω εικόνες παρουσιάζεται και πάλι πως το σύστημα αυτή τη φορά με διαφορετικό SDF αρχείο εξακολουθεί να εξάγει την ορθή αναμενόμενη συμπεριφορά.

```
initial begin
    $sdf_annotate ("hold.sdf", topmodule_0, "minimum");
end
```

Εικόνα 32: Φόρτωση του SDF από τη hold ανάλυση.

Κυματομορφή ορθής λειτουργίας:

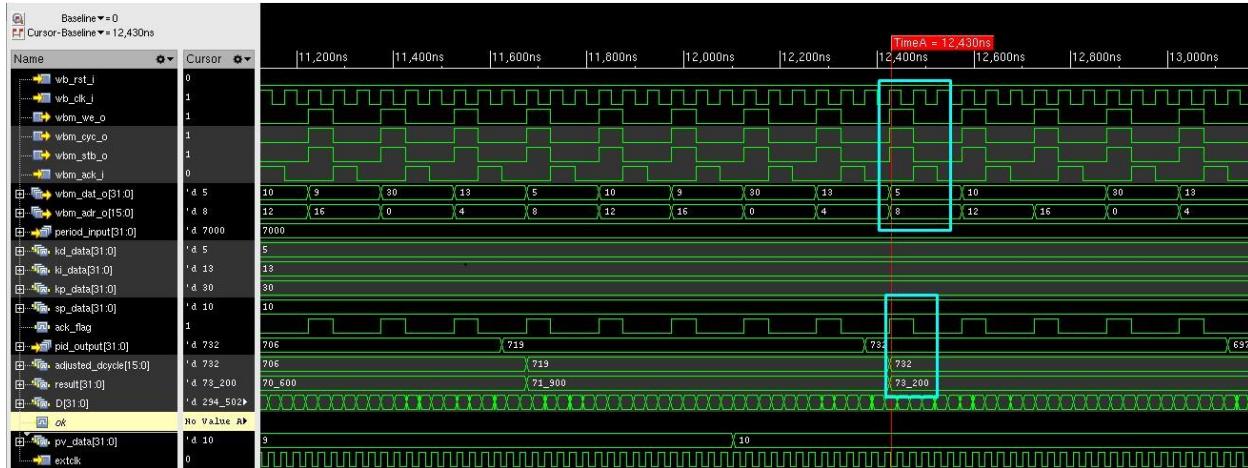


Εικόνα 33: Στη κυματομορφή αυτή παρουσιάζονται τόσα τα σήματα από το Wishbone Master PID όσο και από το behavioral testbench. Τα αποτελέσματα, όπως φαίνονται στα κόκκινα πλαίσια, είναι τα αναμενόμενα και τα ίδια με αυτά που εμφανίστηκαν και στη προηγούμενη προσομοίωση για το setup.

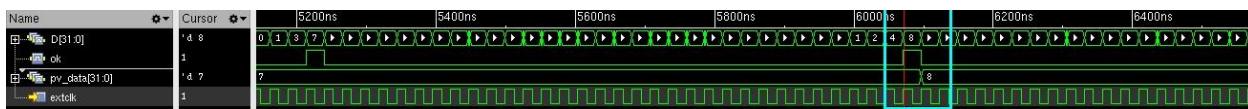
Καθυστερήσεις:

Όπως αναφέρθηκε στη διαδικασία που ακολουθήθηκε για την χρονική ανάλυση και την εξέταση για τα hold violations, στο Design Compiler φορτώθηκαν οι min βιβλιοθήκες. Οι βιβλιοθήκες αυτές αφορούσαν την καλύτερη περίπτωση στις πύλες της τεχνολογίας, όποτε αναμέναμε τις μικρότερες δυνατές καθυστερήσεις και το πιο αισιόδοξο σενάριο για το σύστημα.

Στις προσομοιώσεις, πιθανώς και λόγω της μικρής πολυπλοκότητας του συστήματος σε συνδυασμό με το SDF για το hold που αφορά και τη βέλτιστη κατάσταση, οι καθυστερήσεις στο INCISIVE είναι μηδαμινές, σχεδόν ανύπαρκτες και πλησιάζουν την ιδανική λειτουργία.



Εικόνα 34: Στα γαλάζια πλαίσια φαίνεται η μηδαμινή καθυστέρηση μεταξύ του κοινού ρολογιού και των registers στη μονάδα του Wishbone Master PID.



Εικόνα 35: Στην μονάδα του Wishbone Master PID με έμφαση στα registers που είναι ακμοπυροδότητα από το ρολόι του Divider. Και εδώ η καθυστέρηση είναι κοντά στο μηδέν.

Συμπέρασμα: Και στις δυο περιπτώσεις (setup και hold) τόσο ο design compiler όσο και το incisive δείχνουν πως για τις δεδομένες συχνότητες, το σύστημα αφού μετατράπηκε σε netlist συμπεριφέρεται ορθά και εξάγει την επιθυμητή λειτουργία και συμπεριφορά. Στο Synopsys το slack για hold και setup βγαίνει θετικό για τις περιόδους που ορίστηκαν και στο INCISIVE παρά τις όποιες καθυστερήσεις που δημιουργούνται από τη στιγμή που ξεφεύγουμε από την ιδανική λειτουργία η συμπεριφορά μένει σωστή και απαράλλαχτη.

Εισαγωγή δομών DFT (Design For Test)

Σκοπός:

ο Design Compiler της Synopsys έχει τη δυνατότητα να εισάγει στο σχέδιο τις αλυσίδες σάρωσης (scan chains), με τις οποίες γίνεται εφικτό να εκτελεστεί μετέπειτα στο TetraMAX ο αλγόριθμος του ATPG και να παραχθούν τα patterns που χρειάζονται για την επαλήθευση ενός κατασκευασμένου κυκλώματος.
Στη ροή που ακολουθείται, τα βήματα είναι συνοπτικά τα εξής:

- Εκτελείται η εντολή **dft_drc** με την οποία εμφανίζονται όλες οι παραβιάσεις που πιθανόν υπάρχουν πριν από την εισαγωγή του DFT. Στο δικό μας σύστημα, οι παραβιάσεις αφορούσαν τα ρολόγια και το reset που έπρεπε να δηλωθούν ασύγχρονα.
- Στην εντολή **create_test_protocol** που απαιτείται πριν την εισαγωγή των δομών, προστέθηκαν οι παράμετροι **-infer_asynch -infer_clock** και διόρθωσαν το πρόβλημα. Τα violations είναι πια μηδενικά.
- Στο σημείο αυτό μπορεί τώρα να τρέξει η κυρία εντολή, το **insert_dft**, το οποίο και εισάγει τα scan-chains στο σχέδιο.
- Με τη παράμετρο **-coverage_estimate** στην εντολή του **dft_drc**, μπορεί να δει κανείς ποια θα είναι περίπου η αναμενόμενη κάλυψη-coverage που θα πετύχει το ATPG ως προς τον έλεγχο του σχεδίου, όταν θα τρέξει πάνω στη netlist στο TetraMAX για να εξάγει τα patterns. Ικανοποιητικό αποτέλεσμα ή στόχος είναι η τιμή να ξεπερνά το 95% για να υπάρχει η επιθυμητή ακρίβεια και στο σχέδιο μας, όπως θα φανεί και παρακάτω, το coverage φτάνει το 99,99% .
- Το **compile -incremental_mapping** είναι αυτό που εκτελείται ως επόμενο βήμα, όπου η incremental παράμετρος οδηγεί τον compiler να κάνει μόνο αλλαγές ή βελτιστοποιήσεις στη σύνθεση, εφόσον η εισαγωγή των scan-chains το καθιστά απαραίτητο. Στο δεδομένο σχέδιο θα δειχθεί πως δε χρειάστηκε τελικώς κάποια αλλαγή σε αυτό το σημείο.
- Στη συνέχεια εκτελούμε την **report timing** ξανά τόσο για setup και hold, καθώς η εισαγωγή του DFT είναι δυνατό να επηρεάσει το slack. Όπως και με την incremental διαδικασία, έτσι και εδώ δεν υπήρξε αλλαγή (negative slack δηλαδή) που να αναγκάσει την αύξηση της περιόδου σε τιμή μεγαλύτερη από αυτές που αναφέρθηκαν παραπάνω.
- Τέλος με το **write_test_protocol -output scan_protocol.stil** εξάγουμε το αρχείο stil που μαζί με την netlist απαιτούνται κατά τη ροή εκτέλεσης του TetraMAX, όπως θα δούμε και παρακάτω.

Ακολουθούν το σύνολο των αποτελεσμάτων της ροής, έτσι όπως παράχθηκαν από το Design Compiler στο πλαίσιο της εισαγωγής του DFT.

```
#####
##### create_test_protocol for dft
create_test_protocol -infer_asynch -infer_clock

#####
##### dft_drc + scan chain insertion + dft coverage + report_timing
dft_drc
insert_dft
dft_drc -coverage_estimate
compile -incremental_mapping
report_timing

#####
##### file needed for TetraMAX
change_names -hierarchy -rule verilog
write_test_protocol -output scan_protocol.stil
```

Εικόνα 36: Οι εντολές του script που αφορούν το DFT.

```
...checking for scan equivalents...
...checking vector rules...
...checking pre-dft rules...

-----
DRC Report
Total violations: 0

-----
Test Design rule checking did not find violations

-----
Sequential Cell Report
0 out of 917 sequential cells have violations

-----
SEQUENTIAL CELLS WITHOUT VIOLATIONS
* 917 cells are valid scan cells
```

Εικόνα 37: Τα αποτελέσματα από την εντολή dft_drc. Όπως φαίνεται στην εικόνα οι συνολικές παραβιάσεις είναι μηδενικές.

```
Beginning Incremental Implementation Selection
-----
Selecting implementations
Building model 'DW01_NAND2'
Building model 'DW01_dec_width32' (rpl)
Building model 'DW01_inc_width32' (rpl)
Selecting implementations
Building model 'DW01_sub_width33' (rpl)

Beginning Delay Optimization Phase
-----

ELAPSED          TOTAL
TIME     AREA    WORST   NEG    SETUP   DESIGN
          COST     COST    RULE COST   ENDPOINT
-----
```

ELAPSED TIME	AREA	WORST COST	NEG COST	SETUP COST	DESIGN RULE COST	ENDPOINT
0:00:02	993829.7	0.00	0.0	0.0	0.0	
0:00:03	993829.7	0.00	0.0	0.0	0.0	
0:00:04	993829.7	0.00	0.0	0.0	0.0	
0:00:04	993829.7	0.00	0.0	0.0	0.0	

```
Loading db file '/home/faculty/chsotiriou/IHP-Lib/IHP-0.25um/digital/synopsys/sgc25a.max.db'
Loading db file '/home/faculty/chsotiriou/IHP-Lib/IHP-0.25um/digital/synopsys/sgc25a_io.max.db'

Note: Symbol # after min delay cost means estimated hold TNS across all active scenarios
```

Εικόνα 38: Τα αποτελέσματα μετά το compile -incremental. Η σύνθεση δεν επιδέχθηκε κάποια βελτιστοποίηση μετά την εισαγωγή των scan – chains.

```

Uncollapsed Stuck Fault Summary Report
-----
fault class      -code #faults
-----
Detected          DT    42016
Possibly detected PT     0
Undetectable     UD    3806
ATPG untestable  AU     0
Not detected     ND     4
-----
total faults      45826
test coverage     99.99%
-----
Information: The test coverage above may be inferior
              than the real test coverage with customized
              protocol and test simulation library.

```

Εικόνα 39: Η εκτέλεση του dft_drc -coverage_estimate. Η κάλυψη φτάνει στο 99,99%.

```

clock extclk (rise edge)           25.00   25.00
clock network delay (ideal)        0.00    25.00
wishbone_master_pid_0/Divide_0/result_reg[22]/ck (sdffpr_4)
                                         0.00   25.00 r
library setup time                 -0.81   24.19
data required time                  24.19
data arrival time                   -24.18
-----
slack (MET)                         0.01
clock clk (rise edge)             50.00   50.00
clock network delay (ideal)        0.00    50.00
PWM_0/ct_Reg[1]/ck (sdffpr_2)      0.00   50.00 r
library setup time                 -0.90   49.10
data required time                  49.10
data arrival time                   -32.54
-----
slack (MET)                         16.55

```

Εικόνα 40: Report Timing. Οι αλυσίδες σάρωσης δεν επηρέασαν το slack ώστε αυτό να γίνει αρνητικό για τις ορισμένες περιόδους των 2 ρολογιών.



Εικόνα 41: Τα νέα ιστογράμματα Endpoint Slack (setup ανάλυση δεξιά, hold αριστερά) και Net Capacitance στη μέση μετά την εισαγωγή του DFT. Το endpoint slack και στις δύο περιπτώσεις έχει αυξηθεί ελάχιστα και κατά μια περίπου μονάδα.

Ροή στο TetraMAX, ATPG και προσομοίωση του αυτόματου MAX Testbench στο INCISIVE:

Μετά τις ενέργειες στο Design Compiler και την ενσωμάτωση του DFT στο σχέδιο, το project περνά στο εργαλείο **TetraMAX** της **Synopsys**. Το αρχείο .stil και η netlist του σχεδίου με τις ενσωματωμένες δομές DFT που συντέθηκαν στον Design Compiler χρησιμοποιούνται στο TetraMAX. Κύριοι σκοποί μας ως προς το εργαλείο είναι:

- Να εφαρμόζει το πρόγραμμα Αυτόματης Δημιουργίας Διανυσμάτων Δοκιμής (ATPG) στο σχέδιο και να δείξει το ποσοστό κάλυψης του συστήματος που καταλαμβάνει η εφαρμογή του. **Στόχος η κάλυψη να ξεπερνά το 95%**.
- Να γράψει τα patterns στα οποία εφαρμόζεται ο αλγόριθμος και να παράγει βάσει αυτών το αυτόματο MAX Testbench. Κατόπιν το testbench μπορεί να εκτελεσθεί στο Incisive της Cadence και να δώσει μια άλλη μορφή επαλήθευσης μέσω των κυματομορφών, όπου θα σαρωθούν τα προηγουμένως δημιουργημένα patterns. **Η σάρωση των patterns θα γίνει σειριακά.**

1^o στάδιο ροής: Σε πρώτη φάση φορτώνουμε στο εργαλείο τη netlist του σχεδίου και zxall.v που είναι η βιβλιοθήκη πυλών της Motorola. Όπως φαίνεται παρακάτω διαβάζονται επιτυχώς και χωρίς κανένα σφάλμα.

```
BUILD-T> read_netlist /home/inf2015/paanastasiadis/tmax/netlist.v
  Begin reading netlist ( /home/inf2015/paanastasiadis/tmax/netlist.v )...
  End parsing Verilog file /home/inf2015/paanastasiadis/tmax/netlist.v with 0 errors.
  End reading netlist: #modules=663, top=topmodule, #lines=13252, CPU_time=0.03 sec, Memory=4MB
BUILD-T> read_netlist /home/inf2015/paanastasiadis/tmax/zxall.v
  Begin reading netlist ( /home/inf2015/paanastasiadis/tmax/zxall.v )...
  End parsing Verilog file /home/inf2015/paanastasiadis/tmax/zxall.v with 0 errors.
  End reading netlist: #modules=567, top=iobufsdv_16pu_8, #lines=45843, CPU_time=0.03 sec, Memory=1MB
```

Εικόνα 42: Φόρτωση της zxall.v και της netlist στο TetraMAX.

2^o στάδιο ροής: Φορτώνουμε το αρχείο .stil που παράχθηκε στο Design Compiler και εκτελούμε την εντολή run_drc για να ελεγχθεί το σχέδιο για τυχόν παραβιάσεις πριν την εφαρμογή του ATPG. Καμία παραβίαση, όπως αποδεικνύεται και από την εικόνα, κατά τον DRC έλεγχο.

```
DRC-T> set_drc /home/inf2015/paanastasiadis/tmax/scan_protocol.stil
DRC-T> run_drc
-----
Begin scan design rules checking...
-----
Begin reading test protocol file /home/inf2015/paanastasiadis/tmax/scan_protocol.stil...
End parsing STIL file /home/inf2015/paanastasiadis/tmax/scan_protocol.stil with 0 errors.
Test protocol file reading completed, CPU time=0.00 sec.
-----
Begin simulating test protocol procedures...
Test protocol simulation completed, CPU time=0.00 sec.
-----
Begin scan chain operation checking...
The chain 1 successfully traced with 134 scan cells.
Chain 2 successfully traced with 134 scan cells.
Scan chain operation checking completed, CPU time=0.00 sec.
-----
Begin clock rules checking...
Clock rules checking completed, CPU time=0.00 sec.
Clock grouping results: #pairs=0, #groups=0, #serial_pairs=0, #disturbed_pairs=1, CPU time=0.00 sec.
-----
Begin nonscan rules checking...
Nonscan cell summary: #DFPs=0 #DLATs=0 #RAM_outs=0 tla_usage_type=none
Nonscan rules checking completed, CPU time=0.00 sec.
-----
Begin DRC dependent learning...
Fast-sequential depth results: control=0(0), observe=0(0), detect=0(0), CPU time=0.00 sec
DRC dependent learning completed, CPU time=0.01 sec.
-----
DRC Summary Report
-----
No violations occurred during DRC process.
Design rules checking was successful, total CPU time=0.02 sec.
```

Εικόνα 43: DRC έλεγχος με τα violations να προκύπτουν μηδενικά.

3^ο στάδιο ροής: Στο σημείο αυτό, εκτελούμε την επιλογή ATPG του εργαλείου για να τρέξει ο αλγόριθμος στο σχέδιο. Η κάλυψη φαίνεται στην εικόνα πως φτάνει το **99,99%**, άρα είναι επιτυχής, τα total faults είναι συνολικά **45810**, ενώ τα patterns που δημιουργούνται είναι σε αριθμό **656**. Ύστερα αποθηκεύουμε τα patterns σε αρχείο .stil το οποίο χρησιμοποιείται για την δόμηση του testbench. Στην εντολή για το .stil, τοποθετείται η επιλογή **-serial**, ώστε να γίνει σειριακά στη προσομοίωση ο έλεγχος των patterns. Τέλος γράφουμε σε αρχείο το MAX Testbench, ενώ τρέχουμε την επιλογή simulation που έχει το TetraMAX κατά την οποία τα patterns σαρώνονται και ελέγχονται επιτυχώς.

```

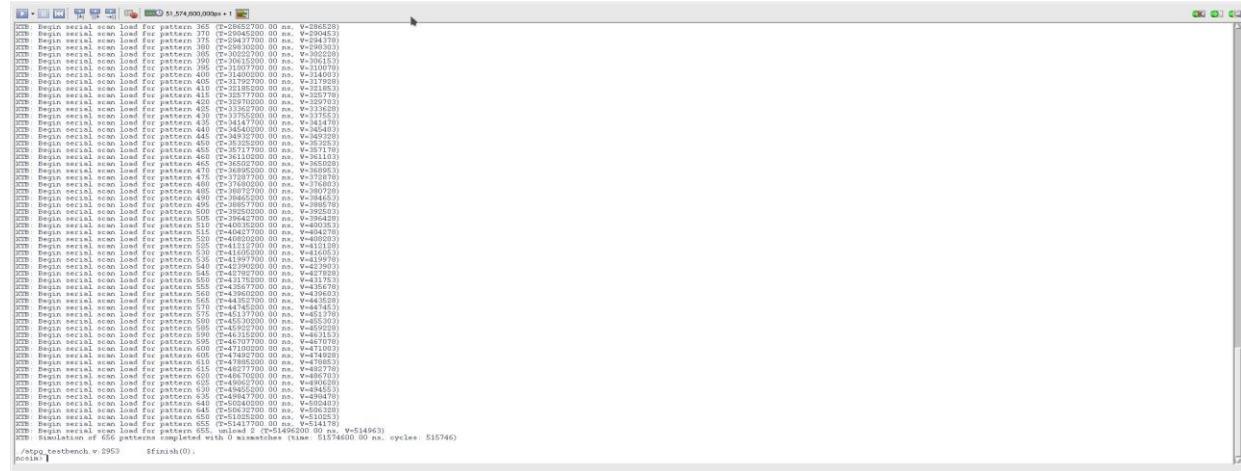
Uncollapsed Stuck Fault Summary Report
-----
fault class          code   #faults
-----
Detected             DT     42001
Possibly detected    PT      0
Undetectable         UD     3805
ATPG untestable      AU      0
Not detected         ND      4
-----
total faults          45810
test coverage        99.99%
-----
Pattern Summary Report
-----
#internal patterns      656
  #basic_scan patterns  656
-----
EST-T> write_patterns atpg_patterns.stil -format stil -serial -internal -replace
Patterns written reference 1972 V statements, generating 515746 test cycles
End writing file 'atpg_patterns.stil' with 656 patterns, File_size = 1503432, CPU_time = 0.1 sec.
EST-T> write_testbench -input /home/inf2015/paanastasiadis/atpg_patterns.stil -output atpg_testbench
Executing 'stil2verilog'...
EST-T> run_simulation
Begin good simulation of 656 internal patterns.
Simulation completed: #patterns=656, #fail_pats=0(0), #failing_meas=0(0), #rejected_pats=0, CPU time=0.03

```

Εικόνα 44: Το τρίτο στάδιο στη ροή του TetraMAX έτσι όπως περιγράφεται παραπάνω.

Προσομοίωση του MAX Testbench στο Incisive:

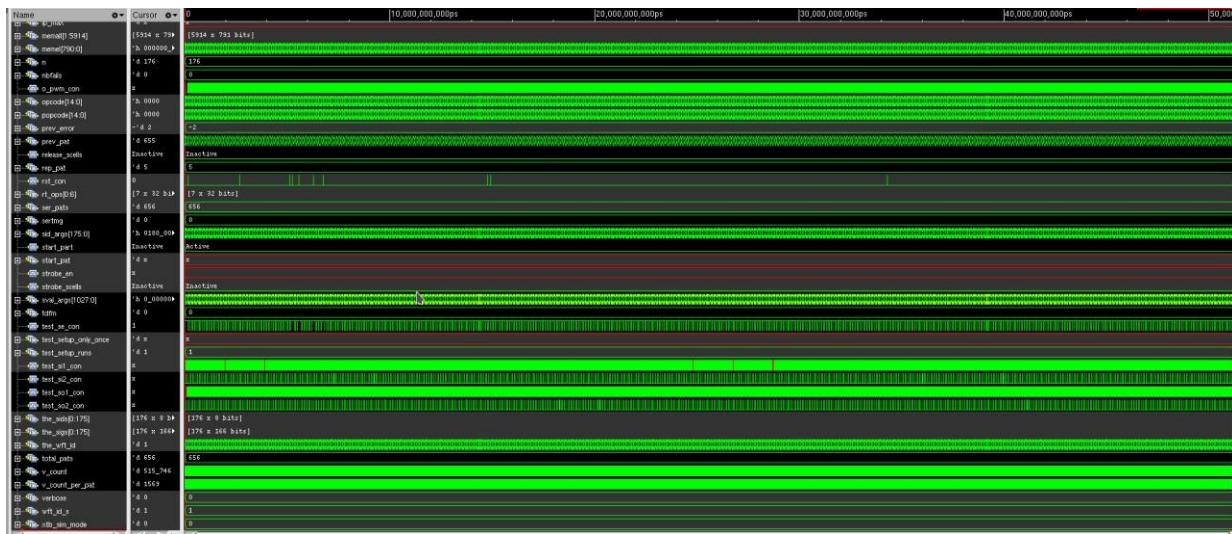
Τελειώνοντας με το πέρασμα του σχεδίου από το TetraMAX και έχοντας εξάγει το αυτόματο MAX Testbench των patterns, εστιάζουμε τη διαδικασία στο Incisive για τον επιπρόσθετο λειτουργικό έλεγχο του DFT. Παρακάτω φαίνεται η σύνοψη των αποτελεσμάτων που εμφανίζονται στη γραμμή εντολών του NCSim, όπου ο σειριακός έλεγχος των 656 patterns ολοκληρώνεται επιτυχώς με 0 mismatches.



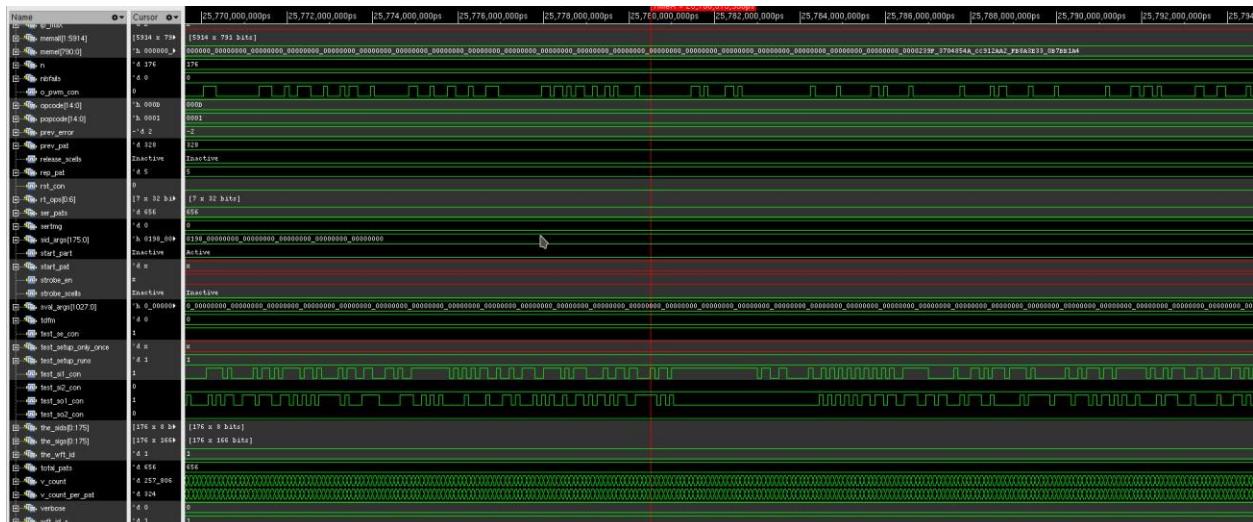
Εικόνα 45: Η επιτυχής ολοκλήρωση της προσομοίωσης των 656 patterns που έγινε σειριακά.

Κυματομορφές του αυτόματου MAX Testbench:

Για λόγους πληρότητας παρουσιάζονται και οι κυματομορφές της προσομοίωσης σε δυο στιγμιότυπα.



Κυματομορφή 1



Κυπατονοφή 2

Φυσική Σχεδίαση

Σκοπός:

Στο τελευταίο στάδιο της ροής του μαθήματος το σχέδιο μεταβιβάζεται στο **INNOVUS** της **Cadence**. Το στάδιο αυτό αφορά τη φυσική σχεδίαση του συστήματος μέσα από το εργαλείο, στην οποία εν ολίγοις γίνονται ενέργειες όπως τοποθέτηση των standard cells, routing και clock tree synthesis. Για τη φυσική σχεδίαση χρησιμοποιείται το αρχείο βιβλιοθήκης της Motorola **LEF**, το οποίο και συμπεριλαμβάνει τις προδιαγραφές και τα στοιχεία των μετάλλων που θα χρησιμοποιηθούν.

Συνοπτικά οι ενέργειες που ακολουθούν:

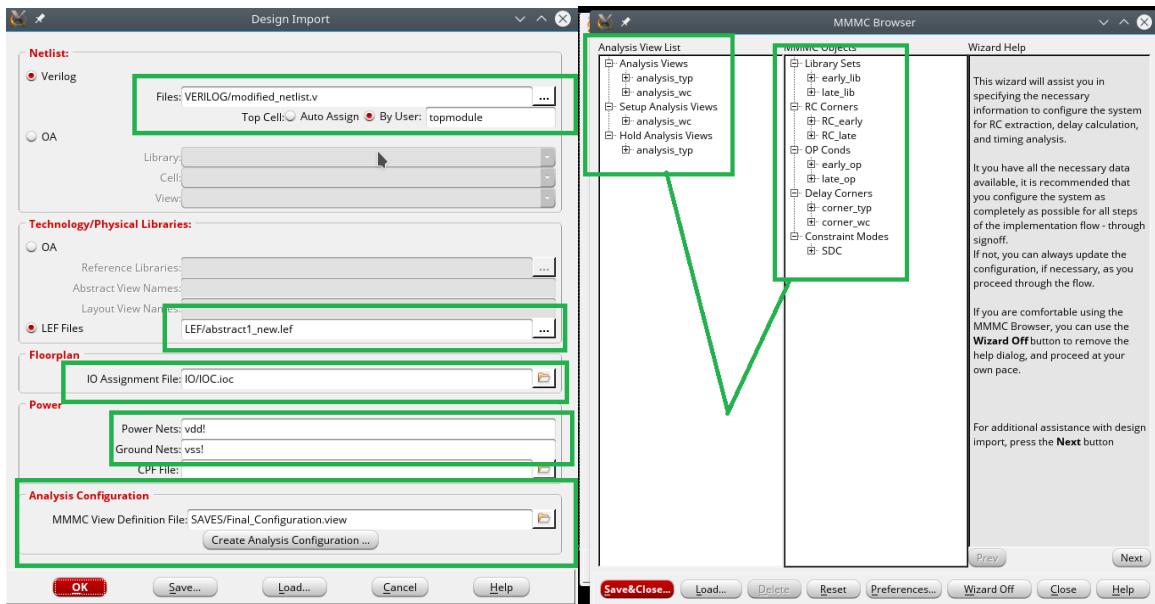
- Εκτέλεση προαπαιτούμενων ενεργειών και φόρτωση των απαιτούμενων αρχείων (LEF, TLF) πριν την εισαγωγή του σχεδίου στο εργαλείο.
- Φόρτωση της Netlist και του SDC στο INNOVUS, αρχεία που έχουν παραχθεί μετά και τη λογική σύνθεση του σχεδίου στα προηγούμενα στάδια του μαθήματος. (Αφού πρώτα προσθέσουμε χειροκίνητα στο αρχείο της netlist τα απαραίτητα I/O Pads της βιβλιοθήκης για να είναι εφικτή η εμφάνιση τους στο εργαλείο).
- Δημιουργία αρχείου επιθεμάτων τάσης, γείωσης και γωνιών (.ios) για τη φόρτωση τους στο εργαλείο και τη τοποθέτηση τους πάνω στο σχέδιο.
- Χωροθέτηση και προσδιορισμός του ωφέλιμου εμβαδού.
- Κλείσιμο πλαισίου με «γέμισμα» των κενών που δημιουργούνται ανάμεσα στα επιθέματα εισόδων και εξόδων, το οποίο γίνεται με τη τοποθέτηση γεμισμάτων – I/O Filler Pads.
- Σχεδίαση και διασύνδεση παροχών ρεύματος (τοποθέτηση Ring, Stripes, Sroute).
- Τοποθέτηση των standard cells μέσα στο πυρήνα του σχεδίου.
- Στατική χρονική ανάλυση και βελτιστοποίηση (Pre-CTS – Πριν τη σύνθεση του δέντρου ρολογιού), ώστε να γίνουν βελτιωτικές αλλαγές στο σχέδιο και να εξαλειφθούν οι χρονικές παραβιάσεις, δηλαδή να υπάρχει θετικό slack χρόνου σε όλα τα critical paths.
- Clock Tree Synthesis – Εισαγωγή δέντρου ρολογιού και αντίστοιχη στατική χρονική ανάλυση πάνω σε αυτή.
- Post – CTS βελτιστοποίηση και στατική χρονική ανάλυση.
- Εκτέλεση Nano Route και στατική χρονική ανάλυση.
- Εισαγωγή core fillers.
- Ενέργειες επαλήθευσης για έλεγχο παραβιάσεων (Verify Connectivity, Verify DRC κ.α.).
- Εξαγωγή αρχείου Netlist και αρχείου καθυστερήσεων SDF και τελική προσομοίωση συμπεριφοράς του σχεδίου στο INCISIVE.

Προετοιμασία του INNOVUS και εισαγωγή του σχεδίου στο εργαλείο:

Σε πρώτο φάση, αφού βρεθούν και τοποθετηθούν σε ανάλογους φακέλους LEF και LIBS τα αρχεία της βιβλιοθήκης της Motorola, επιλέγουμε στο γραφικό περιβάλλον του INNOVUS την επιλογή **Import Design**. Εκεί όπως φαίνεται παρακάτω ορίζουμε τις επιθυμητές ρυθμίσεις στις οποίες θα προσαρμοστεί το σχέδιο.

Ενέργειες κατά το Import Design:

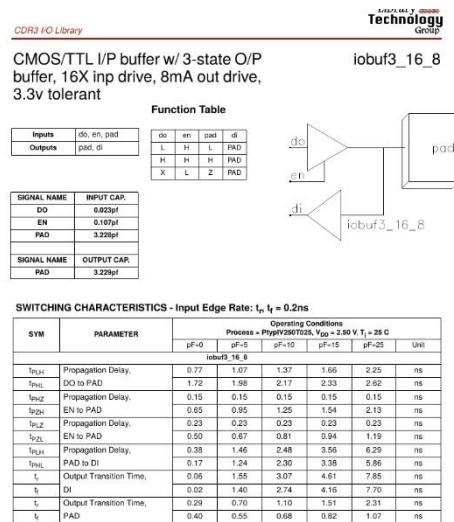
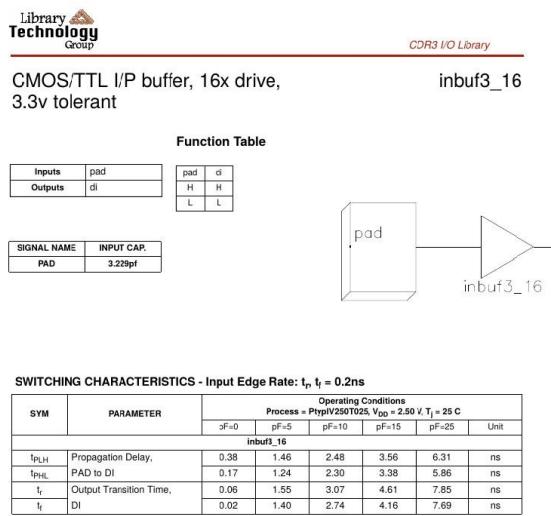
- Επιλέγουμε το αρχείο LEF (Layout Entity File) της βιβλιοθήκης που συμπεριλαμβάνει τις ρυθμίσεις των μετάλλων (π.χ. ελάχιστο πλάτος μετάλλου).
- Ορίζουμε για το Verilog αρχείο φόρτωσης το αρχείο της netlist του σχεδίου (αφού έχουν ενσωματωθεί σε αυτό τα pads για τις εισόδους/εξόδους – ακολουθεί παρακάτω εικόνα).
- Ορίζουμε ως αρχείο των IO ένα αρχείο .ios (περισσότερα παρακάτω), το οποίο περιέχει τη δήλωση και τη τοποθέτηση (N,S,W,E) των επιθεμάτων τάσης/γείωσης – Power Pads καθώς και των Corners για τη φόρτωση αυτών στο σχέδιο.
- Με βάση το αρχείο LEF, προσαρμόζουμε τα Power Nets και τα Ground Nets με τα συμβατά ονόματα τους από το LEF, δηλαδή τα vdd! και vss!.
- Στη δεύτερη εικόνα δεξιά και κατά το Create Analysis Configuration ορίζουμε κατά σειρά τα Library Nets, RC Corners, Op Conds, Delay Corners και το SDC με τα αρχεία .tlf των βιβλιοθηκών της Motorola {cd3_type4.4.tlf (typical), cd3_wc4.4.tlf (worst case)} και το αρχείο SDC που εξάγαμε σε προηγούμενο στάδιο από το Design Compiler.



Εικόνα 46: Η διαδικασία του Import Design με την έναρξη του INNOVUS.

Προσαρμογή της Netlist με την ενσωμάτωση των I/O Pads στο κώδικα.

Στη διαδικασία εισαγωγής των I/O Pads στο κώδικα της Netlist έπρεπε να επιλεχθούν τα κατάλληλα pads εισόδων και εξόδων από τη λίστα αυτών που ορίζει το Datasheet της Motorola. Με βάση το συνολικό ρέυμα που χρειάζεται το κύκλωμα στη λειτουργία του το οποίο ανέρχεται στα **1.8955 mA** (η πληροφορία εξήγηθηκε από την εντολή report_power του Design Compiler, περισσότερα ως προς αυτό παρακάτω) επιλέξαμε τα εξής επιθέματα.



Εικόνα 47: Αριστερά το input pad με όρισμα inbuf3_16 και δεξιά το output pad με όρισμα io buf3_16_8.

Η δόμηση του κώδικα μετά και την ενσωμάτωση των I/O pads στη netlist άλλαξε ως εξής:

```

inbuf3_16 input_zero_0 (.di(input_zero[0]), .pad(input_zero_pad_0[0]));
inbuf3_16 input_zero_1 (.di(input_zero[1]), .pad(input_zero_pad_0[1]));
inbuf3_16 input_zero_2 (.di(input_zero[2]), .pad(input_zero_pad_0[2]));
inbuf3_16 input_zero_3 (.di(input_zero[3]), .pad(input_zero_pad_0[3]));
inbuf3_16 input_zero_4 (.di(input_zero[4]), .pad(input_zero_pad_0[4]));
inbuf3_16 input_zero_5 (.di(input_zero[5]), .pad(input_zero_pad_0[5]));
inbuf3_16 input_zero_6 (.di(input_zero[6]), .pad(input_zero_pad_0[6]));
inbuf3_16 input_zero_7 (.di(input_zero[7]), .pad(input_zero_pad_0[7]));
inbuf3_16 input_zero_8 (.di(input_zero[8]), .pad(input_zero_pad_0[8]));
inbuf3_16 input_zero_9 (.di(input_zero[9]), .pad(input_zero_pad_0[9]));
inbuf3_16 input_zero_10 (.di(input_zero[10]), .pad(input_zero_pad_0[10]));
inbuf3_16 input_zero_11 (.di(input_zero[11]), .pad(input_zero_pad_0[11]));
inbuf3_16 input_zero_12 (.di(input_zero[12]), .pad(input_zero_pad_0[12]));
inbuf3_16 input_zero_13 (.di(input_zero[13]), .pad(input_zero_pad_0[13]));
inbuf3_16 input_zero_14 (.di(input_zero[14]), .pad(input_zero_pad_0[14]));
inbuf3_16 input_zero_15 (.di(input_zero[15]), .pad(input_zero_pad_0[15]));
inbuf3_16 input_zero_16 (.di(input_zero[16]), .pad(input_zero_pad_0[16]));
inbuf3_16 input_zero_17 (.di(input_zero[17]), .pad(input_zero_pad_0[17]));
inbuf3_16 input_zero_18 (.di(input_zero[18]), .pad(input_zero_pad_0[18]));
inbuf3_16 input_zero_19 (.di(input_zero[19]), .pad(input_zero_pad_0[19]));
inbuf3_16 input_zero_20 (.di(input_zero[20]), .pad(input_zero_pad_0[20]));
inbuf3_16 input_zero_21 (.di(input_zero[21]), .pad(input_zero_pad_0[21]));
inbuf3_16 input_zero_22 (.di(input_zero[22]), .pad(input_zero_pad_0[22]));
inbuf3_16 input_zero_23 (.di(input_zero[23]), .pad(input_zero_pad_0[23]));
inbuf3_16 input_zero_24 (.di(input_zero[24]), .pad(input_zero_pad_0[24]));
inbuf3_16 input_zero_25 (.di(input_zero[25]), .pad(input_zero_pad_0[25]));
inbuf3_16 input_zero_26 (.di(input_zero[26]), .pad(input_zero_pad_0[26]));
inbuf3_16 input_zero_27 (.di(input_zero[27]), .pad(input_zero_pad_0[27]));
inbuf3_16 input_zero_28 (.di(input_zero[28]), .pad(input_zero_pad_0[28]));
inbuf3_16 input_zero_29 (.di(input_zero[29]), .pad(input_zero_pad_0[29]));
inbuf3_16 input_zero_30 (.di(input_zero[30]), .pad(input_zero_pad_0[30]));
inbuf3_16 input_zero_31 (.di(input_zero[31]), .pad(input_zero_pad_0[31]));

inbuf3_16 clk_0 (.di(clk), .pad(clk_pad_0));
inbuf3_16 extclk_0 (.di(extclk), .pad(extclk_pad_0));

inbuf3_16 rst_0 (.di(rst), .pad(rst_pad_0));

inbuf3_16 test_si1_0 (.di(test_si1), .pad(test_si1_pad_0));
inbuf3_16 test_si2_0 (.di(test_si2), .pad(test_si2_pad_0));

io buf3_16_8 test_so1_0 (.do(test_so1), .pad(test_so1_pad_0), .en(1'b1));
io buf3_16_8 test_so2_0 (.do(test_so2), .pad(test_so2_pad_0), .en(1'b1));

inbuf3_16 test_se_0 (.di(test_se), .pad(test_se_pad_0));

io buf3_16_8 o_pwm_0 (.do(o_pwm), .pad(o_pwm_pad_0), .en(1'b1));

```

Εικόνα 48: Δήλωση όλων των εισόδων και εξόδων του Top Module του συστήματος με τα προαναφερθέντα επιθέματα της βιβλιοθήκης inbuf3_16 και io buf3_16_8.

Δημιουργία αρχείου IO File (.ioc) :

Για τη δήλωση και τη τοποθέτηση των Power Pads και των Corners πάνω στο σχέδιο στο INNOVUS, δημιουργήσαμε ένα αρχείο .ioc στο οποίο εμπεριέχεται η δήλωση ενός ζευγαριού Vss! / Vdd! αλλά και αυτή των τεσσάρων corners. Επιλέξαμε ένα ζευγάρι Power Pads, καθώς δεδομένου ότι το ένα ζευγάρι μπορεί να τροφοδοτήσει ρεύμα μέχρι τα 30mA , το συνολικό ρεύμα του πυρήνα-σχεδίου που φτάνει τα **1.8995 mA**, δεν ξεπερνά τη παραπάνω τιμή. Το περιεχόμενο του αρχείου έχει την εξής μορφή:

```
Version: 1
Pad: vss_2           W allvss
Pad: vdd_2           W allvdd
Pad: corner1 NW ulcnr_tie
Pad: corner2 SW llcnr_tie
Pad: corner3 NE urcnr_tie
Pad: corner4 SE lrcnr_tie
```

Εικόνα 49: Το περιεχόμενο του IO File. Τα Vdd και Vss επιθέματα θα τοποθετηθούν δυτικά (W = West) του σχεδίου.

Design Compiler, εντολή report_power και συνολικό ρεύμα του κυκλώματος:

Για να υπολογιστεί το συνολικό ρεύμα που απαιτεί το κύκλωμα για τη λειτουργία του, εκτελείται η εντολή report_power στο εργαλείο DC της Synopsys. Προκειμένου όμως το αποτέλεσμα που θα εμφανιστεί να είναι ορθό και ρεαλιστικό, εξάγουμε πρώτα ένα αρχείο SAIF και το φορτώνουμε στο εργαλείο. Το αρχείο SAIF εξάγεται στο INCISIVE της Cadence αφού το κύκλωμα περάσει την ανάλογη προσομοίωση, ώστε να καταγράψει σε αυτό την ισχύ που καταναλώνεται σε μια ρεαλιστική λειτουργία του σχεδίου, δηλαδή το Net Switching Activity. Αφού εξάγομε το αρχείο SAIF και το φορτώσαμε στο Design Compiler τα αποτελέσματα που προέκυψαν είναι τα εξής:

```
Dynamic Power Units = 1mW (derived from V,C,T units)
Leakage Power Units = Unitless

Cell Internal Power = 3.8386 mW (90%)
Net Switching Power = 435.4187 uW (10%)
Total Dynamic Power = 4.2740 mW (100%)

Cell Leakage Power = 0.0000

Error: Either dynamic power or leakage power, in the library, is unitless. Unable to display complete power group summary.

Power Group      Internal Power      Switching Power      Leakage Power      Total Power ( % )      Attrs
-----+-----+-----+-----+-----+-----+
io_pad          0.0000            0.0000            0.0000            NA ( N/A )
memory          0.0000            0.0000            0.0000            NA ( N/A )
black_box        0.0000            0.0000            0.0000            NA ( N/A )
clock_network   0.0000            0.0000            0.0000            NA ( N/A )
register         3.3731          8.7793e- 02          0.0000            NA ( N/A )
sequential       0.0000            0.0000            0.0000            NA ( N/A )
combinational   0.4655            0.3476            0.0000            NA ( N/A )

-----+-----+-----+-----+-----+
Total           3.8386 mW          0.4354 mW          0.0000            NA
1
design_vision>
```

Αποτελέσματα:

Συνολική Ισχύς στα 4.2740 mW.

Τάση στη χειρότερη περίπτωση στα 2.25 V.

Συνολικό ρεύμα στα 1.8995 mA.

Εικόνα 50: Αποτέλεσμα της εντολής report power (συγκεκριμένα report_power -analysis_effort high) κατά την εκτέλεση της στο Design Compiler. Η συνολική ισχύς ανέρχεται στα 4.2740 mW.

Πρώτη μορφή του συστήματος στο INNOVUS:

Με το πέρας των παραπάνω βημάτων, και την κατάλληλη φόρτωση των αρχείων στο INNOVUS το σχέδιο παράγεται στο εργαλείο με τη πρώτη του μορφή όπως φαίνεται παρακάτω:



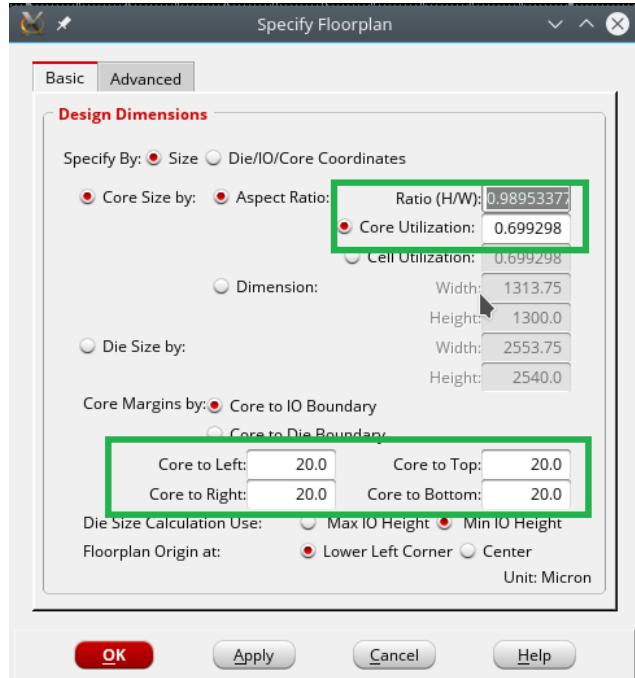
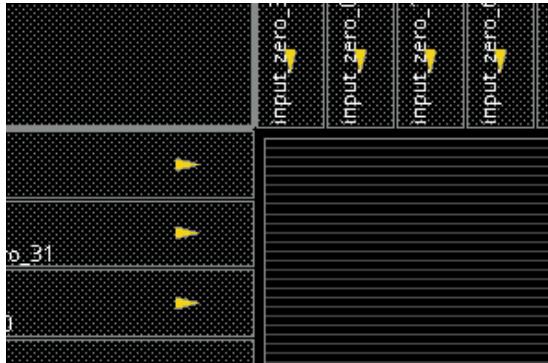
Εικόνα 51: Πρώτη μορφή του σχεδίου στο INNOVUS.

Χωροθέτηση:

Φτάνοντας στο σημείο της χωροθέτησης, με την επιλογή **Specify Floorplan** ορίζουμε τα όρια του πυρήνα και άλλες επιλογές, όπως το Core Utilization και το Ratio (Height / Width). Επιλέγουμε για το λόγο ύψους/πλάτους περίπου το 1 καθώς επιθυμούμε σχήμα τετραγώνου, για το Utilization την τιμή 0.7 και τέλος για τα core boundaries τιμή στα 20, έτσι ώστε να εξαλειφθούν παραβιάσεις κατά την τοποθέτηση των standard cells και να χωρέσει το Core Ring μετέπειτα.

Ακριβείς αλλαγές:

- Ratio (H/W): 0.98953377
- Core Boundaries: 20.0
- Core Utilization: 0.699208



Εικόνα 52: (Αριστερά) Η αλλαγή στο σχέδιο μετά την χωροθέτηση. (Δεξιά) Οι ρυθμίσεις που πραγματοποιήθηκαν κατά το Specify Floorplan στο γραφικό περιβάλλον του INNOVUS.

Εισαγωγή fillers:

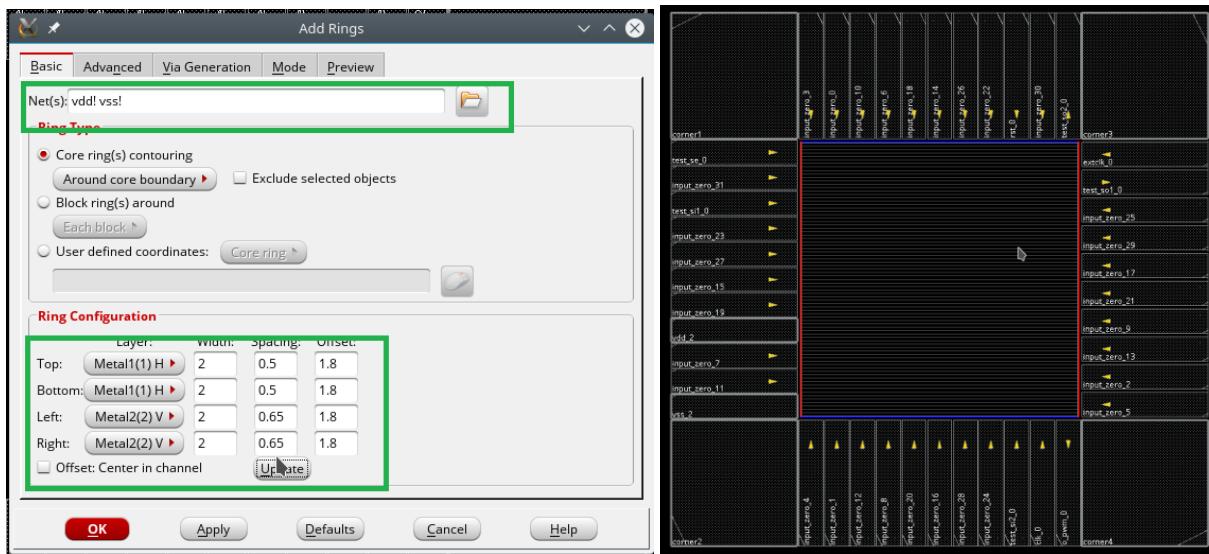
Η εισαγωγή των I/O στο σχέδιο επιφέρει στο εξωτερικό πλαίσιο μερικά κενά, τα οποία χρήζουν «γεμίσματος». Την λειτουργία αυτή πραγματοποιούν τα IO fillers τα οποία ορίζονται στο LEF αρχείο και έχουν μέγεθος 80, 40 ή 20. Στο σύστημα που περιγράφεται και όπως φαίνεται στην προηγούμενη εικόνα οι 40 σε αριθμό είσοδοι και έξοδοι που εισάχθηκαν στο σχέδιο δεν αφήνουν πολλά περιθώρια για τοποθέτηση IO Fillers. Αυτό αποδεικνύεται και με το πέρας της εκτέλεσης της εντολής addIoFiller, η οποία δε κατάφερε να προσθέσει fillers σε κανένα σημείο του σχεδίου.

```
innovus 6> addIoFiller -cell io_fill_80 -prefix IOFILLER_80
**WARN: (IMPFP-127): Incorrect LEF/OA class of cell 'io_fill_80', expected cell with class 'PAD SPACER'.
Added 0 of filler cell 'io_fill_80' on top side.
Added 0 of filler cell 'io_fill_80' on left side.
Added 0 of filler cell 'io_fill_80' on bottom side.
Added 0 of filler cell 'io_fill_80' on right side.
innovus 7> addIoFiller -cell io_fill_40 -prefix IOFILLER_40
**WARN: (IMPFP-127): Incorrect LEF/OA class of cell 'io_fill_40', expected cell with class 'PAD SPACER'.
Added 0 of filler cell 'io_fill_40' on top side.
Added 0 of filler cell 'io_fill_40' on left side.
Added 0 of filler cell 'io_fill_40' on bottom side.
Added 0 of filler cell 'io_fill_40' on right side.
innovus 8> addIoFiller -cell io_fill_20 -prefix IOFILLER_20
**WARN: (IMPFP-127): Incorrect LEF/OA class of cell 'io_fill_20', expected cell with class 'PAD SPACER'.
Added 0 of filler cell 'io_fill_20' on top side.
Added 0 of filler cell 'io_fill_20' on left side.
Added 0 of filler cell 'io_fill_20' on bottom side.
Added 0 of filler cell 'io_fill_20' on right side.
innovus 9>
```

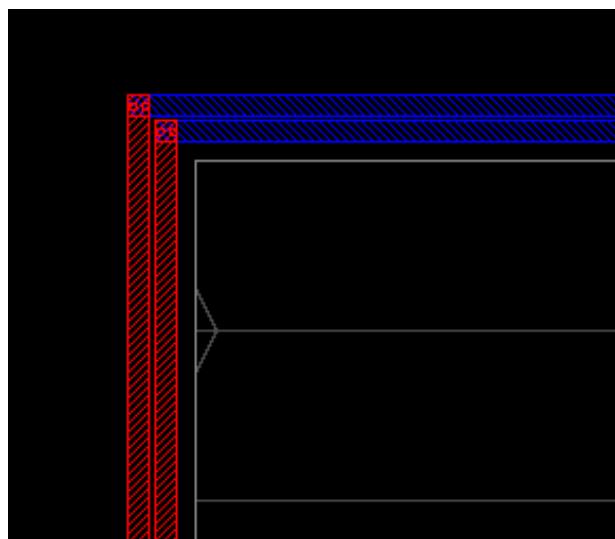
Εικόνα 53: Εντολή addIoFiller για την εισαγωγή των I/O Fillers.

Εισαγωγή δακτυλίου Ring:

Με την επιλογή Add Ring από το γραφικό περιβάλλον στο INNOVUS, αμέσως μετά τη χωροθέτηση, εισάγουμε στο σχέδιο τον δακτύλιο παροχής ρεύματος για το πυρήνα του ολοκληρωμένου. Με το ρεύμα του συνολικού κυκλώματος να ανέρχεται στα 1.89 mA και **με τη παραδοχή ότι για 1 um ανά μα για τα μέταλλα 1 και 2, ορίζουμε το πλάτος – width του ring στα 2 um** (στρογγυλοποίηση προς τα πάνω).



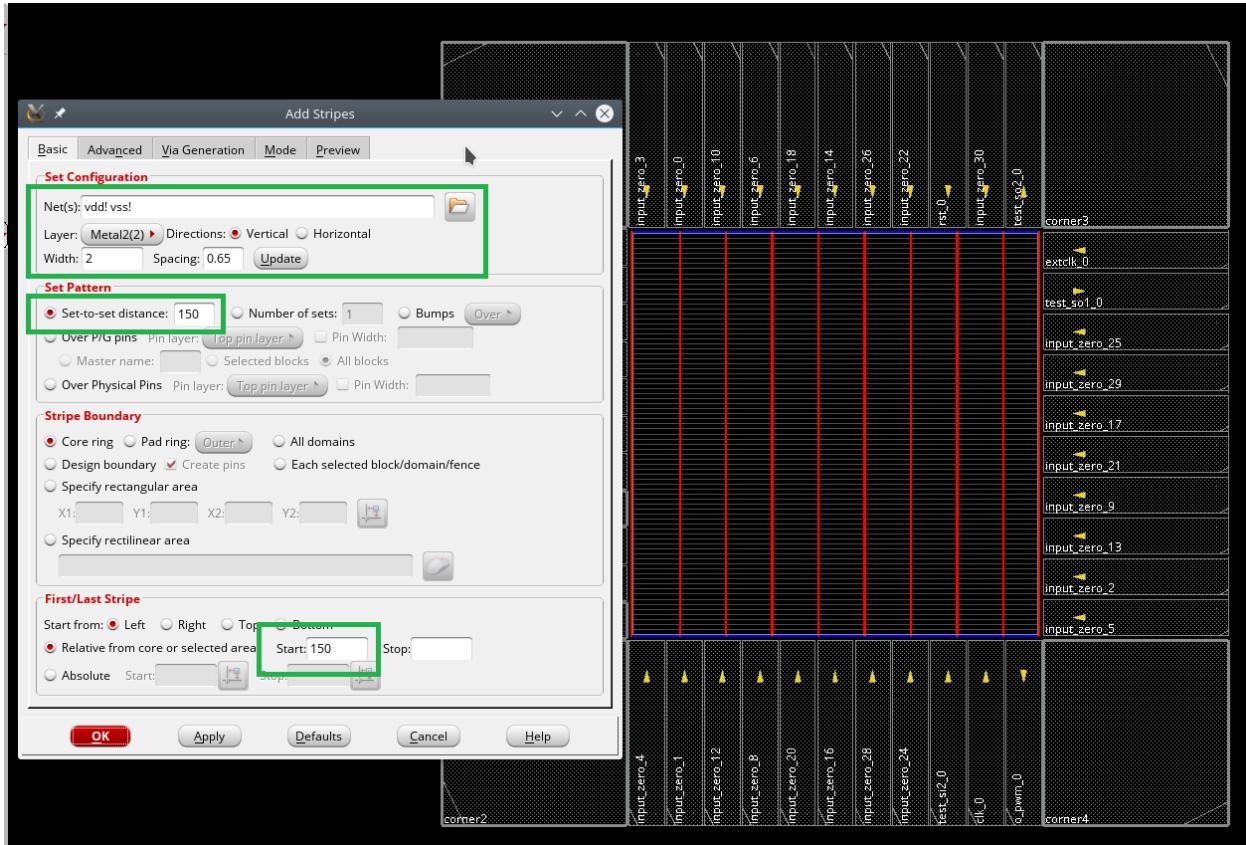
Εικόνα 54: Επιλογή πλάτους για vss και vdd στα 5um στο δακτύλιο του πυρήνα και (δεξιά) το αποτέλεσμα.



Εικόνα 55: Η αριστερή πάνω γωνία στο core ring μετά από μεγέθυνση.

Εισαγωγή Stripes:

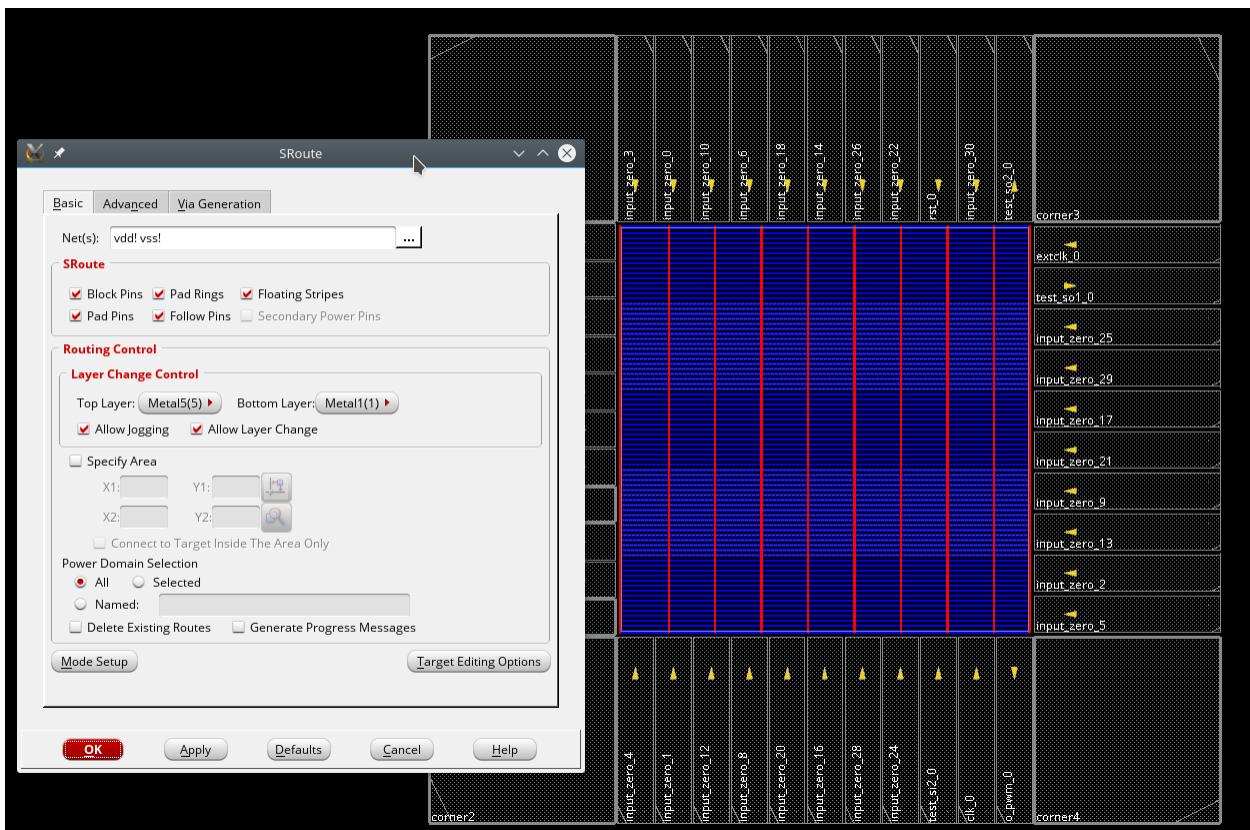
Επόμενο βήμα μετά και την τοποθέτηση του Core Ring, είναι η εισαγωγή των κάθετων παροχών ρεύματος-Stripes στις σειρές των cells. Επιλέγουμε και πάλι **width = 2 um** για το μέταλλο 2 και απόσταση μεταξύ των stripes **150 um**.



Εικόνα 56: Εισαγωγή των stripes και ορισμός των παραμέτρων όπως προαναφέρθηκαν.

Διασύνδεση των συνδέσεων ρεύματος μεταξύ τους:

Σημειώνεται ότι δοκιμάστηκε να εκτελεστεί η εντολή **globalNetConnect**, παρόλα αυτά παρουσιάστηκαν στη συνέχεια violations στα power connections κατά το Verify Connectivity. Χωρίς αυτή, στο τέλος της ροής το Verify Connectivity έδειξε μηδενικές παραβιάσεις. Οπότε βασιζόμαστε στην υπόθεση ότι οι διασυνδέσεις των ρευμάτων πραγματοποιούνται σε αυτή την έκδοση του INNOVUS με την διαδικασία του **Special Route** και όχι με την εντολή **globalNetConnect** την οποία εν τέλει παραλείψαμε.

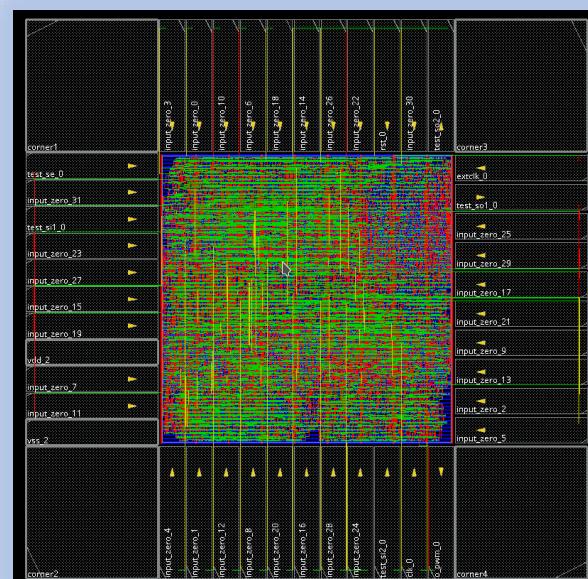


Εικόνα 57: Εφαρμογή Special Route και διασύνδεση των παροχών μεταξύ τους στο σχέδιο.

Διαδικασία τοποθέτησης των standard cells και δήλωση των scan – chains στο INNOVUS:

Την ολοκλήρωση των βημάτων χωροθέτησης και σύνδεσης παροχών ρεύματος ακολουθεί η τοποθέτηση των standard cells στο σχέδιο με την επιλογή **Place Standard Cells** και **Run Full Placement** από το γραφικό περιβάλλον. Με την ενέργεια αυτή εισέρχονται τα cells στο εσωτερικό του πυρήνα και γίνονται ορατές στο GUI του INNOVUS.

Εικόνα 58: Τοποθέτηση των cells στο core του συστήματος.



Για τη δήλωση των scan – chains ώστε αυτές να αναγνωρίζονται από το εργαλείο εκτελέστηκαν οι παρακάτω εντολές:

```
innovus 10> setScanReorderMode -compLogic true
innovus 11> specifyScanChain chain1 -start test_si1_pad_0 -stop test_so1_pad_0
innovus 12> specifyScanChain chain2 -start test_si2_pad_0 -stop test_so2_pad_0
innovus 13>
innovus 13> scanTrace
*** Scan Trace Summary (runtime: cpu: 0:00:00.0 , real: 0:00:00.0):
Successfully traced 2 scan chains (total 1240 scan bits).
*** Scan Sanity Check Summary:
*** 2 scan chains passed sanity check.
```

Εικόνα 59: Εκτέλεση των εντολών για τη δήλωση των scan – chains στο τερματικό του INNOVUS (setScanReorderMode, specifyScanChain, scanTrace).

Στατικές χρονικές αναλύσεις -----

«Από το σημείο αυτό και μετέπειτα, σε κάθε ενέργεια που ακολουθεί (τοποθέτηση standard cells, Pre – CTS Optimization, Clock Tree Synthesis, Post – CTS Optimization, Nano Route) θα πραγματοποιείται στατική χρονική ανάλυση για την εξαγωγή συμπερασμάτων.»

«Η χρονική ανάλυση είναι κάθε φορά δυική και το ένα μέρος αφορά την ανάλυση για το Setup Time και το άλλο για το Hold time.»

«Η εντολή που χρησιμοποιείται είναι η **report_timing -max_path 100** ενώ όταν προηγείται κάποιο στάδιο βελτιστοποίησης ή αυτό του Nano Route χρησιμοποιείται η εντολή **timeDesign (-preCTS ή -postCTS ή -postRoute και -hold όταν πρόκειται για τη hold ανάλυση)**.»-----

Στατική χρονική ανάλυση – 1^η φάση:

Setup time:

```
#####
# Generated by: Cadence Innovus 17.11-s080.1
# OS: Linux x86_64(Host ID linux-e4n3)
# Generated on: Wed May 30 19:47:36 2018
# Design: topmodule
# Command: report_timing -max_path 100 > TIMING_ANALYSIS/report_timing.txt
#####
Path 1: VIOLATED Setup Check with Pin wishbone_master_pid_0/Divide_0/result_reg[23]/ck
Endpoint: wishbone_master_pid_0/Divide_0/result_reg[23]/d (^) checked with
leading edge of 'extclk_pad_0'
Beginpoint: wishbone_master_pid_0/adjusted_dcycle_reg[0]/q (v) triggered by
leading edge of 'clk_pad_0'
Path Groups: {extclk_pad_0}
Analysis View: analysis_wc
Other End Arrival Time      0.000
- Setup                      0.739
+ Phase Shift                25.000
= Required Time              24.261
- Arrival Time               25.014
= Slack Time                 -0.753
    Clock Rise Edge          0.000
    + Clock Network Latency (Ideal) 0.000
    = Beginpoint Arrival Time   0.000
```

Στη πρώτη φάση μετά την τοποθέτηση των standard cells, η στατική χρονική ανάλυση παρουσιάζει setup time παραβιάσεις με αρνητικό slack.

Slack Time = -0.753 ns.

Εικόνα 60: Αποτελέσματα της εντολή **report timing για το **setup**.**

Hold Time:

```
#####
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (4 T). (MEM=1769.54)
*** Calculating scaling factor for early_lib libraries using the default operating condition of each library.
Total number of fetched objects 6279
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1809.73 CPU=0:00:00.5 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1809.73 CPU=0:00:00.7 REAL=0:00:00.0)
Path 1: MET Hold Check with Pin kd_data_reg[7]/ck
Endpoint: kd_data_reg[7]/sdi (v) checked with leading edge of 'clk_pad_0'
Beginpoint: PID0/ki_reg[8]/q (v) triggered by leading edge of 'clk_pad_0'
Path Groups: {clk_pad_0}
Analysis View: analysis_typ
Other End Arrival Time -11.921
+ Hold -0.254
+ Phase Shift 0.000
= Required Time -12.175
Arrival Time -11.598
Slack Time 0.577
Clock Rise Edge 0.000
+ Clock Network Latency (Ideal) -11.921
= Beginpoint Arrival Time -11.921
+
| Instance | Arc | Cell | Delay | Arrival | Required |
|          |      |      |       | Time    | Time     |
| PID0/ki_reg[8] | ck ^ |      |      | -11.921 | -12.497 |
| PID0/ki_reg[8] | ck ^ -> q v | sdffpr_2 | 0.322 | -11.598 | -12.175 |
| kd_data_reg[7] | sdi v | sdffpr_2 | 0.000 | -11.598 | -12.175 |
+-----+
```

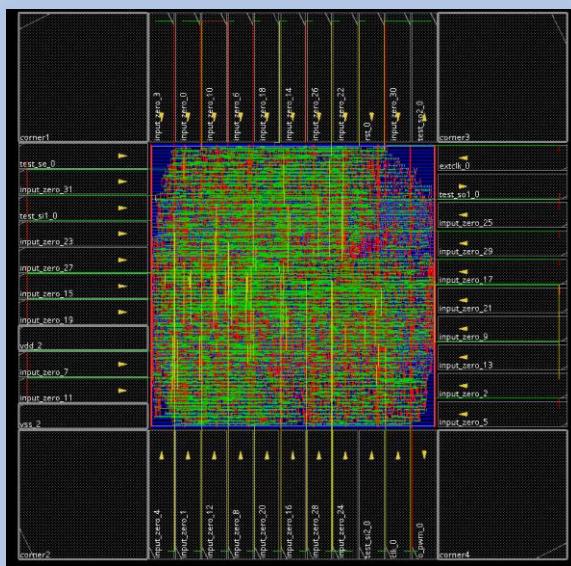
Η πρώτη φάση παρουσιάζει και hold time παραβιάσεις με αρνητικό slack.

Slack Time = - 0.577 ns.

Εικόνα 61: Αποτελέσματα της εντολή report timing για το hold.

Pre – CTS Βελτιστοποίηση:

Η Pre – CTS βελτιστοποίηση κάνει βελτιστοποιήσεις, όπως αναφέρεται και στο όνομα της, πριν το Clock Tree Synthesis. Η πρώτη αλλαγή με το πέρας της βελτιστοποίησης φαίνεται και στη τοποθέτηση των πυλών, αφού ο αλγόριθμος αλλάζει εκεί που κρίνει απαραίτητο τις θέσεις των standard cells. Η νέα μορφή του σχεδίου μετά και την Pre – CTS βελτιστοποίηση έχει ως εξής:



Εικόνα 62: Αναδιάταξη των cells στο σχήμα μετά την Pre – CTS βελτιστοποίηση.

Στατική χρονική ανάλυση – Pre CTS βελτιστοποίηση:

Setup Time:

```
#####
# Generated by: Cadence Innovus 17.11-s080_1
# OS: Linux x86_64(Host ID linux-e4n3)
# Generated on: Wed May 30 19:53:39 2018
# Design: topmodule
# Command: report_timing -max_path 100 > TIMING_ANALYSIS/report_timing.txt
#####
Path 1: MET Setup Check with Pin wishbone_master_pid_0/Divide_0/result_reg[23]/ck
Endpoint: wishbone_master_pid_0/Divide_0/result_reg[23]/d (^) checked with
leading edge of 'extclk_pad_0'
Beginpoint: wishbone_master_pid_0/adjusted_dcycle_reg[0]/q (^) triggered by
leading edge of 'clk_pad_0'
Path Groups: {extclk_pad_0}
Analysis View: analysis_wc
Other End Arrival Time 0.000
- Setup 0.844
+ Phase Shift 25.000
= Required Time 24.156
- Arrival Time 24.117
= Slack Time 0.039
    Clock Rise Edge 0.000
        + Clock Network Latency (Ideal) 0.000
        = Beginpoint Arrival Time 0.000
```

Με το πέρας της Pre – CTS βελτιστοποίησης το αρνητικό slack έγινε θετικό και εκμηδενίστηκαν οι setup παραβιάσεις.

Setup Time = 0.039 ns.

**Εικόνα 63: Report timing για το setup μετά τη Pre-CTS
Βελτιστοποίηση.**

Hold Time:

```
-----  
timeDesign Summary  
-----  
  
Hold views included:  
analysis_typ  
-----  
|  
| Hold mode | all | reg2reg | default |  
|  
| WNS (ns): | 0.433 | 0.577 | 0.433 |  
| TNS (ns): | 0.000 | 0.000 | 0.000 |  
| Violating Paths: | 0 | 0 | 0 |  
| All Paths: | 3719 | 2477 | 1402 |  
|  
Density: 62.485%
```

Κατά την βελτιστοποίηση Pre – CTS, διορθώθηκαν και οι hold παραβιάσεις.

Hold Time = 0.433 ns.

Εικόνα 64: Εντολή timeDesign και τα αποτελέσματα για hold μετά τη Βελτιστοποίηση.

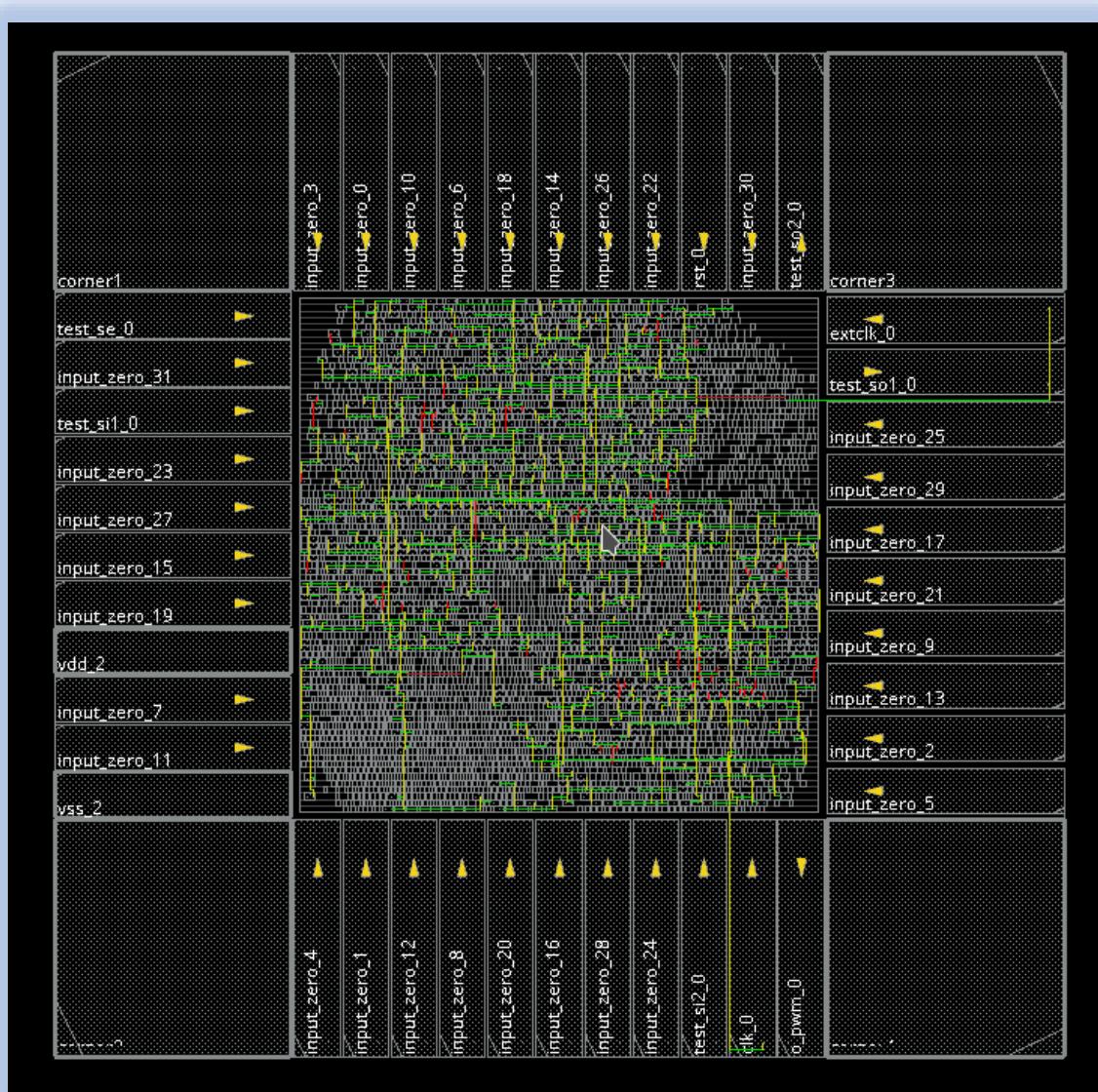
«Με την εντολή time Design σημειώνεται ότι υπολογίζεται και το Density (Πυκνότητα) του σχεδίου. Στο στάδιο αυτό βρίσκεται στο 62.485%.

Clock Tree Synthesis:

Το Clock Tree Synthesis ακολουθεί αμέσως μετά τη Pre – CTS Βελτιστοποίηση. Το ρολόι από ιδανικό γίνεται πια πραγματικό και αυτό επιτυγχάνεται με την εισαγωγή του δέντρου ρολογιού στο σχέδιο. Αυτό σημαίνει ότι η σύνδεση του δέντρου ρολογιού με το σύστημα επιφέρει ανακατατάξεις και περαιτέρω καθυστερήσεις στο δομή.

Η ενσωμάτωση του δέντρου πραγματοποιείται με την εκτέλεση της εντολής **ccopt_design** στο τερματικό του INNOVUS.

Στη παρακάτω εικόνα επιτυγχάνεται μετά από επιλογή από το γραφικό περιβάλλον, να φαίνεται το path που δημιουργεί το δέντρο ρολογιού στο πυρήνα.



Εικόνα 65: Το δέντρο ρολογιού στο πυρήνα του σχεδίου.

Στατική χρονική ανάλυση – Clock Tree Synthesis:

Setup Time:

```
#####
# Generated by: Cadence Innovus 17.11-s080_1
# OS: Linux x86_64(Host ID linux-e4n3)
# Generated on: Wed May 30 20:00:57 2018
# Design: topmodule
# Command: report_timing -max_path 100 > TIMING_ANALYSIS/report_timing.txt
#####
Path 1: MET Setup Check with Pin wishbone_master_pid_0/Divide_0/result_reg[23]/d
ck
Endpoint: wishbone_master_pid_0/Divide_0/result_reg[23]/d (^) checked with
leading edge of 'extclk_pad_0'
Beginpoint: wishbone_master_pid_0/adjusted_dcycle_reg[0]/q (v) triggered by
leading edge of 'clk_pad_0'
Path Groups: {extclk_pad_0}
Analysis View: analysis_wc
Other End Arrival Time -11.873
- Setup 0.858
+ Phase Shift 25.000
= Required Time 12.269
- Arrival Time 11.840
= Slack Time 0.429
    Clock Rise Edge 0.000
    + Clock Network Latency (Ideal) -12.296
    = Beginpoint Arrival Time -12.296
```

Μετά και την εκτέλεση του ccopt -design και το Clock Tree Synthesis, το slack time για το Setup διαφοροποιήθηκε αλλά έμεινε θετικό.

Slack Time = 0.429 ns.

Εικόνα 66: Report timing για το setup μετά το Clock Tree Synthesis.

Hold Time:

```
#####
# Generated by: Cadence Innovus 17.11-s080_1
# OS: Linux x86_64(Host ID linux-e4n3)
# Generated on: Wed May 30 20:02:30 2018
# Design: topmodule
# Command: report_timing -check_type hold
#####
# Design Stage: PreRoute
# Design Name: topmodule
# Design Mode: 90nm
# Analysis Mode: MMC Non-OCV
# Parasitics Mode: No SPEF/RCDB
# Signoff Settings: SI Off
#####
Calculate delays in BcWc mode...
Start delay calculation (fullDC) (4 T). (MEM=1769.54)
*** Calculating scaling factor for early_lib libraries using the default operating condition of each library.
Total number of fetched objects 6279
AAE_INFO: Total number of nets for which stage creation was skipped for all views 0
End delay calculation. (MEM=1809.73 CPU=0:00:00.5 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1809.73 CPU=0:00:00.7 REAL=0:00:00.0)
Path 1: MET Hold Check with Pin kd_data_reg[7]/ck
Endpoint: kd_data_reg[7]/sdi (v) checked with leading edge of 'clk_pad_0'
Beginpoint: PID0/k1_reg[8]/q (v) triggered by leading edge of 'clk_pad_0'
Path Groups: {clk_pad_0}
Analysis View: analysis_typ
Other End Arrival Time -11.921
+ Hold -0.254
+ Phase Shift 0.000
= Required Time -12.175
Arrival Time -11.598
Slack Time 0.577
    Clock Rise Edge 0.000
    + Clock Network Latency (Ideal) -11.921
    = Beginpoint Arrival Time -11.921
```

Το ccopt -design άλλαξε και το hold slack time το οποίο όμως παρέμεινε και πάλι θετικό.

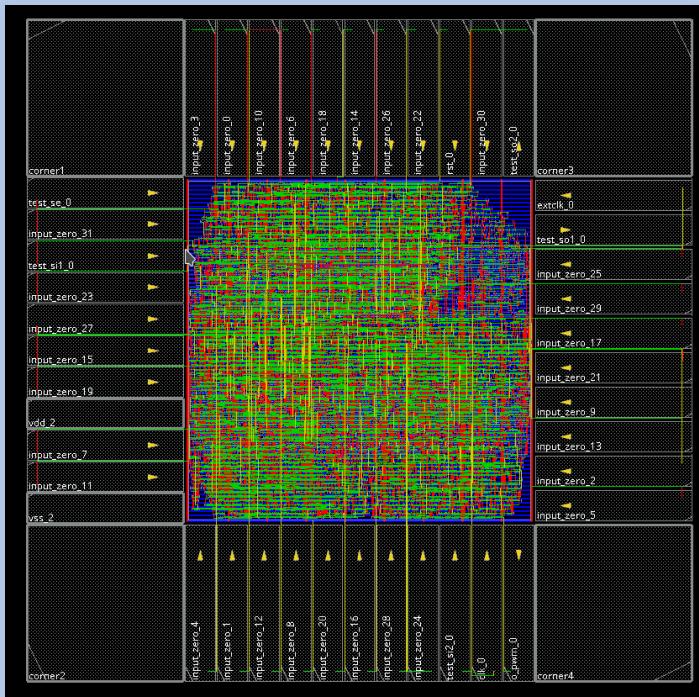
Slack Time = 0.577 ns.

Εικόνα 67: Report timing για το hold μετά το Clock Tree Synthesis.

Post – CTS Βελτιστοποίηση:

Μετά και την εισαγωγή του δέντρου ρολογιού, υπάρχει δυνατότητα να πραγματοποιηθεί ακόμα ενός είδους βελτιστοποίηση, αυτή του Post – CTS (μετά δηλαδή της πραγματοποίησης του Clock Tree Synthesis).Η βελτιστοποίηση πραγματοποιήθηκε και οι αλλαγές είναι οι εξής:

Η νέα μορφή του σχεδίου δεν επιδέχεται σημαντικές αλλαγές και ο πυρήνας με τα cells μένει περίπου ίδιος.



Εικόνα 68: Η μορφή του σχεδίου μετά το Post - CTS Optimization.

Στατική χρονική ανάλυση – Post CTS Βελτιστοποίηση:

Setup Time:

```
#####
# Generated by: Cadence Innovus 17.11-s080_1
# OS: Linux x86_64(Host ID linux-e4n3)
# Generated on: Wed May 30 20:05:17 2018
# Design: topmodule
# Command: report_timing -max_path 100 > TIMING_ANALYSIS/report_timing.txt
#####
Path 1: MET Setup Check with Pin wishbone_master_pid_0/Divide_0/result_reg[22]/
ck
Endpoint: wishbone_master_pid_0/Divide_0/result_reg[22]/d (^) checked with
leading edge of 'extclk_pad_0'
Beginpoint: wishbone_master_pid_0/adjusted_dcycle_reg[0]/q (v) triggered by
leading edge of 'clk_pad_0'
Path Groups: {extclk_pad_0}
Analysis View: analysis_wc
Other End Arrival Time      -11.873
- Setup                      0.904
+ Phase Shift                25.000
= Required Time              12.223
- Arrival Time               11.750
= Slack Time                 0.473
  Clock Rise Edge            0.000
  + Clock Network Latency (Ideal) -12.296
  = Beginpoint Arrival Time   -12.296
```

Η βελτιστοποίηση δεν δημιούργησε νέες παραβιάσεις και το slack παρέμεινε θετικό. Παρόλα αυτά **αυξήθηκε** ελάχιστα κατά **0.044 ns**.

Slack Time = 0.473 ns.

Εικόνα 69: Report timing για το setup μετά τη Post-CTS Βελτιστοποίηση.

Hold Time:

```
timeDesign Summary

Hold views included:
analysis_typ

+-----+-----+-----+
| Hold mode | all | reg2reg | default |
+-----+-----+-----+
| WNS (ns): | 0.299 | 0.516 | 0.299 |
| TNS (ns): | 0.000 | 0.000 | 0.000 |
| Violating Paths: | 0 | 0 | 0 |
| All Paths: | 3719 | 2477 | 1402 |
+-----+-----+-----+

Density: 61.534%
Routing Overflow: 0.00% H and 0.00% V

Reported timing to dir ./timingReports
Total CPU time: 1.41 sec
Total Real time: 1.0 sec
Total Memory Usage: 1503.972656 Mbytes
innovus 112> innovus 112>
```

To hold time παρέμεινε και αυτό θετικά ενώ **μειώθηκε** κατά **0.278 ns.**

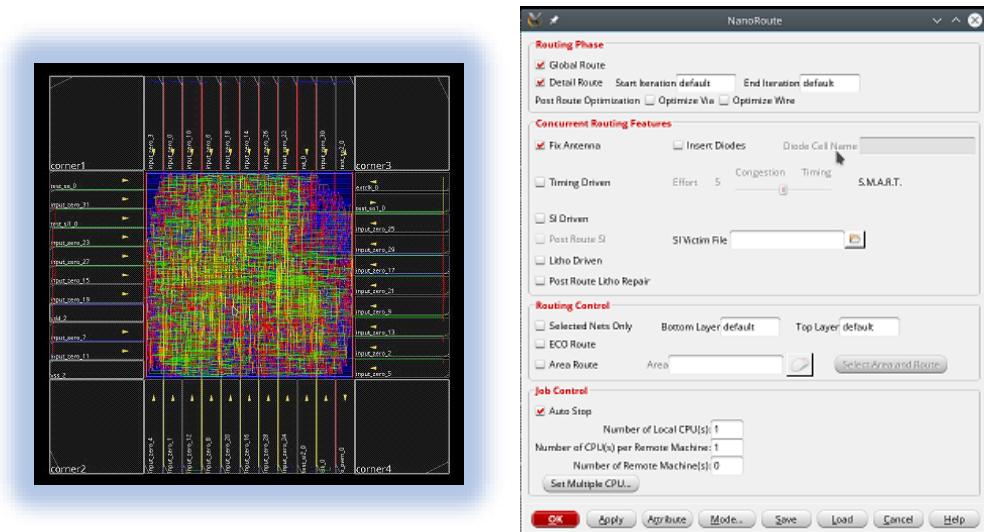
Slack Time = 0.299 ns.

Εικόνα 70: timeDesign για το hold μετά τη Post-CTS
Βελτιστοποίηση.

«Η πυκνότητα μετά τη Post - CTS βελτιστοποίηση βρίσκεται στο 61.534%»

Nano – Routing:

Το Nano Routing αποτελεί το τελικό σχεδιαστικό βήμα. Σε αυτό πραγματοποιείται η οριστική διασύνδεση και αυτό πραγματοποιείται με τον αντίστοιχο αλγόριθμο διασύνδεσης. Εκτελούμε το Nano Route από το GUI του INNOVUS.



Εικόνα 71: (Αριστερά) Η μορφή του σχεδίου μετά και το Nano Route και (Δεξιά) το παράθυρο για την εκτέλεση του από το INNOVUS.

Στατική χρονική ανάλυση – Nano Routing:

Setup Time:

```
-----  
          timeDesign Summary  
-----  
  
Setup views included:  
  analysis_wc  
  
-----  
      Setup mode | all | reg2reg | default |  
-----  
      WNS (ns): | 0.158 | 0.158 | 21.187 |  
      TNS (ns): | 0.000 | 0.000 | 0.000 |  
 Violating Paths: | 0 | 0 | 0 |  
   All Paths: | 3719 | 2477 | 1402 |  
-----  
  
-----  
      DRVs | Real | Total |  
-----  
      | Nr nets(terms) | Worst Vio | Nr nets(terms) |  
-----  
 max_cap | 0 (0) | 0.000 | 0 (0) |  
 max_tran | 0 (0) | 0.000 | 35 (35) |  
 max_fanout | 88 (88) | -43 | 109 (109) |  
 max_length | 0 (0) | 0 | 0 (0) |  
-----  
  
Density: 61.534%  
        (99.592% with Fillers)  
-----  
Reported timing to dir ./timingReports  
Total CPU time: 1.53 sec  
Total Real time: 1.0 sec  
Total Memory Usage: 1389.613281 Mbytes  
innovus 14> innovus 14> █
```

To Nano Routing επέφερε μια μείωση στο slack του setup time και συγκεκριμένα κατά **0.315 ns**.

Slack Time = **0.158 ns**.

Εικόνα 72: Χρονική ανάλυση με την εντολή timeDesign για το setup μετά το Nano Route.

Hold Time:

```
-----  
          timeDesign Summary  
-----  
  
Hold views included:  
  analysis_typ  
  
-----  
      Hold mode | all | reg2reg | default |  
-----  
      WNS (ns): | 0.299 | 0.516 | 0.299 |  
      TNS (ns): | 0.000 | 0.000 | 0.000 |  
 Violating Paths: | 0 | 0 | 0 |  
   All Paths: | 3719 | 2477 | 1402 |  
-----  
  
Density: 61.534%  
        (99.592% with Fillers)  
-----  
Reported timing to dir ./timingReports  
Total CPU time: 3.05 sec  
Total Real time: 2.0 sec  
Total Memory Usage: 1351.789062 Mbytes  
innovus 15> innovus 15> █
```

Με το Nano Routing το hold slack δεν υπέστη αλλαγή και έμεινε απαράλλαχτο.

Slack Time = **0.299 ns**.

Εικόνα 73: Εντολή timeDesign για το hold μετά το Nano Routing.

Post-Route Βελτιστοποίηση – Δε πραγματοποιήθηκε!

Μετά το Nano Route από το INNOVUS δίνεται η δυνατότητα το σχέδιο να επιδεχθεί άλλη μια βελτιστοποίηση, αυτή του Post Route. Κάνοντας την, το slack time στο hold παραβιάζεται, και παίρνει την τιμή των **-11.622 ns**, ενώ το slack time του setup αυξάνεται και παίρνει την τιμή των **12.346 ns**. Με την πιθανότητα να υπάρχει πρόβλημα στον router του INNOVUS η Post – Route Βελτιστοποίηση **αγνοήθηκε**, μιας και τόσο το setup όσο και το hold ήταν θετικά στο προηγούμενο στάδιο, όποτε δεν υπήρχε ανάγκη για περαιτέρω βελτιστοποίηση.

Εισαγωγή core fillers:

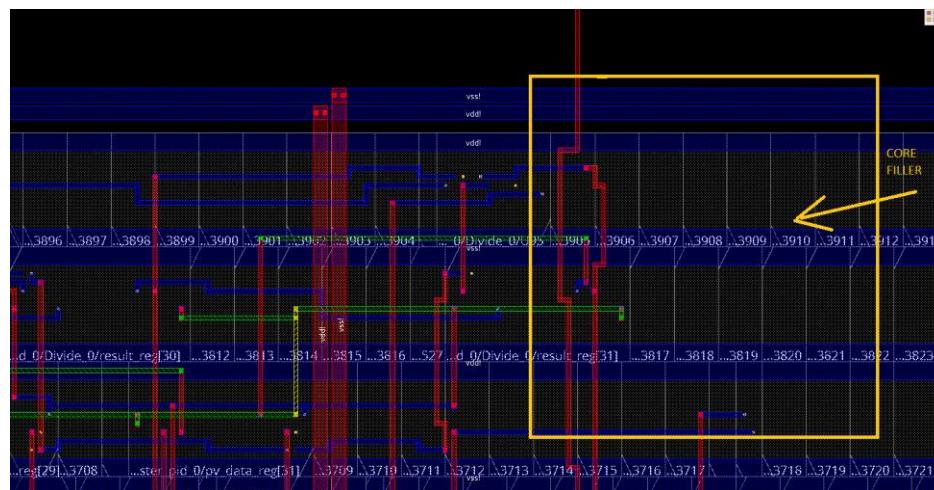
Στο τελείωμα του σχεδίου, χρησιμοποιούμε την εντολή που εισάγει **core fillers**, όπου ο άδειος χώρος εσωτερικά του πυρήνα πρέπει να γεμίσει με γεμίσματα τα οποία περιέχουν επαφές στο υπόστρωμα, έτσι ώστε να μην δημιουργηθούν παρασιτικά μονοπάτια κατά μήκος του υποστρώματος σε κενά τμήματα.

```
addfiller -cell corefill_6 -prefix COREFILLER6  
addfiller -cell corefill_5 -prefix COREFILLER5  
addfiller -cell corefill_4 -prefix COREFILLER4  
addfiller -cell corefill_3 -prefix COREFILLER3  
addfiller -cell corefill_2 -prefix COREFILLER2  
addfiller -cell corefill_1 -prefix COREFILLER1
```

```
Density: 61.534%  
(99.592% with Fillers)  
-----  
Reported timing to dir ./timingReports  
Total CPU time: 1.53 sec  
Total Real time: 1.0 sec  
Total Memory Usage: 1389.613281 Mbytes  
innovus 14> innovus 14> ■
```

Εικόνα 74: (Αριστερά) Η εντολή που εισάγει τα core fillers.(Δεξιά) Το νέο Density μαζί με τα fillers που φθάνει το 99.592%

Αφού οι εντολές παραπάνω εκτελεστούν, ο πυρήνας του σχεδίου έχει συμπληρωμένο κάθε κενό στο εσωτερικό του με ένα core filler. Ως συνέπεια αυτού, το Density (μαζί με τα fillers) πλησιάζει το 100% (**99.592%**).



Τελική στατική χρονική ανάλυση:

Μετά το Nano Routing αλλά και το τελείωμα του σχεδίου με την ενσωμάτωση των core fillers, το σύστημα μπορεί πλέον να δεχθεί την τελική του στατική χρονική ανάλυση. Εκτελούμε την εντολή **timeDesign** τόσο για ανάλυση setup όσο και ανάλυση hold. Όπως φαίνεται και αμέσως παρακάτω τα αποτελέσματα στο slack time και στις δυο περιπτώσεις δεν διαφοροποιήθηκαν μετά το Nano Routing, οπότε οι τελικές τιμές είναι οι εξής:

- Slack = 0.158 ns για το setup time.
- Slack = 0 .299 ns για το hold time.

Όσον αφορά και τη τελική πυκνότητα του πυρήνα του σχεδίου, οι τιμές που προκύπτουν είναι οι ακόλουθες:

- Density = 61.534%
- Density = 99.592 % (μαζί με τα core fillers).

The screenshot shows two terminal windows side-by-side. Both windows have a header "timeDesign Summary".

Left Terminal (Setup mode):

```

timeDesign Summary

Setup views included:
analysis_wc

+-----+
| Setup mode | all | reg2reg | default |
+-----+
| WNS (ns): | 0.158 | 0.158 | 21.187 |
| INS (ns): | 0.000 | 0.000 | 0.000 |
| Violating Paths: | 0 | 0 | 0 |
| All Paths: | 3719 | 2477 | 1402 |
+-----+

```

Right Terminal (Hold mode):

```

timeDesign Summary

Hold views included:
analysis_typ

+-----+
| Hold mode | all | reg2reg | default |
+-----+
| WNS (ns): | 0.299 | 0.516 | 0.299 |
| INS (ns): | 0.000 | 0.000 | 0.000 |
| Violating Paths: | 0 | 0 | 0 |
| All Paths: | 3719 | 2477 | 1402 |
+-----+

```

Density Information:

Left window: Density: 61.534% (99.592% with Fillers)

Right window: Density: 61.534% (99.592% with Fillers)

Reported timing information at the bottom of both windows:

```

Reported timing to dir ./timingReports
Total CPU time: 1.53 sec
Total Real time: 1.0 sec
Total Memory Usage: 1389.613281 Mbytes
innovus 14> innovus 14>

```

```

Reported timing to dir ./timingReports
Total CPU time: 3.05 sec
Total Real time: 2.0 sec
Total Memory Usage: 1351.789062 Mbytes
innovus 15> innovus 15>

```

Εικόνα 76: timeDesign αριστερά για το setup και δεξιά για το hold.

Επαληθεύσεις (Verify Geometry, DRC, Connectivity, AC Limit):

Με την ολοκλήρωση του σχεδίου αλλά και σε οποιοδήποτε προηγούμενο στάδιο αν αυτό είναι επιθυμητό, υπάρχει η δυνατότητα από το γραφικό περιβάλλον του INNOVUS να εκτελεστεί ένας αριθμός επαληθεύσεων για διάφορες λειτουργίες του σχεδίου.

Οι επαληθεύσεις που αφορούσαν το **Connectivity**, το **Geometry** και το **AC Limit** παρήγαγαν αποτελέσματα που επιδεικνύουν **μηδενικά** σφάλματα πάνω στο σχέδιο.

```

Num Violations: 0

End Time: Sat Jun 16 18:56:52 2018
***** End: verifyACLimit *****
(CPU Time: 0:00:06.6 MEM: 246.906M)

```

Εικόνα 77: Verify AC Limit.

```

Begin Summary ...
  Cells      : 0
  SameNet    : 0
  Wiring     : 0
  Antenna   : 0
  Short      : 0
  Overlap    : 0
End Summary

Verification Complete : 0 Viols. 0 Wrngs.

*****End: VERIFY GEOMETRY*****
*** verify geometry (CPU: 0:00:02.1 MEM: 0.0M)

```

Εικόνα 78: Verify Geometry.

```

***** Start: VERIFY CONNECTIVITY *****
Start Time: Sat Jun 16 18:51:49 2018

Design Name: topmodule
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (2553.7500, 2540.0000)
Error Limit = 1000; Warning Limit = 50
Check all nets
Use 4 pthreads

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Sat Jun 16 18:51:49 2018
Time Elapsed: 0:00:00.0

***** End: VERIFY CONNECTIVITY *****
  Verification Complete : 0 Viols. 0 Wrngs.
  (CPU Time: 0:00:00.4 MEM: 24.012M)

```

Εικόνα 79: Verify Connectivity.

Μια ακόμα επαλήθευση που εκτελέστηκε ήταν αυτή του **Verify DRC**. Αντίθετα με τους προηγούμενους ελέγχους, το Verify DRC έδειξε έναν μεγάλο αριθμό από σφάλματα που όμως όπως τονίστηκε και κατά την εισαγωγή της τεχνικής αναφοράς, αφορούν το αρχείο LEF και τη κακή δόμηση του. Τα σφάλματα αυτά αφορούν κυρίως διαφορετικές αποστάσεις από αυτές που περιμένει το INNOVUS, που υφίστανται βασιζόμενες στο LEF και αφορούν Vias , Wires και αλλά σημεία του σχεδίου.

```

Verification Complete : 1051 Viols.

*** End Verify DRC (CPU: 0:00:01.2 ELAPSED TIME: 1.00 MEM: 8.0M) ***

Regular Wire of Net P1D0/mr[11]
Actual: 1.110000 Required: 1.210000 Type: Minimum Area
bbox = (1028.450, 1327.200) (1030.300, 1327.800)

```

Εικόνα 80: (Πάνω) 1051 Violations από το Verify DRC. (Κάτω) Παράδειγμα διαφορετικών αποστάσεων σε Wire από αυτές που αναμένονται σύμφωνα με τον Violation Browser του INNOVUS.

Τελική ανάλυση ισχύος:

Μαζί με την στατική χρονική ανάλυση πραγματοποιήθηκε ανάλυση ισχύος με την ολοκλήρωση του σχεδίου. Για την εξαγωγή των αποτελέσμάτων εκτελέστηκε η εντολή report_power που εμφανίζει αποτελέσματα για το Internal, Switching, Leakage αλλά και Total Power του συστήματος.

Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)
Sequential	10.33	0.8934	0	11.22	51.94
Macro	0	0	0	0	0
IO	0	0.359	0	0.359	1.662
Combinational	4.404	2.948	0	7.372	34.13
Clock (Combinational)	0.4125	2.236	0	2.649	12.26
Clock (Sequential)	0	0	0	0	0
Total	15.14	6.457	0	21.6	100

Εικόνα 81: Αποτελέσματα εντολής report_power.

Τα αποτελέσματα που προκύπτουν είναι τα εξής:

- **Internal Power 15.14 mw.**
- **Switching Power 6.457 mw.**
- **Leakage Power 0 mw.**
- **Total Power = 21.6 mw.**

Ως προς την ισχύ του συστήματος, το GUI του INNOVUS διαθέτει την επιλογή report-> power histograms την οποία είναι δυνατή η παραγωγή ιστογραμμάτων για την ισχύ της κάθε μονάδας ξεχωριστά.



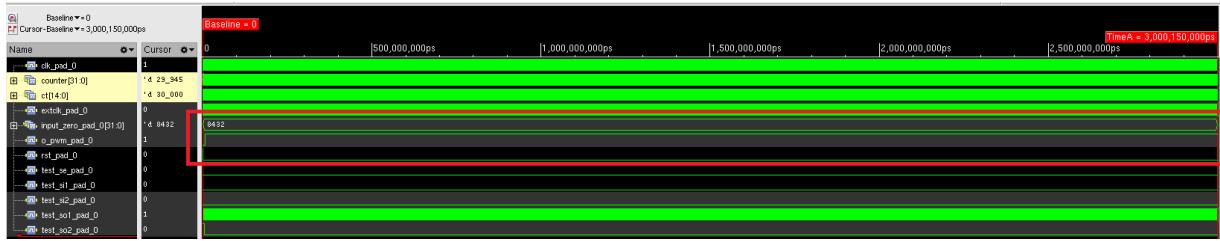
Εικόνα 82: Ιστογράμματα ισχύος της κάθε μονάδας.

Τελική Προσομοίωση συμπεριφοράς:

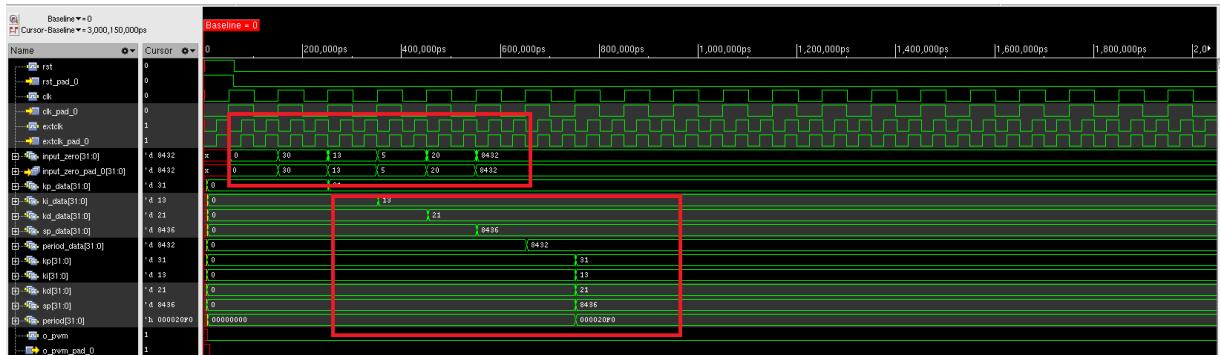
Ολοκληρώνοντας το σχέδιο στο INNOVUS, τελικό βήμα αποτελεί η προσομοίωση συμπεριφοράς της Netlist για να διαπιστωθεί αν το σύστημα μετά τη διοχέτευση του στο εργαλείο και κατά επέκταση τη φυσική του σχεδίαση, λειτουργεί ορθά. Οπότε τελευταία διαδικασία στο INNOVUS αποτελεί η εξαγωγή:

- Αρχείου Netlist με την επιλογή **Save Netlist** από το γραφικό περιβάλλον του INNOVUS.
- Αρχείου καθυστερήσεων SDF με την εντολή **write_sdf**.

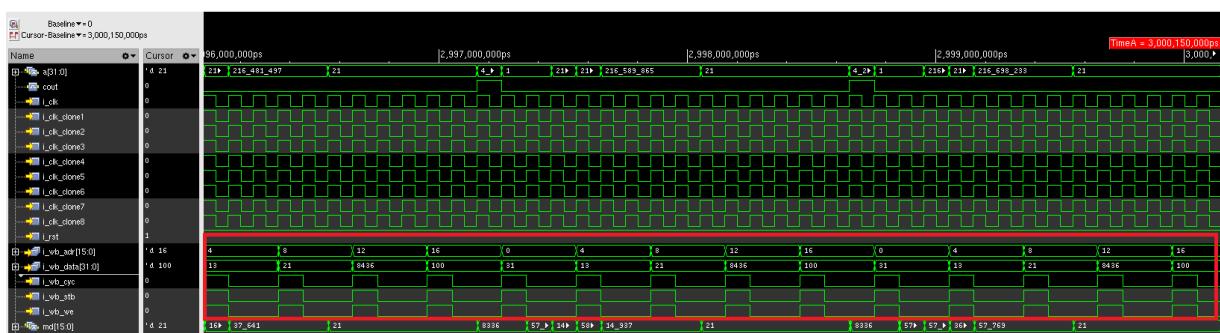
Στην προσομοίωση όπως έχει τονιστεί και παραπάνω, το σενάριο λειτουργίας παραμένει το ίδιο και αναμένεται στην έξοδο να προκύψει το σήμα παλμού PWM με duty – cycle στο 10%, σύμφωνα με το behavioral Testbench που έχει δημιουργηθεί για το σκοπό αυτό.



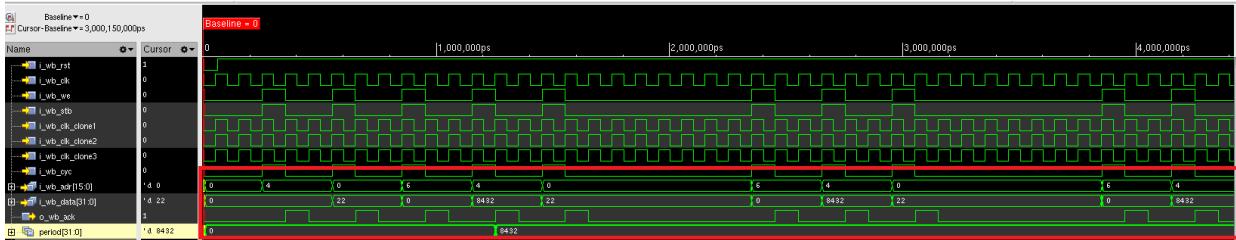
Εικόνα 83: Προσομοίωση συμπεριφοράς στο INCISIVE.



Εικόνα 84: Εσωτερικά του topmodule με εστίαση στα δεδομένα που εισέρχονται από τις εισόδους και αποθηκεύονται στους καταχωρητές.



Εικόνα 85: Στη μονάδα του PID, όπου τα δεδομένα εισέρχονται κανονικά και τη σωστή στιγμή.



Εικόνα 86: Στη μονάδα του PWM, όπου τα δεδομένα εισέρχονται κανονικά και τη σωστή στιγμή.

Αποτέλεσμα: Στην παραπάνω εικόνα που διακρίνεται το αποτέλεσμα της προσομοίωσης, φαίνεται πως η λειτουργία του συστήματος δεν είναι η αναμενόμενη. Το `o_pwm` που είναι και το σήμα εξόδου, σε κατάσταση σωστής λειτουργίας θα έπρεπε να έχει duty cycle στο 10%. Στην προκειμένη περίπτωση η τιμή του παραμένει ανεξήγητα κολλημένη στο 1 και ως εκ τούτου το αποτέλεσμα είναι λανθασμένο.

Στην εικόνα 82 φαίνεται πως εσωτερικά του `topmodule`, τα δεδομένα εισέρχονται επιτυχώς σειριακά στο σύστημα και αποθηκεύονται σωστά στους καταχωρητές. Στις μονάδες PWM και PID (εικόνα 83,84) αποδεικνύεται ότι τα δεδομένα είναι και εδώ τα αναμενόμενα και εισέρχονται στο εσωτερικό τους στον απαιτούμενο χρόνο. Αυτό σημαίνει, όπως και αποδείχθηκε ότι οι μονάδες των δυο Masters επίσης διανέμουν σωστά τις τιμές τους γιατί ο PID και ο PWM παίρνουν ως είσοδο τις εξόδους των Masters.

Συνεπώς, τη στιγμή που όλα κυλούν ομαλά γύρω από τη ροή των δεδομένων δεν είναι δυνατό να διακριθεί το σφάλμα που οδηγεί το παλμό της εξόδου να έχει τη κολλημένη τιμή στο 1, αφού κάθε μονάδα έχει την απαιτούμενη τιμή στον απαιτούμενο χρόνο για να πραγματοποιήσει τους υπολογισμούς της.

Διευκρινήσεις:

- Η λειτουργία του συστήματος, με βάση τη τελική στατική χρονική ανάλυση θα έπρεπε να είναι η αναμενομένη, λόγω της ένδειξης για μηδενικές παραβιάσεις τόσο για το setup time όσο και για το hold.
- Αρχική εκτίμηση ήταν πως το πρόβλημα οφειλόταν στο **Post – Route Optimization**, ενέργεια που έβγαζε παραβιάσεις για το hold και το setup τη στιγμή που στο ακριβώς προηγούμενο στάδιο αυτό του Nano Route, οι παραβιάσεις ήταν μηδενικές. Ως εκ τούτου, ούτε τότε η Netlist είχε ορθή προσομοίωση συμπεριφοράς και με προτροπή του καθηγητή του μαθήματος παραλείψαμε τη βελτιστοποίηση. Το αποτέλεσμα όμως που προέκυψε είναι το παραπάνω και συνεπώς το πρόβλημα δεν διορθώθηκε.
- Παρά την φιλότιμη προσπάθεια, δεν ήταν εφικτό να εντοπιστεί το πρόβλημα. Το σχέδιο αν και πέρασε -όπως τονίστηκε- επιτυχώς τα στάδια της φυσικής σχεδίασης, τις επαληθεύσεις που του καταβλήθηκαν αλλά και τη στατική χρονική ανάλυση, παράγει μια Netlist που το αποτέλεσμα είναι μη ορθό. Συμπεραίνουμε πως η διόρθωση του βγαίνει εκτός του πεδίου γνώσης μας.
- Χωρίς να υπάρχει η απαιτούμενη σιγουριά για αυτό, το πρόβλημα πιθανώς να οφείλεται στη κακή δόμηση του αρχείου LEF την οποία προαναφέραμε, και επιπροσθέτως στις παραβιάσεις που προκύπτουν από το Verify DRC για τις οποιες δεν ήταν εφικτή κάποια ενεργεία που να μπορέσει να τις εξαλείψει.

Συμπέρασμα

Η τελική αναφορά του συστήματος αποτελεί και την ολοκλήρωση του project της ομάδας ως προς την απαιτούμενη εργασία του μαθήματος « Εργαστήριο σχεδίασης SoC με εργαλεία CAD».

Με σκοπό την εισαγωγή στη σχεδίαση των ASIC συστημάτων, το μάθημα έδωσε την ευκαιρία βάσει της δομής των εργασιών του, να εμβαθύνουμε ως φοιτητές τόσο στις διαδικασίες που απαιτούνται ώστε ένα σύστημα από τη θεωρία να φτάσει στο σημείο της ολοκλήρωσης του όσο και στις δυσκολίες και τα πολλά προβλήματα που προκύπτουν ανά τα στάδια της ροής και που οδηγούν σε αναδιαμορφώσεις στον τρόπο υλοποίησης του.

Με την ιδιότητα του φοιτητή ήρθαμε σε επαφή με τα εργαλεία INCISIVE και INNOVUS από τη Cadence και Design Compiler και TetraMAX από τη Synopsys που χρησιμοποιούνται ευρέως στη βιομηχανία για τη κατασκευή ASIC συστημάτων. Η αξιοποίηση τους στη διάρκεια του εξάμηνου συνέβαλλε στην απόκτηση βασικών γνώσεων ως προς τη χρήση τους, ενώ μέσω αυτών παράλληλα αποκτήθηκε μια δόση εμπειρίας ως προς την σχεδίαση ενός System on Chip.

Τέλος, η επιλογή project και η διοχέτευση του από το πλαίσιο του προσδιορισμού του μέχρι και τη φυσική σχεδίαση ,συνέβαλλε στην θεωρητική κατανόηση μέσω όμως της πρακτικής εφαρμογής, των σταδίων που απαρτίζουν τη ροή σχεδίασης ενός SoC. Παράδειγμα των χρήσιμων γνώσεων και πληροφοριών που αποκτήθηκαν, η λογική σύνθεση, οι δομές DFT, η ανάγκη για στατική χρονική ανάλυση και ανάλυση ισχύος, η φυσική σχεδίαση κ.α.

Εκ μέρους της ομάδας ευχαριστούμε για τη συνεργασία τους βοηθούς Σταύρο Σίμογλου και Νικόλαο Σκετόπουλο και τον διδάσκοντα του μαθήματος, Χρήστο Σωτηρίου.