



Picorv 32

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΝΑΦΟΡΑ

Νίκος Χαραλάμπους | AEM: 9740 | Ψηφιακά VLSI

ΑΣΚΗΣΗ 1η

Bήμα 1:

Για τη διαδικασία της σύνθεσης, ορίζουμε τα μονοπάτια για τις βιβλιοθήκες χρονισμού (*.lib), φυσικών πληροφοριών (*.lef) και παρασιτικών (.tch). Αυτό επιτυγχάνεται εκτελώντας τις παρακάτω εντολές στο command prompt του εργαλείου **Genus** :

Αρχικά ορίζουμε τα μονοπάτια ως εξής :

```
#set paths leading to selected directories  
  
set_db init_lib_search_path/  
mnt/apps/prebuilt/eda/designkits/GPDK/gsclib045/lan/flow/t1u1/  
reference_libs/GPDK045/gsclib045_svt_v4.4/gsclib045/  
  
set_db script_search_path /home/n/nikolaoac/Desktop/Scripts/  
  
set_db init_hdl_search_path /home/n/nikolaoac/Desktop/HDL/
```

Στη συνέχεια, διαβάζουμε τις βιβλιοθήκες χρονισμού, τεχνολογίας και παρασιτικών από το 1^o path που ορίσαμε παραπάνω :

```
#set default library files from path of data  
  
set_db library ./timing/slow_vdd1v0_basicCells.lib  
  
set_db lef_library {./lef/gsclib045_tech.lef  
./lef/gsclib045_macro.lef }  
  
read_qrc ./qrc/qx/gpdk045.tch
```

Μετά την εκτέλεση των παραπάνω εντολών, λαμβάνουμε κάποια warnings από το εργαλείο που αφορούν τις βιβλιοθήκες χρονισμού και φυσικών πληροφοριών, σχετικά με output pins ή κελιών που δεν βρίσκονται ή δεν χρησιμοποιούνται από τις εν λόγω βιβλιοθήκες.

Bήμα 2:

Έχοντας κάνει τις αρχικές μας ρυθμίσεις, στη συνέχεια διαβάζουμε τα αρχεία που περιγράφουν το κύκλωμα μας. Το top-level module του κυκλώματός μας

είναι το **picorv32** . Μπορούμε να διαβάσουμε το αρχείο της Verilog που περιγράφει το σχέδιό μας με την παρακάτω εντολή :

```
#read picorv32.v Verilog design file  
read_hdl picorv32.v
```

Βέβαια, αυτό συμβαίνει αφού έχουμε αποθηκεύσει το αρχείο στο κατάλληλο μονοπάτι που ορίσαμε ως προεπιλογή για τα HDL αρχεία.

Bήμα 3:

Αφού αναγνώσουμε με επιτυχία το top-level module, κάνουμε μια πρώτη ανάλυση του κυκλώματος με την βοήθεια του εργαλείου. Αυτή υλοποιείται με την εντολή :

```
#perform design elaboration  
elaborate "picorv32"
```

Στη συνέχεια μπορούμε να ελέγξουμε το κύκλωμα για τυχόν unresolved references ή πρόσθετα προβλήματα. Έτσι εκτελούμε :

```
#export design check file  
check_design -all >  
/home/n/nikolaoac/Desktop/Check/Check_Design
```

Με αυτό τον τρόπο δημιουργούμε ένα αρχείο με τις απαραίτητες πληροφορίες μετά τον έλεγχο.

	Name	Total
1795		
1796		
1797	Unresolved References	0
1798	Empty Modules	0
1799	Unloaded Port(s)	32
1800	Unloaded Sequential Pin(s)	1
1801	Unloaded Combinational Pin(s)	51
1802	Assigns	68
1803	Undriven Port(s)	0
1804	Undriven Leaf Pin(s)	0
1805	Undriven hierarchical pin(s)	0
1806	Multidriven Port(s)	0
1807	Multidriven Leaf Pin(s)	0
1808	Multidriven hierarchical Pin(s)	0
1809	Multidriven unloaded net(s)	0
1810	Constant Port(s)	2
1811	Constant Leaf Pin(s)	0
1812	Constant hierarchical Pin(s)	1454
1813	Preserved leaf instance(s)	0
1814	Preserved hierarchical instance(s)	0
1815	Feedthrough Modules(s)	0
1816	Libcells with no LEF cell	0
1817	Physical (LEF) cells with no libcell	94
1818	Subdesigns with long module name	0
1819	Physical only instance(s)	0
1820	Logical only instance(s)	0
1821		
1822	Done Checking the design.	

Όπως παρατηρούμε και από τα περιεχόμενα του αρχείου που παράγεται, δεν λαμβάνουμε unresolved references.

Bήμα 4:

Σε αυτό το βήμα θα δημιουργήσουμε ένα αρχείο περιορισμών *.sdc , το οποίο θα παραχθεί από τους παρακάτω περιορισμούς που θα εισάγουμε στην σχεδίαση και θα αποθηκευτεί σε κατάλληλο φάκελο που ορίσαμε ως μονοπάτι σε προηγούμενο βήμα. Είναι περιορισμοί που κατά βάση έχουν να κάνουν με τον χρονισμό αλλά και με το φορτίο του κυκλώματος. Έτσι δημιουργούμε βήμα-βήμα τους περιορισμούς, δίνοντας τις παρακάτω εντολές :

```
#1)create a clock named 'clk' with 50% duty cycle and 5ns period
create_clock [get_ports clk] -name clk -period 5 -waveform {0 2.5}
```

```

#2)set clock latency @ 250ps
set_clock_latency -source 0.25 [get_clocks clk]

#3)set clock uncertainty @ 15ps for setup analysis
set_clock_uncertainty -setup 0.015 [get_clocks clk]
#set clock uncertainty @ 10ps for hold analysis
set_clock_uncertainty -hold 0.01 [get_clocks clk]

#4)set clock transition @ 1% of total period
set_clock_transition 0.05 [get_clocks clk]

#5)set output delay @ 0.75ns for setup time analysis
set_output_delay -max 0.75 -clock clk -network_latency_included
[all_outputs]

#6)set output delay @ 0.25ns for hold time analysis
set_output_delay -min 0.25 -clock clk -network_latency_included
[all_outputs]

#7)set load for setup time analysis @ 0.4pf
set_load 0.4 -max [all_outputs]

#8)set load for hold time analysis @ 0.05pf
set_load 0.05 -min [all_outputs]

#9)set input delay for setup time analysis @ 0.75ns
set_input_delay -max 0.75 -clock clk -network_latency_included
[all_inputs]

```

```

#10)set input delay for hold time analysis @ 0.25ns

set_input_delay -min 0.25 -clock clk -network_latency_included
[all_inputs]

#11)set input driving cell BUFX4 for setup time

set_driving_cell -max -lib_cell BUFX4 [all_inputs]

#set input driving cell BUFX8 for hold time

set_driving_cell -min -lib_cell BUFX8 [all_inputs]

```

Στη συνέχεια, αφού θέσαμε τους παραπάνω περιορισμούς, θα πρέπει να διαβάσουμε το αρχείο περιορισμών και να τους ελέγξουμε. Αυτό επιτυγχάνεται ακολούθως :

```

#read constraints file

read_sdc Constraints.sdc

#create checking timing intent file

check_timing_intent >
/home/n/nikolaoac/Desktop/Check/Check_Time_Intent

```

Έπειτα λαμβάνουμε το παραγόμενο αρχείο με τις εξής αναφορές :

Lint summary	
Unconnected/logic driven clocks	0
Sequential data pins driven by a clock signal	0
Sequential clock pins without clock waveform	0
Sequential clock pins with multiple clock waveforms	0
Generated clocks without clock waveform	0
Generated clocks with incompatible options	0
Generated clocks with multi-master clock	0
Paths constrained with different clocks	0
Loop-breaking cells for combinational feedback	0
Nets with multiple drivers	0
Timing exceptions with no effect	0
Suspicious multi_cycle exceptions	0
Pins/ports with conflicting case constants	0
Inputs without clocked external delays	0
Outputs without clocked external delays	0
Inputs without external driver/transition	0
Outputs without external load	0
Exceptions with invalid timing start-/endpoints	0
Total:	0

Πριν τελειώσουμε με το συγκεκριμένο βήμα, προκειμένου να δώσουμε την εντολή στο Genus για να μην δημιουργήσει Scan Flip-Flops , ενώ ταυτόχρονα δεν υπάρχουν αλυσίδες ανίχνευσης, εκτελούμε :

```
set_db / .use_scan_seqs_for_non_dft false
```

Είμαστε τώρα έτοιμοι για την σύνθεση του κυκλώματος.

Bήμα 5:

Εκτελούμε τα 3 βήματα της σύνθεσης (generic, mapping, optimization) σύμφωνα με τις παρακάτω εντολές.

```
#perform synthesis  
syn_generic  
syn_map  
syn_opt
```

Το εργαλείο, τρέχει τους αλγόριθμους παράγοντας το επιθυμητό αποτέλεσμα και κάνοντας βελτιστοποίηση του κυκλώματος.

Bήμα 6:

Στη συνέχεια, αφού έχουμε συνθέσει το κύκλωμα, μπορούμε να αποτιμήσουμε τα χαρακτηριστικά του, δίνοντας την εντολή στο εργαλείο να δημιουργήσει τα κατάλληλα reports. Πιο συγκεκριμένα με τις παρακάτω εντολές, λαμβάνοντας αναφορές σχετικά με την επιφάνεια, την ισχύ, τις πύλες, τον χρονισμό και την ποιότητα των αποτελεσμάτων αντίστοιχα. Οι αναφορές εξάγονται ως αρχεία σε επιλεγμένο προορισμό :

```
report_area > /home/n/nikolaoac/Desktop/Reports/Area  
report_power > /home/n/nikolaoac/Desktop/Reports/Power  
report_gates > /home/n/nikolaoac/Desktop/Reports/Gates
```

```
report_timing > /home/n/nikolaoac/Desktop/Reports/Timing
```

```
report_qor > /home/n/nikolaoac/Desktop/Reports/QoR
```

Έτσι λαμβάνουμε τα κάτωθι αποτελέσματα :

Total Area (μm^2)	Number of Cells	Slack (ns)	Power (W)
48527.595	10302	0	5.38122e-03

Με βάση τα αποτελέσματα, μας παραξενεύει μάλλον το γεγονός ότι το Slack είναι 0. Αν ρίξουμε μια προσεκτική ματιά στο Quality of Results αρχείο που δημιουργήθηκε θα καταλάβουμε πως δεν είναι απόλυτα μηδενικό αλλά πρακτικά ίσο με μηδέν.

```
Timing
-----
Clock Period
-----
clk 5000.0

Cost Critical Violating
Group Path Slack TNS Paths
-----
clk 0.5 0.0 0
default No paths 0.0
-----
Total 0.0 0
```

Αυτό συμβαίνει γιατί στο εν λόγω αρχείο ο χρόνος μετριέται σε picoseconds (π.χ. το clock είναι 5000 αντί για 5 ns), επομένως στην στήλη *Critical Path Slack* λαμβάνουμε Slack ίσο με 0.5 picoseconds το οποίο είναι 0.5e-03 nanoseconds, άρα προσεγγίζει το μηδέν.

Bήμα 7:

Έχοντας εκτελέσει όλη την παραπάνω προεργασία, χρησιμοποιώντας το εργαλείο Genus, είναι ώρα να εξάγουμε τα απαραίτητα αρχεία που θα

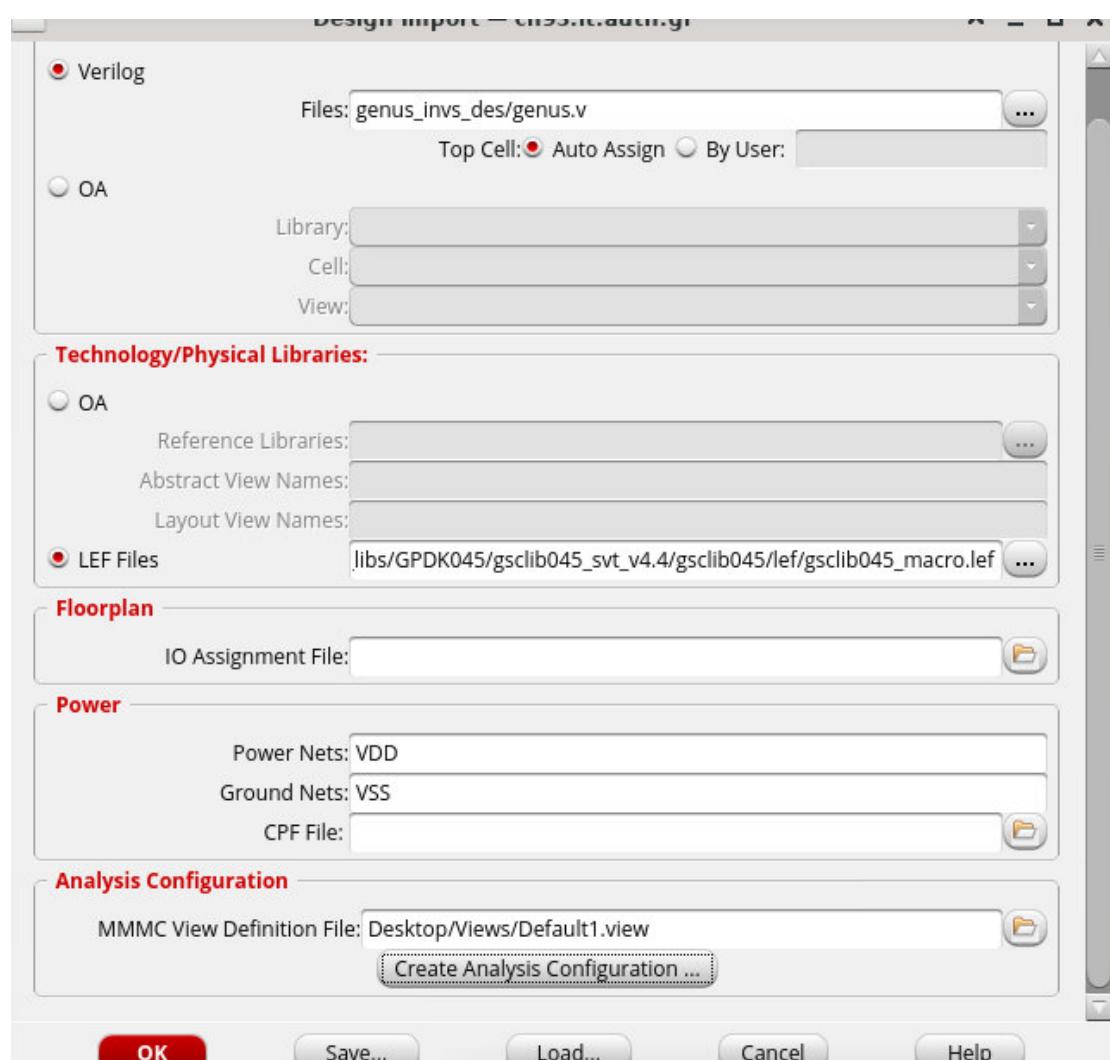
χρησιμοποιήσουμε στα επόμενα στάδια της ροής σχεδίασης με το εργαλείο Innovus. Δίνουμε την εντολή :

```
#create files for Innovus  
write_design -innovus picorv32
```

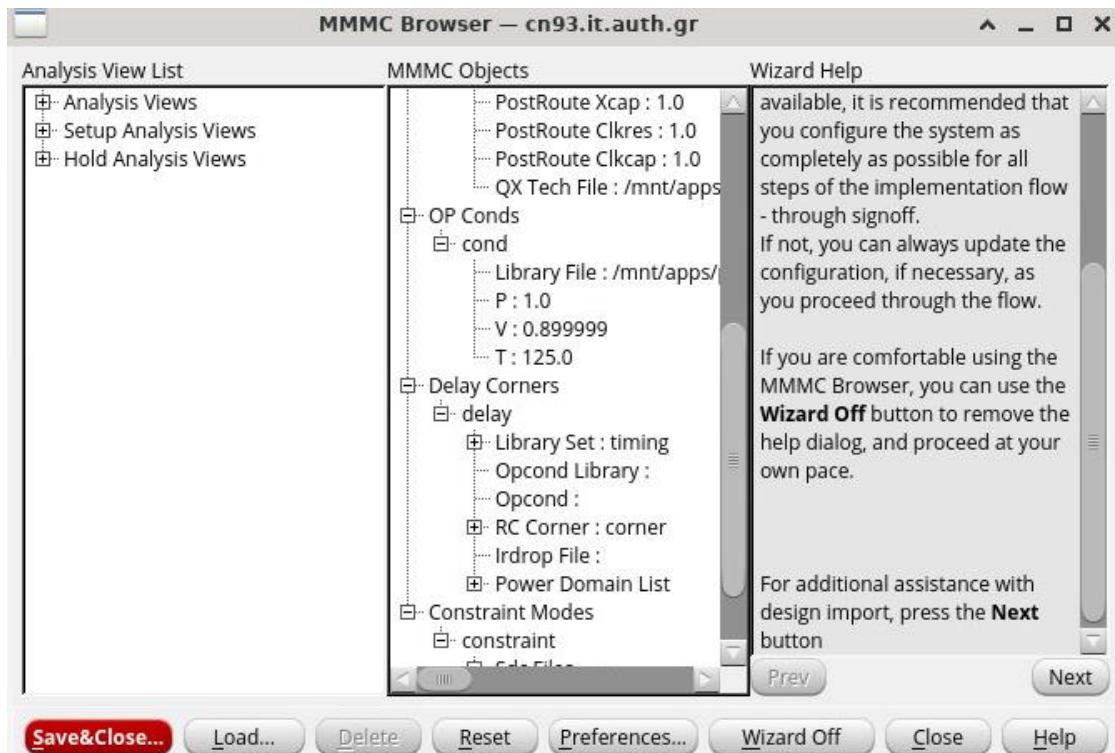
Με αυτό τον τρόπο, δημιουργείται ένας φάκελος με όνομα *genus_invs_des* στο User Directory ‘nikolaoac’ , ο οποίος περιέχει αρχεία τα οποία θα φανούν εξαιρετικά χρήσιμα στη συνέχεια.

Bήμα 8:

Στην συνέχεια εισάγουμε το Design στο Innovus κάνοντας Import Design κι έχοντας επιλέξει τις ακόλουθες ρυθμίσεις :



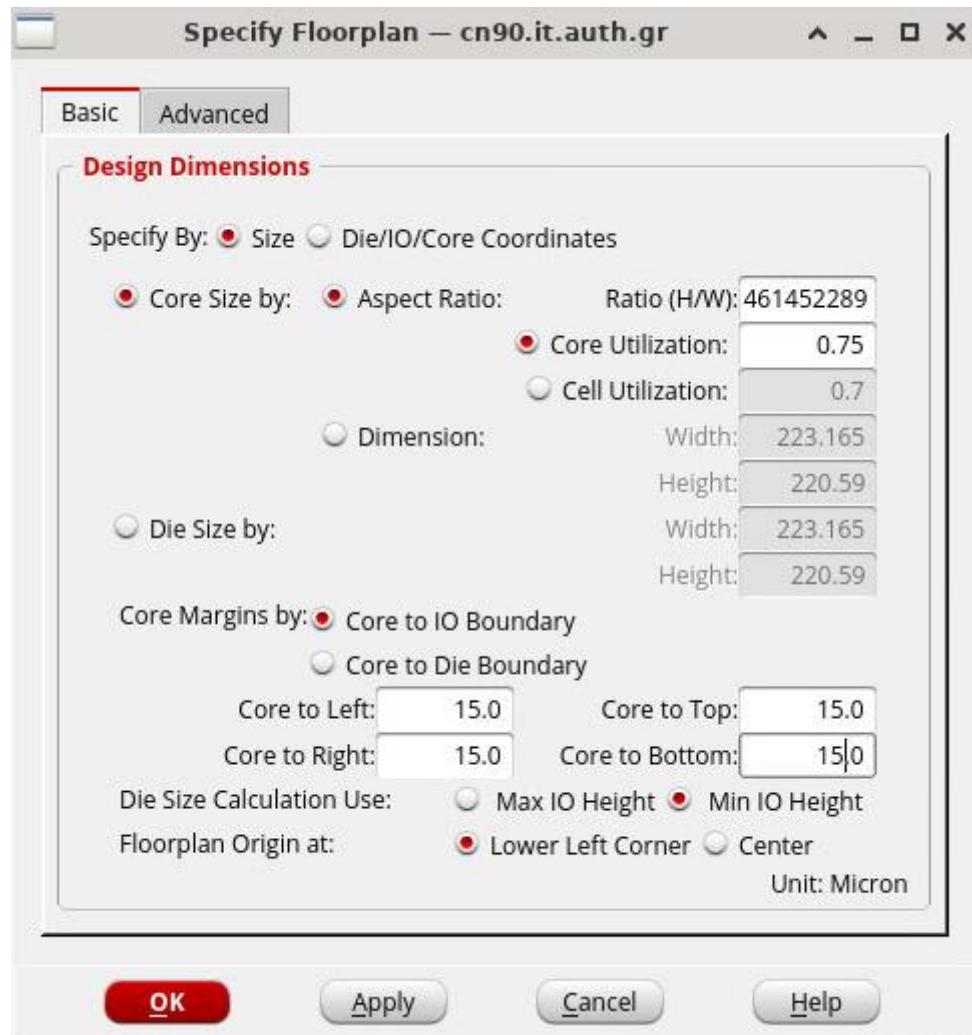
Στο πεδίο LEF files προσθέσαμε τις δύο lef βιβλιοθήκες που διαβάσαμε στο 1^o Βήμα από το προεπιλεγμένο path (macro & tech lef files). Κατόπιν δημιουργήσαμε την όψη (View) μέσω των MMMC ρυθμίσεων.



Στοιχεία για τις παραπάνω ρυθμίσεις βρήκαμε από το αρχείο *genus.mmmc.tcl* το οποίο ανήκει στον φάκελο *genus_invs_des* που δημιουργήθηκε στο προηγούμενο βήμα.

Βήμα 9:

Εφαρμόζουμε τη χωροθέτηση στο design με ποσοστό χρήσης του πυρήνα 75%. Επίσης, για το δίκτυο διανομής ισχύος, δημιουργούμε χώρο 15 μμ μέχρι τα I/O για τους δακτύλιους.

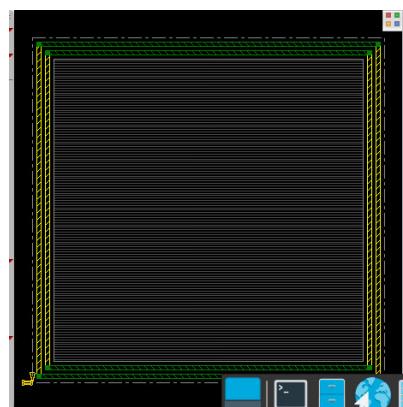


Bήμα 10:

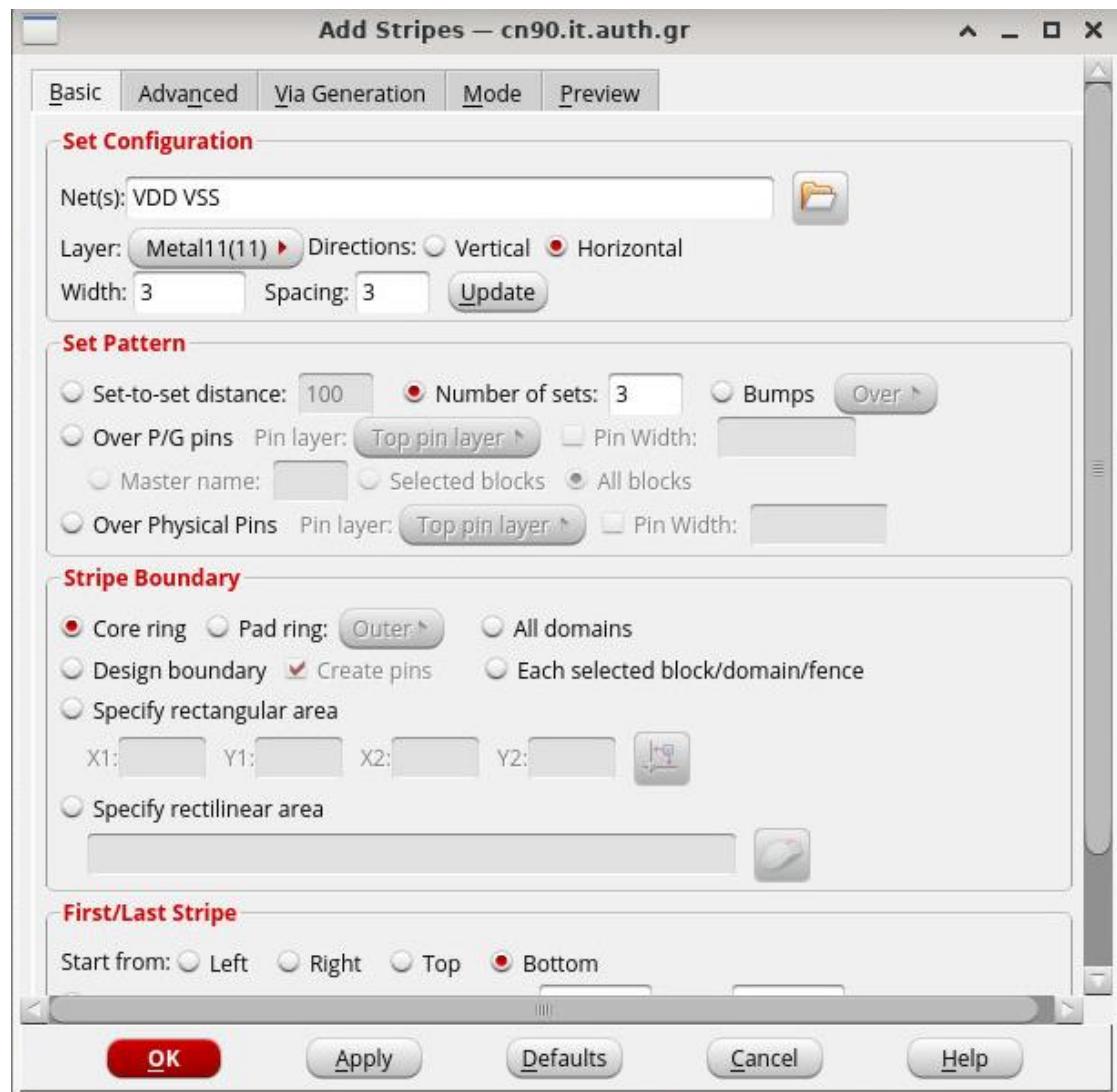
Στη συνέχεια, δημιουργούμε το δίκτυο διανομής ισχύος, με τους δακτύλιους (rings) να χρησιμοποιούν τα δύο ανώτερα μέταλλα (**M10-M11**). Επίσης, οι δακτύλιοι έχουν πάχος 3 μμ, κενό διάστημα ανάμεσα τους επίσης 3 μμ και βρίσκονται κεντραρισμένοι στο διάκενο ανάμεσα στον πυρήνα και στα I/O.



Στη συνέχεια, το design μας μοιάζει κάπως έτσι :



Όσον αφορά τις γραμμές, επιλέγουμε παρόμοιο πάχος και κενό, και αριθμό σετ ίσο με τρία.



Στη συνέχεια το design μας (μετά την προσθήκη των γραμμών-Stripes) μοιάζει κάπως έτσι :

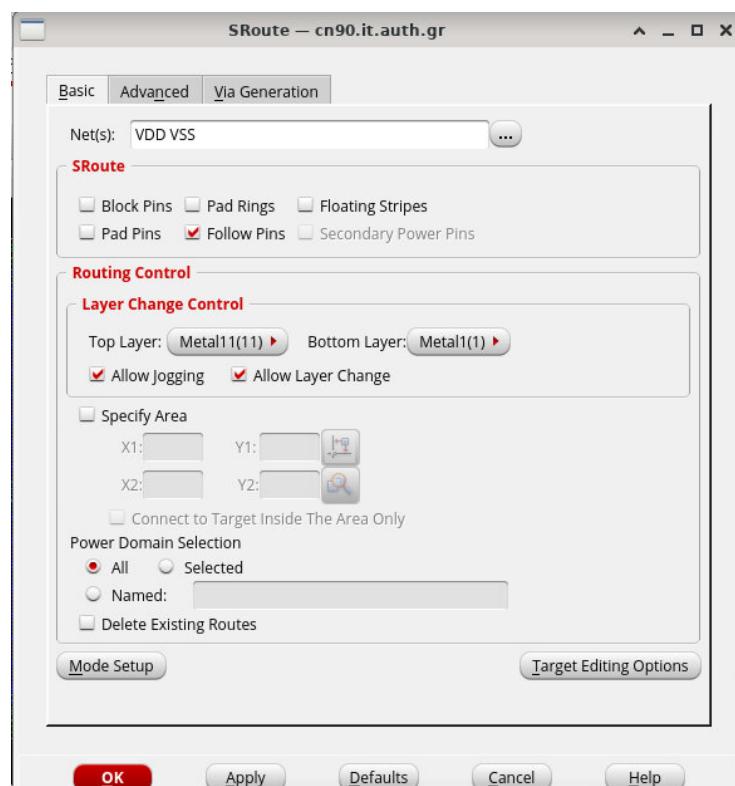


Τώρα πάμε να δημιουργήσουμε τους ακροδέκτες I/O τροφοδοσίας και γείωσης (pins). Αυτό επιτυγχάνεται με τη χρήση των παρακάτω εντολών :

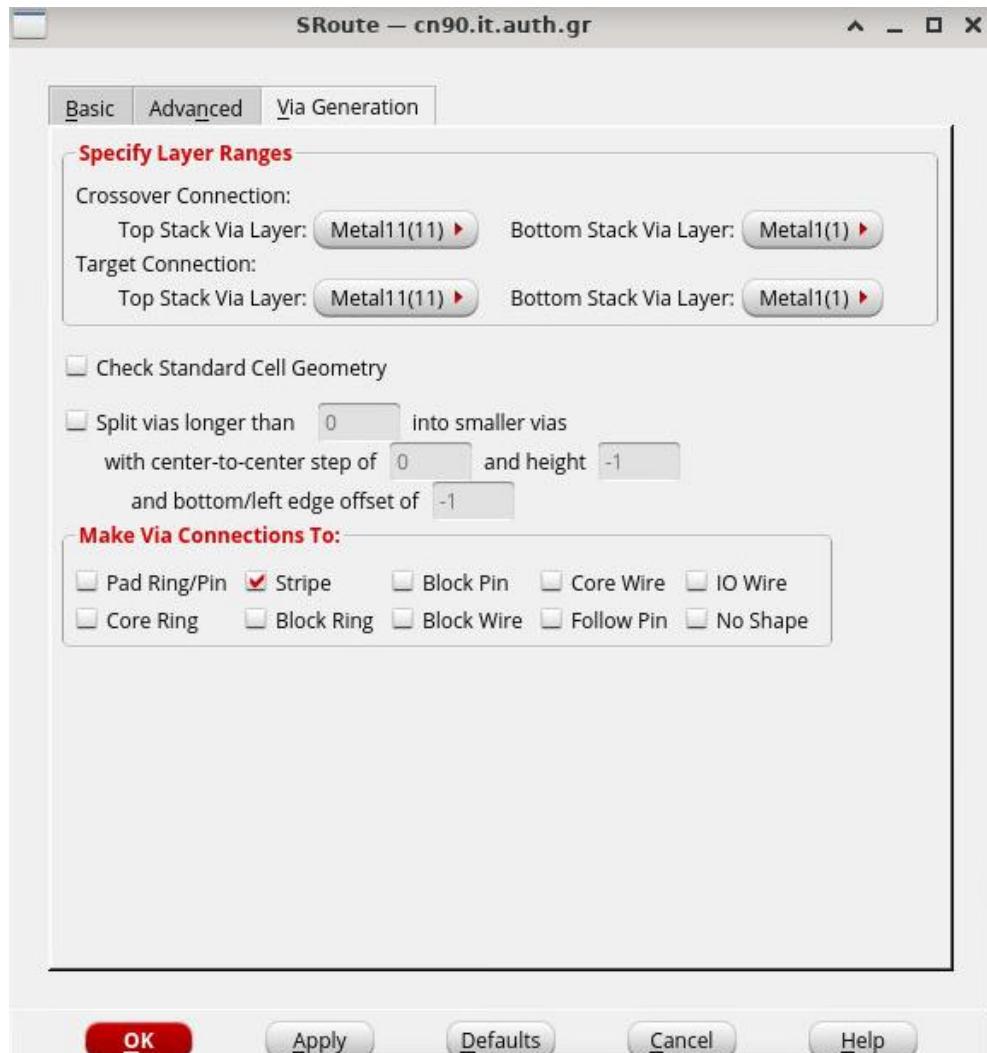
```
#create pins
globalNetConnect VDD -type pgpin -pin VDD -inst *
globalNetConnect VDD -type tiehi -instanceBasename *
globalNetConnect VSS -type pgpin -pin VSS -inst *
globalNetConnect VSS -type tielo -instanceBasename *

#specify pins location
createPGPin VDD -net VDD -geom Metal10 10 0 15 15
createPGPin VSS -net VSS -geom Metal10 2 0 8 8
```

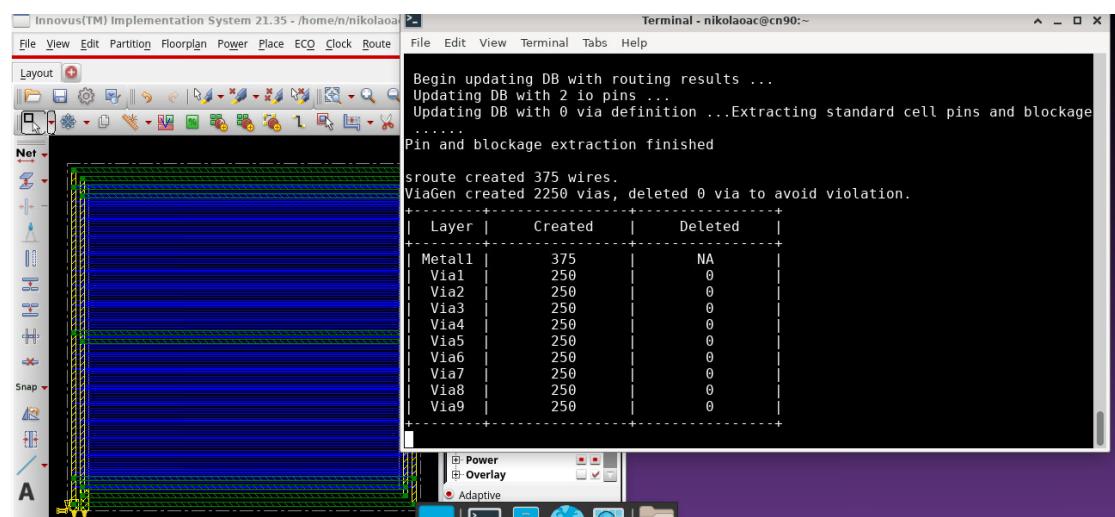
Στην ουσία, με τα παραπάνω δημιουργήσαμε τα pins και τα τοποθετήσαμε στο design. Επιλέξαμε σχετικά αυθαίρετα τις συντεταγμένες των δύο pins, καθώς η σύνδεση των ακροδεκτών με τους δακτυλίους γίνεται αυτόματα με τη διαδικασία SRoute (*Special Route*).



Επίσης στην καρτέλα Via Generation επιλέγουμε τη ρύθμιση Stripe για τις συνδέσεις Via.

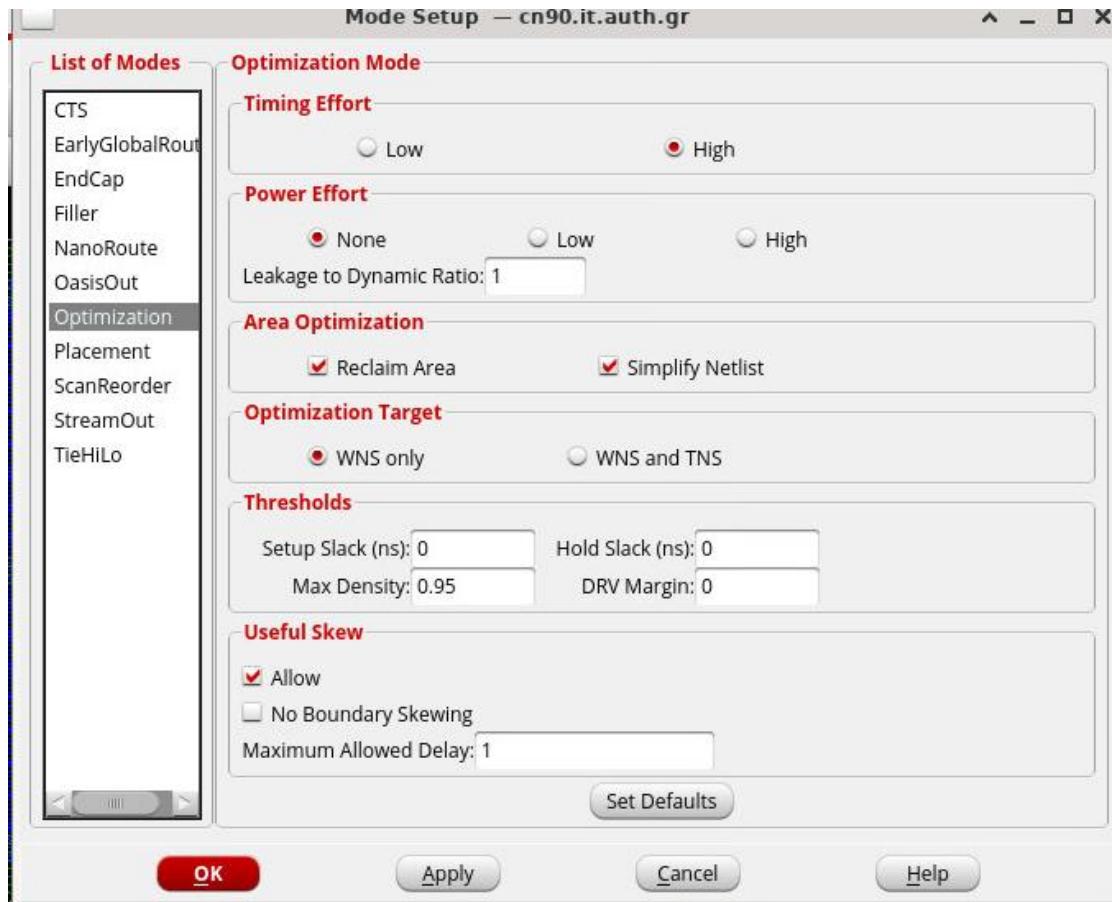


Μετά το SRoute λαμβάνουμε τα ακόλουθα αποτελέσματα τόσο στο Design όσο και στο Command Prompt :

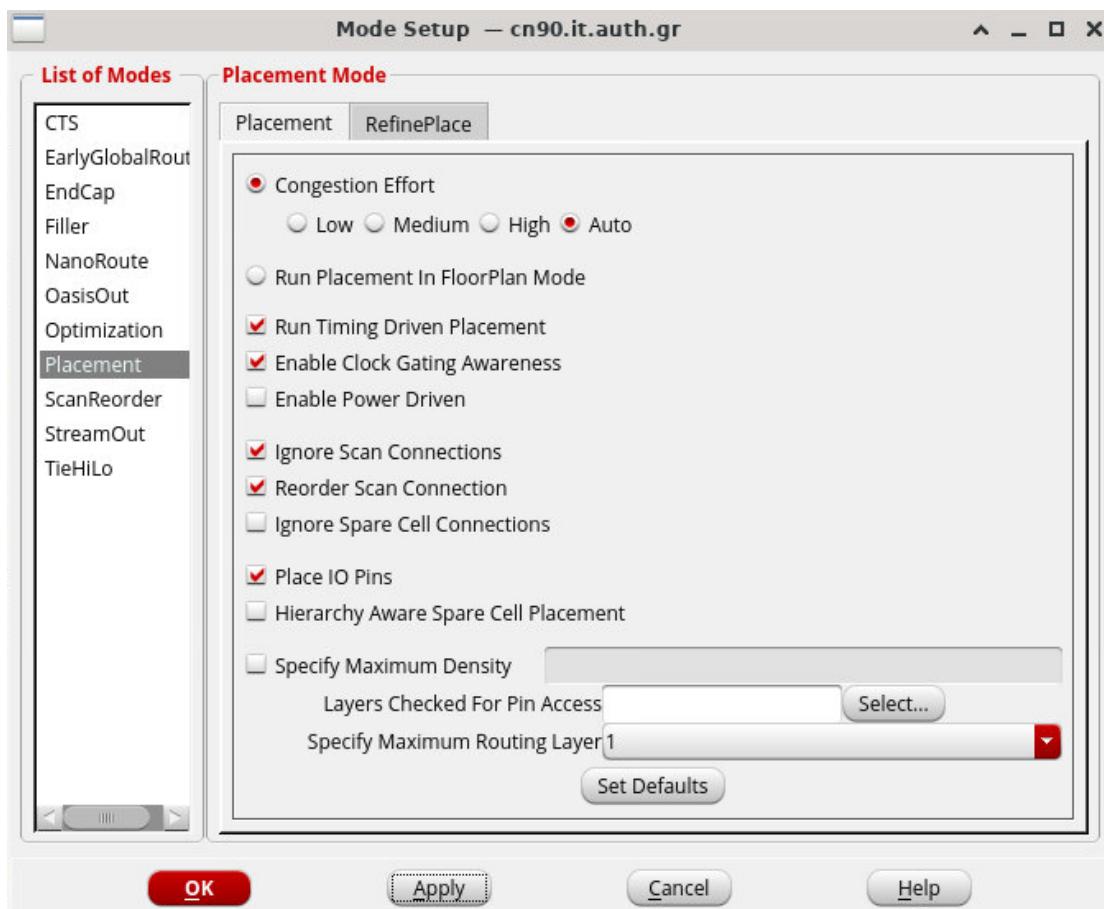


Βήμα 11:

Είμαστε έτοιμοι για την τοποθέτηση (Placement), αφού πρώτα ρυθμίσουμε την βελτιστοποίηση μέγιστα για τον χρονισμό και χωρίς για την ισχύ.



Στην καρτέλα Placement ενεργοποιούμε την επιλογή Place IO Pins, καθώς αν δεν το κάνουμε, το εργαλείο θα δημιουργήσει μόνο reg2reg μονοπάτια. Επίσης, το Place IO Pins κάνει το εργαλείο να προσθέσει τα I/O ανάλογα με την τοποθέτηση, προσπαθώντας ταυτόχρονα να βρει τις βέλτιστες θέσεις για αυτά.



Τώρα πριν από την τοποθέτηση, ελέγχουμε αν οι επιλογές *place_global_timing_effort* & *place_global_cong_effort* είναι ρυθμισμένες στο auto. Έπειτα εκτελούμε την τοποθέτηση των κελιών.

```
#check placement settings
getPlaceMode
#perform placement
place_opt_design
```

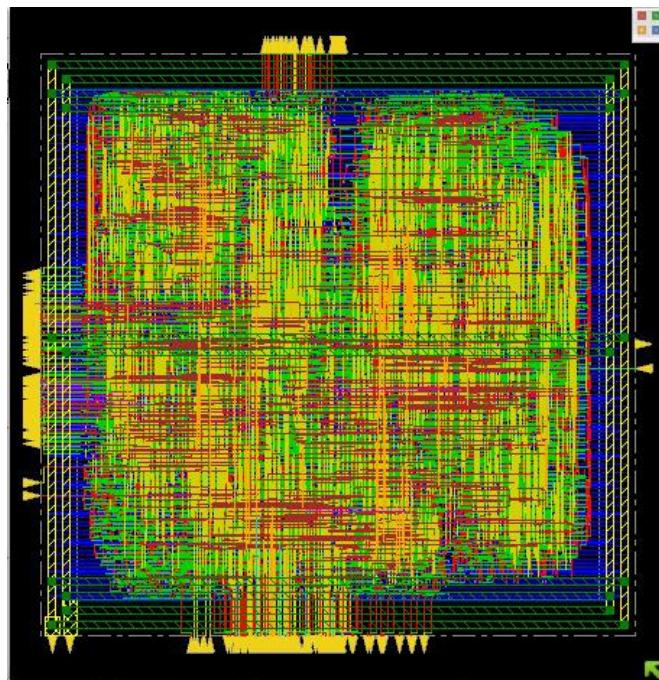
Στη συνέχεια, παράγουμε reports σχετικά με την ισχύ, την επιφάνεια και τον χρονισμό του κυκλώματος, όπως στο Βήμα 6. Τελικά λαμβάνουμε :

Internal Power (mW)	Switching Power (mW)	Leakage Power (mW)	Total Power (mW)	Slack (ns)	Total Area (μm²)
1.527	2.269	0.000854	3.797	0.014	34619.976

Τα παραπάνω αποτελέσματα παρατίθενται και στις ακόλουθες φωτογραφίες από τις αναφορές που δημιουργήθηκαν.

Area_Step11_3 ~/Desktop/Reports								
1	Hinst Name	Module Name	Inst Count	Total Area				
2	<hr/>		10757	34619.976	<hr/>			
<hr/>								
i6								
i7	Total Internal Power:	1.52696926	40.2136%					
i8	Total Switching Power:	2.26932098	59.7639%					
i9	Total Leakage Power:	0.00085396	0.0225%					
i0	Total Power:	3.79714420						
i1	<hr/>							
i2								
i3								
i4	Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)		
i5								
i6	<hr/>							
i7	Sequential	0.9762	0.08515	0.0002696	1.062	27.96		
i8	Macro	0	0	0	0	0		
i9	IO	0	0	0	0	0		
i0	Combinational	0.5508	2.184	0.0005844	2.736	72.04		
i1	Clock (Combinational)	0	0	0	0	0		
i2	Clock (Sequential)	0	0	0	0	0		
i3	<hr/>							
i4	Total	1.527	2.269	0.000854	3.797	100		
i5	<hr/>							
i6								
i7								
i8	Rail	Voltage	Internal Power	Switching Power	Leakage Power	Total Power (%)		
i9								
i0	<hr/>							
'1	VDD	0.9	1.527	2.269	0.000854	3.797 100		
'2	<hr/>							
<hr/>								
↳ Path 1: MET Late External Delay Assertion								
↳ Endpoint: mem_la_addr[31] (v) checked with leading edge of 'clk'								
↳ Beginpoint: resetn (▲) triggered by leading edge of 'clk'								
↳ Path Groups: {clk}								
↳ Analysis View: View1								
↳ Other End Arrival Time								
↳ - External Delay								
↳ + Phase Shift								
↳ - Uncertainty								
↳ = Required Time								
↳ - Arrival Time								
↳ = Slack Time								
↳ Clock Rise Edge								
↳ + Input Delay								
↳ + Drive Adjustment								
↳ = Beginpoint Arrival Time								
↳ +								
↳ Instance								
↳ Arc								
↳ Cell								
↳ Delay								
↳ Arrival Time								
↳ Req								
↳ +-----+								
↳ resetn ▲								
↳ A ▲ -> Y v								
↳ CLKINVX8								
↳ 0.136								
↳ NOR2X4								
↳ 0.122								
↳ NAND3X2								
↳ 0.117								
↳ 0.005								
↳ 0.001								

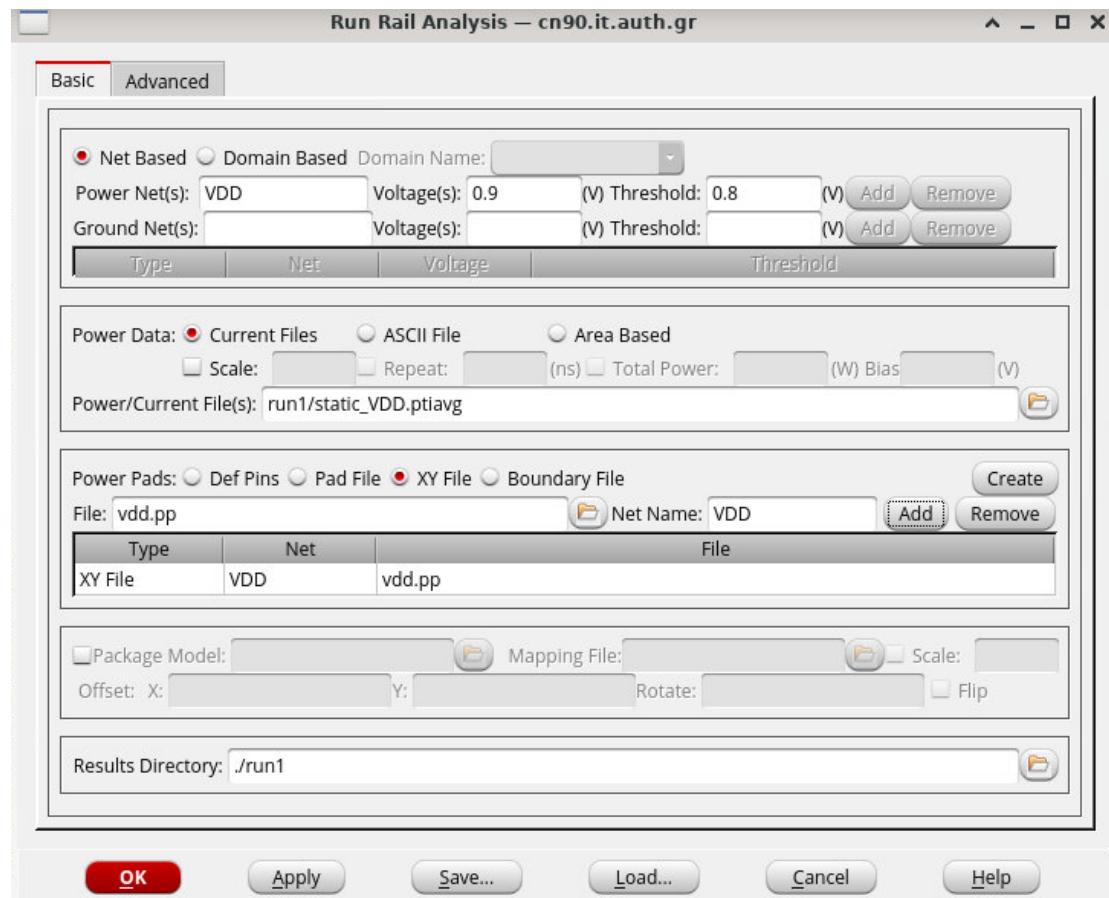
Μετά την τοποθέτηση, το κύκλωμά μας μοιάζει κάπως έτσι:



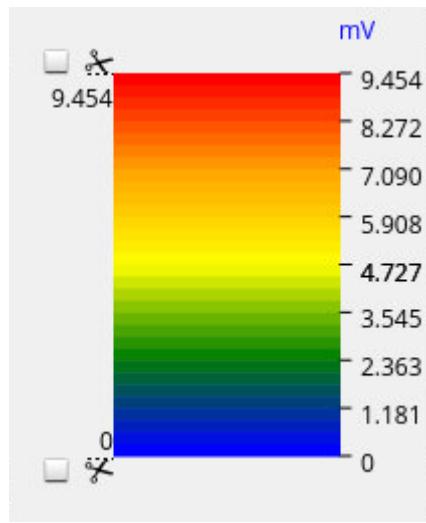
Τέλος, με την εντολή *Check Placement* παρατηρούμε πως δεν είχαμε κάποιο πρόβλημα με την τοποθέτηση. Αν θέλουμε, στη συνέχεια, μπορούμε να εκτελέσουμε την εντολή *optDesign -preCTS*, ωστόσο **δεν θα εκτελέσουμε τη συγκεκριμένη εντολή σε καμία άσκηση εκτός από την 8η και την 10η Άσκηση!!!**

Βήμα 12:

Στην συνέχεια εκτελούμε *Early Power Rail Analysis*. Εκτελούμε τα βήματα του Εργαστηριακού Οδηγού και στο βήμα του *Run Rail Analysis* βάζουμε τις εξής ρυθμίσεις με VDD = 0.9 V & Threshold = 0.8 V.



Τελικά λαμβάνουμε το παρακάτω IR-Drop σχήμα :



Παρατηρούμε ότι υπάρχει ένα εύρος IR drop , γεγονός που σημαίνει πως δεν έχει την ίδια τιμή για κάθε σημείο του κυκλώματος. Αυτό είναι λογικό, καθώς η πτώση τάσης σχετίζεται με την συγκέντρωση των κελιών σε κάθε περιοχή του κυκλώματος. Επίσης λαμβάνουμε μικρές τιμές IR drop σε σχέση με τη θεωρητική VDD, άρα δεν λαμβάνουμε μεγάλη πτώση τάσης γενικά κατά μήκος του design!

Βήμα 13:

Εφαρμόζουμε *Early Global Routing* πρώτα για όλα τα μέταλλα και ύστερα για τα μέταλλα M7 έως M10. Ταυτόχρονα με την εντολή `reportCongestion -hotspot` ελέγχουμε αν υπάρχει συμφόρηση για κάθε περίπτωση αντίστοιχα.

```
Terminal - nikolaoc@cn90:~  
File Edit View Terminal Tabs Help  
[NR-eGR] Early Global Route overflow of layer group 1: 0.00% H + 0.00% V. EstWL:  
2.075701e+05um  
[NR-eGR] Overflow after Early Global Route 0.00% H + 0.00% V  
[NR-eGR] Length (um) Vias  
[NR-eGR] -----  
[NR-eGR] Metal1 (1H) 16654 42787  
[NR-eGR] Metal2 (2V) 74333 27487  
[NR-eGR] Metal3 (3H) 74887 5071  
[NR-eGR] Metal4 (4V) 35342 1516  
[NR-eGR] Metal5 (5H) 13661 180  
[NR-eGR] Metal6 (6V) 2046 31  
[NR-eGR] Metal7 (7H) 308 21  
[NR-eGR] Metal8 (8V) 75 7  
[NR-eGR] Metal9 (9H) 61 3  
[NR-eGR] Metal10 (10V) 0 1  
Innovus 13>  
Innovus 13> reportCongestion -hotspot  
[hotspot] +-----+ +-----+  
[hotspot] | max hotspot | total hotspot |  
[hotspot] +-----+ +-----+  
[hotspot] | normalized | 0.00 | 0.00 |  
[hotspot] +-----+ +-----+  
local HotSpot Analysis: normalized max congestion hotspot area = 0.00, normalize  
d total congestion hotspot area = 0.00 (area is in unit of 4 std-cell row bins)  
Innovus 14>
```

Wirelength (μm)	Number of Vias
217370	77104

```

2.077394e+05um
NR-eGR] Overflow after Early Global Route 0.00% H + 0.02% V
NR-eGR]                                     Length (um)    Vias
NR-eGR] -----
NR-eGR]   Metal1   (1H)          0  40942
NR-eGR]   Metal2   (2V)          0  41033
NR-eGR]   Metal3   (3H)          0  41111
NR-eGR]   Metal4   (4V)          0  41158
NR-eGR]   Metal5   (5H)          0  41198
NR-eGR]   Metal6   (6V)          0  41217
NR-eGR]   Metal7   (7H)         74082  62420
NR-eGR]   Metal8   (8V)        92068  5509
NR-eGR]   Metal9   (9H)        32851  2179
NR-eGR]   Metal10  (10V)       20091  1
NR-eGR]   Metal11  (11H)          0  0
NR-eGR] -----
NR-eGR]   Total           219091  316768
NR-eGR] -----
NR-eGR] Total half perimeter of net bounding box: 175293um
NR-eGR] Total length: 219091um, number of vias: 316768
NR-eGR] -----
nnovus 14> reportCongestion -hotspot
hotspot] +-----+ +-----+
hotspot] |           | max hotspot | total hotspot |
hotspot] +-----+ +-----+
hotspot] | normalized |      0.00 |      0.00 |
hotspot] +-----+ +-----+
ocal HotSpot Analysis: normalized max congestion hotspot area = 0.00, normalize
total congestion hotspot area = 0.00 (area is in unit of 4 std-cell row bins)
nnovus 15> █

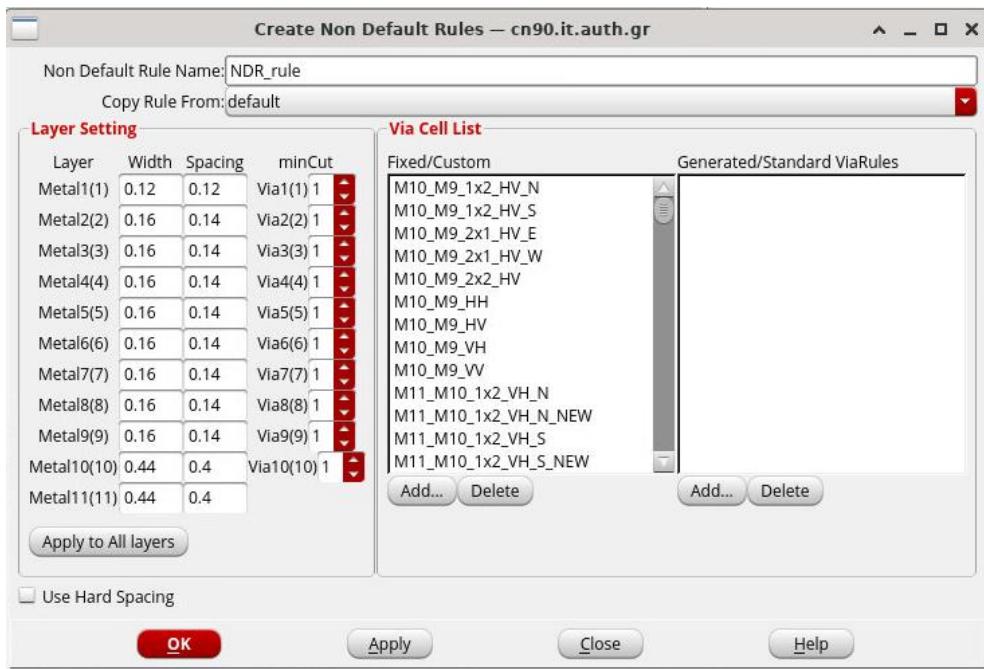
```

Wirelength (μm)	Number of Vias
219091	316768

Παρατηρούμε πως παίρνουμε στην έξοδο του εργαλείου, πως δημιουργούνται πολλά vias στα κατώτερα μέταλλα, ενώ το routing γίνεται στα υψηλότερα M7-M10. Αυτό συμβαίνει καθώς, παρόλο που γίνεται βελτιστοποίηση στα υψηλότερα μέταλλα, τα κατώτερα μέταλλα είναι χρήσιμα για τη λειτουργία του κυκλώματος κι έτσι καθίσταται απαραίτητη η δημιουργία περισσότερων vias για την επικοινωνία όλων των layers μετάλλων.

Βήμα 14:

Επόμενο βήμα, αποτελεί η σύνθεση του δέντρου ρολογιού (clock tree). Αρχικά, δημιουργούμε ένα *Non Default Rule* (NDR) με διπλάσιο πάχος και κενό σε σχέση με το default (2W2S) για όλα τα μέταλλα. Αυτό μπορεί αλλάζοντας τις default τιμές για το πάχος και το κενό στις διπλάσιες τιμές όπως παρακάτω :



Έστερα, δημιουργούμε έναν τύπο δρομολόγησης που να ακολουθεί το συγκεκριμένο NDR (μόνο για τα trunks του δέντρου) για τα μέταλλα M7-M9 και τον ονομάζουμε *NDR_rule*. Για τα φύλλα του δέντρου αφήνουμε ως NDR την default προεπιλογή (1W1S). Επίσης δημιουργούμε *Halos* για όλα τα κελιά που θα χρησιμοποιηθούν στην σύνθεση του δέντρου ρολογιού με μήκος και ύψος 0.1 μμ. Αυτά γίνονται με τις παρακάτω εντολές :

```
#create trunk route type

create_route_type -top_preferred_layer 9 -
bottom_preferred_layer 7 -non_default_rule NDR_rule -name
trunk_route

set_ccopt_property -route_type trunk_route -net_type trunk

#create leaf route type
```

```

create_route_type -top_preferred_layer 9 -
bottom_preferred_layer 7 -name leaf_route

set_ccopt_property -route_type leaf_route -net_type leaf

#create halos

set_ccopt_property -cell_halo_x 0.1

set_ccopt_property -cell_halo_y 0.1

```

Στη συνέχεια, θέτουμε την στρέβλωση ρολογιού στα 0.5 ns (10% της περιόδου) και το μέγιστο ρυθμό μετάβασης ρολογιού ίσο με 0.06 ns (1,2% της περιόδου). Κατόπιν, προχωράμε στη σύνθεση του ρολογιού, σύμφωνα με τις εντολές :

```

#set skew @ 0.5ns

set_ccopt_property -target_skew 0.5

#set max transition @ 0.06ns because of error with 0.05ns

set_ccopt_property -target_max_trans 0.06

#create clock tree spec file

create_ccopt_clock_tree_spec -file Clock_Tree.spec

#create clock tree

ccopt_design

#design optimization after clock tree synthesis

optDesign -postCTS

```

Μετά τη σύνθεση του ρολογιού μπορούμε να τρέξουμε την εντολή `report_ccopt_cell_halo_violations` για να δούμε αν έχουμε τυχόν παραβιάσεις σχετικά με τα halos που δημιουργήσαμε. Έτσι λαμβάνουμε στην έξοδο του εργαλείου :

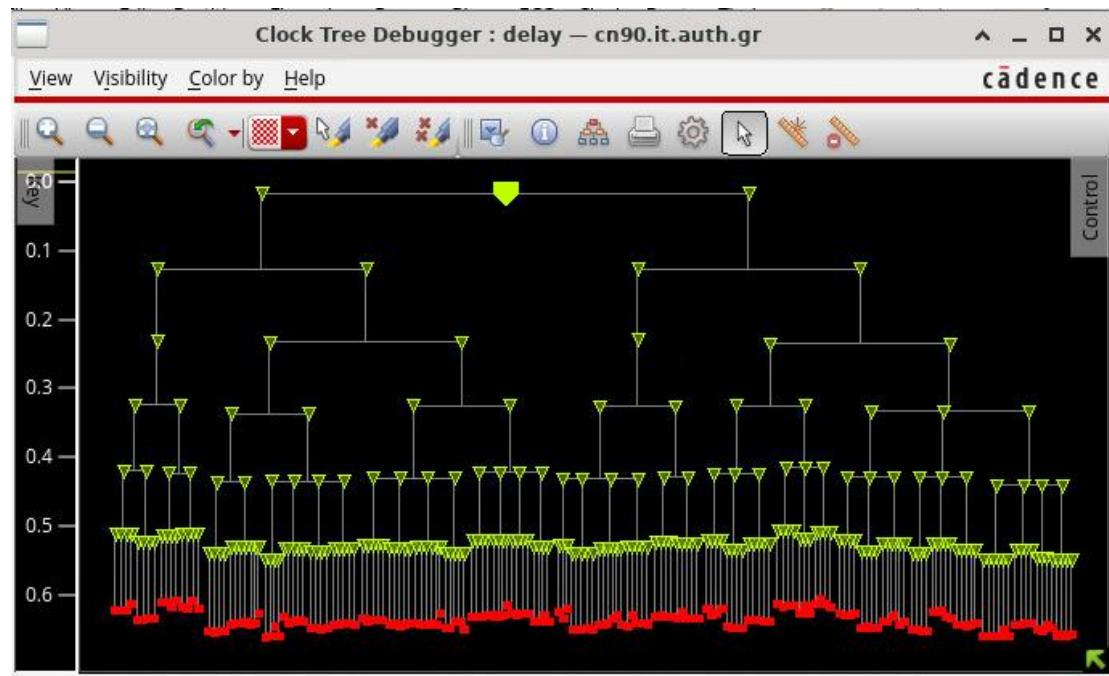
```

Terminal - nikolaaoac@cn90:~
File Edit View Terminal Tabs Help
Clock Cell Halo Rule Check in Summary
=====
Clock      Instances with cell_halo      Failed instances
clk          245                      0

Total number of violating instances: 0
Clock Cell Halo Rule Violations
=====

No.   Cell     Instance    In clock    Power domain    Cell Halo (x, y)    Vio-
ting instance   From clock   Intrusion (x, y)
=====
(empty table)
=====
```

Επομένως βλέπουμε ότι δεν υπήρξε κάποιο πρόβλημα στην τοποθέτηση των halos. Ακόμα, το δέντρο ρολογιού έχει την ακόλουθη μορφή :



Τέλος, παρόμοια με προηγούμενα βήματα, εξάγουμε αναφορές σχετικά με την κατανάλωση ισχύος, το slack & την επιφάνεια. Τα νέα στοιχεία που προέκυψαν παρουσιάζονται εδώ :

Internal Power (mW)	Switching Power (mW)	Leakage Power (mW)	Total Power (mW)	Slack (ns)	Total Area (μm^2)
1.691	2.485	0.0008784	4.176	0.048	35165.124

Επίσης μπορούμε να παράξουμε αναφορές σχετικά με πληροφορίες για το δέντρο ρολογιού, όσο και για τα skew groups ως εξής :

```
#create clock tree report
report_ccopt_clock_trees >
/home/n/nikolaoac/Desktop/Reports/Ex1/Clock_Tree

#create skew groups report
report_ccopt_skew_groups >
/home/n/nikolaoac/Desktop/Reports/Ex1/Skew_Groups
```

Μέσα από αυτές τις αναφορές μπορούμε να βγάλουμε χρήσιμα συμπεράσματα :

Cell type	Count	Area	Capacitance
<hr/>			
Buffers	245	546.858	0.091
Inverters	0	0.000	0.000
Integrated Clock Gates	0	0.000	0.000
Discrete Clock Gates	0	0.000	0.000
Clock Logic	0	0.000	0.000
All	245	546.858	0.091
<hr/>			

Παρατηρούμε ότι ο αριθμός των **buffers** ρολογιού ανέρχεται στους **245**.

Skew Group	Sources	Constrained Sinks	Unconstrained Sinks
clk/constraint	1	1961	0

Skew Group Summary:
=====

Timing Corner Target Type	Skew Group Target	ID Skew	Target	Min ID	Max ID	Avg ID	Std.Dev. ID	Skew window occupancy
delay:both.early	clk/constraint	-	0.580	0.635	0.610	0.012		
ignored	-	0.056	-					
delay:both.late	clk/constraint	none	0.589	0.644	0.620	0.012		
explicit	0.500	0.056	100% {0.589, 0.644}					

Τελικώς, δημιουργείται ένα (1) μόνο **skew group**, που όμως παράγει στην έξοδο διαφορετικές αναλύσεις για κάθε Corner Type που έχουμε ορίσει.

```

3 clk          0          0          0          0      383
4 -----
5 Total        0          0          0          0      383
6 -----
7
8 Note the above table per clock tree is based on CCopt clock tree view. The violations are counted
9 across half corners.
?
10 Found a total of 0 clock tree pins with max capacitance violations.
11 Found a total of 0 clock tree nets with max resistance violations.
12 Found a total of 0 clock tree nets with max length violations.
13 Found a total of 0 clock tree nets with max fanout violations.
14 Found a total of 383 clock tree pins with a slew violation.
15
16 Slew violation summary across all clock trees - Top 10 violating pins:
17 =====
18
19 Target and measured clock slews (in ns):
20 -----
21
22 Half corner     Violation   Slew    Slew   Dont   Ideal   Target   Pin
23           amount     target   achieved touch   net?    source
24           net?
25
26 delay:both.late 0.006    0.060   0.066   N     N     explicit genblk2.pcpi_div_dividend_reg[27]/
27 CK
28 delay:both.late 0.006    0.060   0.066   N     N     explicit genblk2.pcpi_div_dividend_reg[28]/

```

Επίσης, παρατηρούμε πως έχουμε **383 slew (max transition)** violations με τη μέγιστη παραβίαση να ανέρχεται στα 0.006 ns (δηλ. αντί για 0.06 -> 0.066). Επομένως, δεν ικανοποιούνται πάντα οι στόχοι που θέσαμε για τον μέγιστο ρυθμό μετάβασης ρολογιού. Ωστόσο, δεν έχουμε skew (στρέβλωση) violations καθώς το **skew** που παίρνουμε είναι μόλις **0.056 ns** (!) και μπορούμε να το δούμε στο αρχείο των *Skew Groups*.

Metric	Minimum	Average	Maximum	Std.dev
Source-sink routed net length (um)	6.810	22.163	83.155	9.323
Source-sink manhattan distance (um)	6.030	20.821	83.145	9.159
Source-sink resistance (Ohm)	29.828	81.402	153.629	29.464

Clock Tree Buffering Structure (Logical):

```

# Buffers          : 245
# Inverters       : 0
      Total        : 245
      Minimum depth : 6
      Maximum depth : 6
      Buffering area (um^2) : 546.858

```

Το **ελάχιστο βάθος δέντρου ρολογιού** ανέρχεται στα **6.81 μm**, ενώ το **μέγιστο** ανέρχεται στα **83.155 μm**. Το **μέγιστο & ελάχιστο λογικό βάθος**

```

5 Clock DAG wire lengths:
=====
3
3 -----
3 Type      Wire Length
L -----
? Top          0.000
3 Trunk        2210.140
! Leaf         7218.685
5 Total        9428.825
3 -----

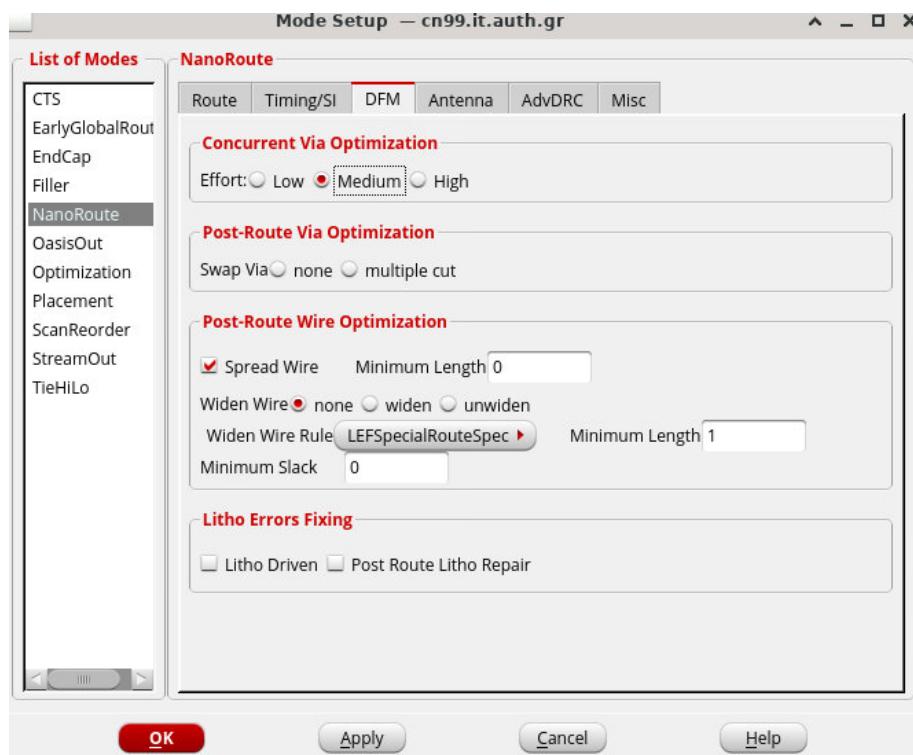
```

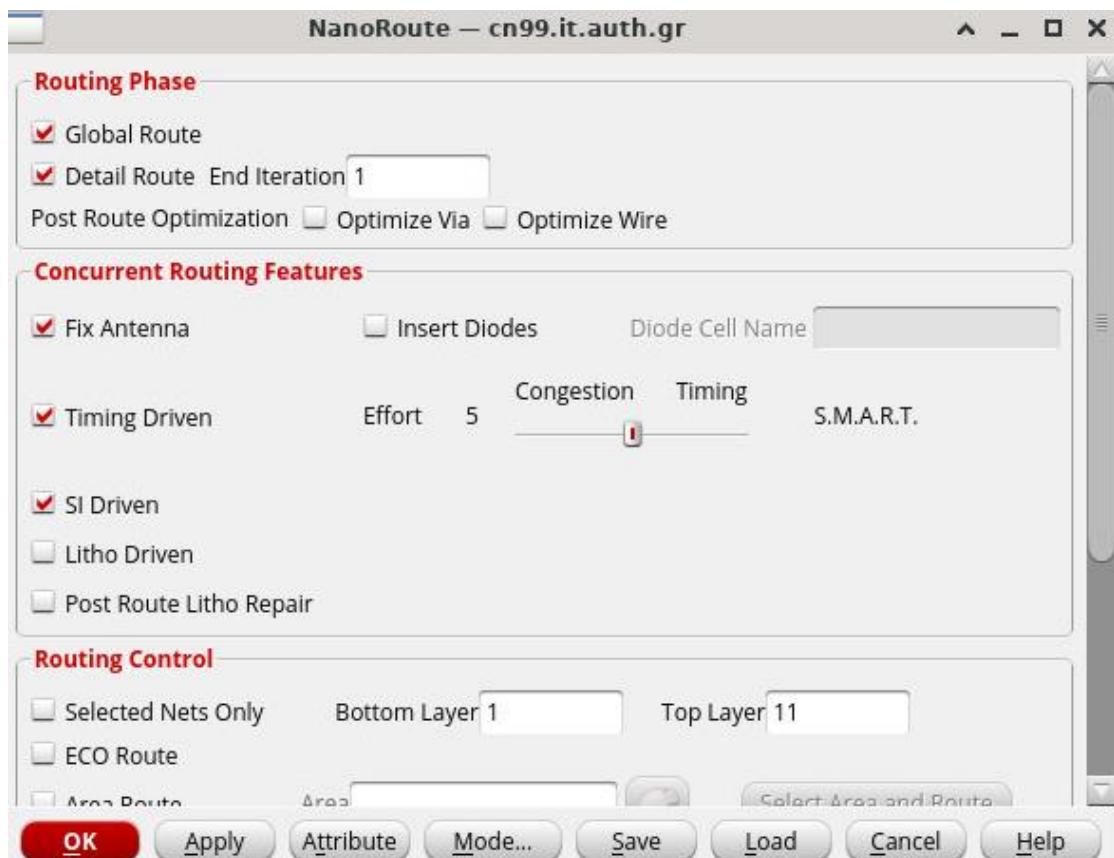
είναι ίσα με **6**.

Τέλος, το **μήκος δρομολόγησης** είναι **2210.14 μm** & **7218.685 μm** για το **trunk** και για τα **leaves**, αντίστοιχα.

Bήμα 15:

Στη συνέχεια προχωράμε σε *NanoRoute* δρομολόγηση με τις παρακάτω ρυθμίσεις όπως υποδεικνύει η εκφώνηση της εργασίας :



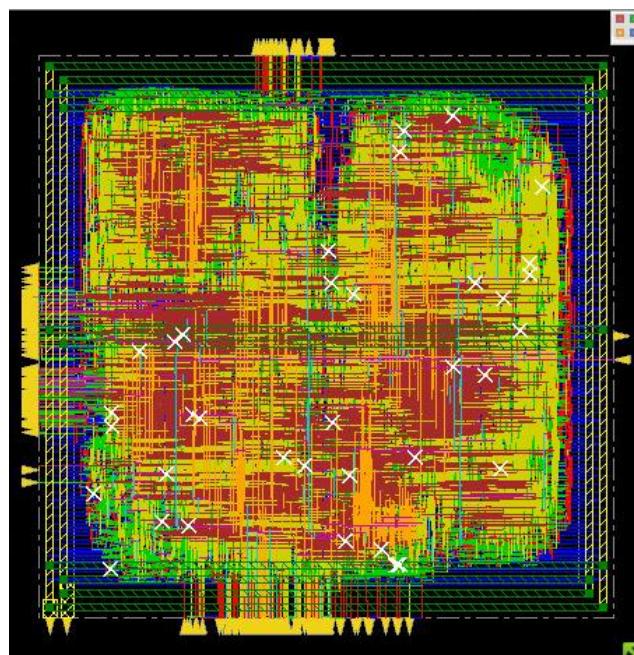


Στο τέλος της δρομολόγησης, λαμβάνουμε τα πρώτα μας DRC violations. Δεν προσπαθούμε να τα επιλύσουμε ακόμα. Θα δοκιμάσουμε να κάνουμε βελτιστοποίηση του κυκλώματος όσο αφορά το κομμάτι του χρονισμού. Για το σκοπό, θα βοηθήσει η εντολή optDesing -postRoute -setup -hold . Όμως αν δοκιμάσουμε να την εισάγουμε στο εργαλείο απευθείας παίρνουμε errors. Σε αυτό το βήμα συμβουλευτήκαμε κάποια AI GPT Chats τα οποία μας πρότειναν να εισάγουμε τις ακόλουθες εντολές για να λυθεί το πρόβλημα :

```
setAnalysisMode -analysisType onChipVariation -cppr both
setDelayCalMode -engine aae -SIAware true
#now perform postRoute optimization
optDesing -postRoute -setup -hold
```

Bήμα 16:

Στο τελικό βήμα της άσκησης, θα προσπαθήσουμε αρχικά να επιλύσουμε τις DRC παραβάσεις που λαμβάνουμε. Για το σκοπό αυτό, κάνουμε *Verify DRC* κι έπειτα εκτελούμε την εντολή *ecoRoute -fix_drc*. Στη συνέχεια βλέπουμε πως συνεχίζουμε να λαμβάνουμε *DRC Violations*. Για να αναγνωρίσουμε τις παραβάσεις αυτές ανοίγουμε τον *Violation Browser* κι έχουμε τα παρακάτω αποτελέσματα :



Violation Browser — cn99.it.auth.gr

Violation Type:

- NanoRoute (56/56)
 - Connectivity (1/1)
 - Metal_Loop (1/1)
 - Metal3(3) (1/1)
 - Geometry (55/55)

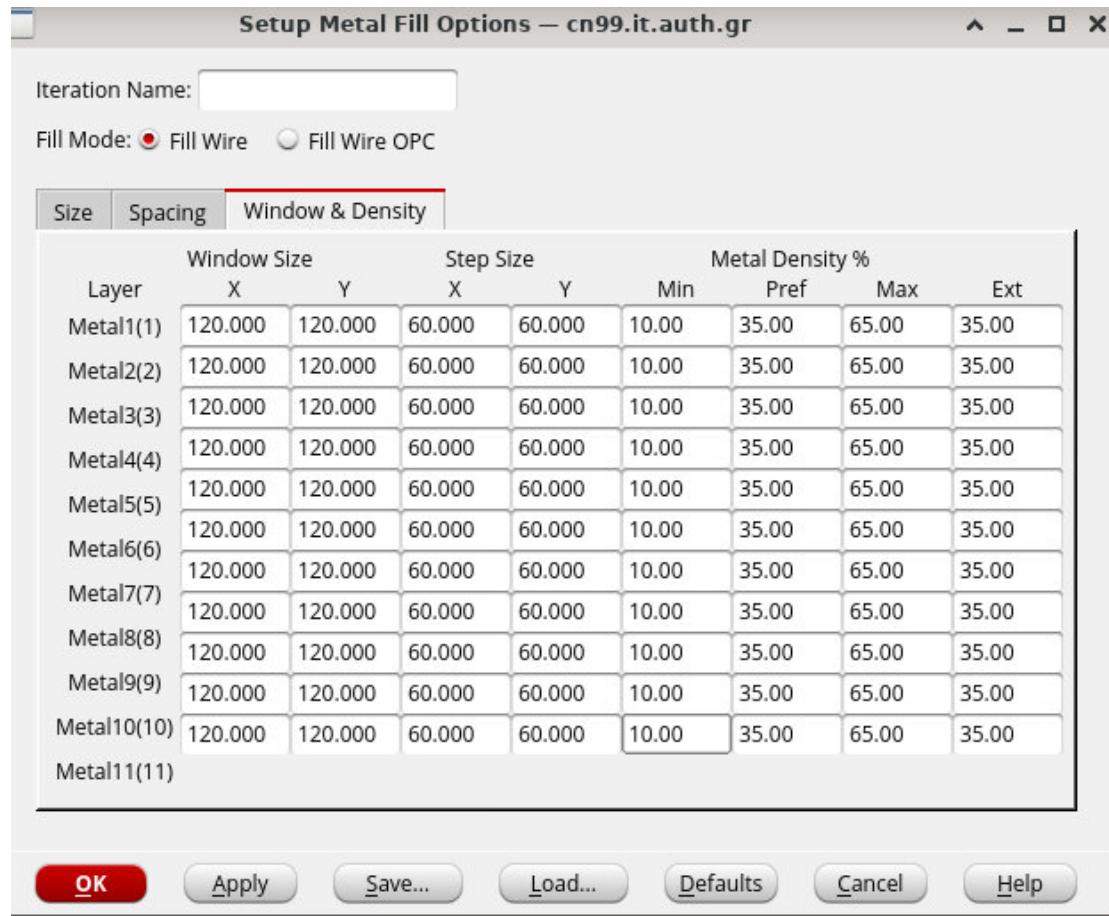
Verify (46/46)
 - Geometry (46/46)
 - Metal_EndOfLine_Spacing (1/1)
 - Metal1(1) (1/1)
 - Metal_Short (24/24)
 - Metal1(1) (8/8)
 - Metal2(2) (16/16)
 - MinHole (1/1)
 - Metal3(3) (1/1)
 - Parallel_Run_Length_Spacing (20/20)
 - Metal1(1) (15/15)
 - Metal2(2) (5/5)

Violation:

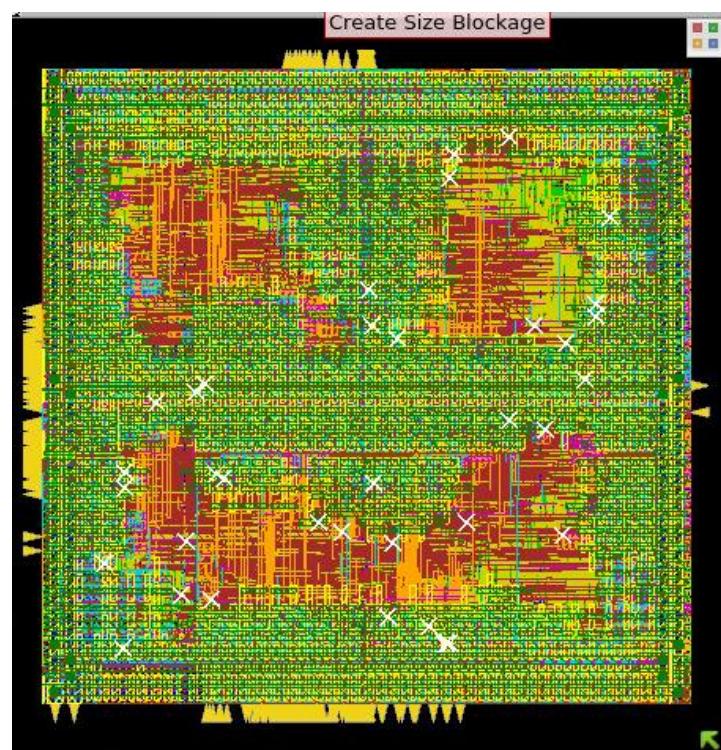
LAYER	OBJECT1	OBJECT2
Metal1(1)	NET CTS_118	
Metal1(1)	NET CTS_118	
Metal2(2)	NET CTS_138	NET n_3089
Metal2(2)	NET CTS_125	NET n_2166
Metal2(2)	NET CTS_49	NET n_2167
Metal1(1)	NET CTS_31	
Metal1(1)	NET CTS_31	
Metal1(1)	NET CTS_73	
Metal1(1)	NET CTS_73	
Metal1(1)	NET CTS_44	
Metal1(1)	NET CTS_44	
Metal2(2)	NET CTS_48	NET n_2924

Παρατηρούμε πως παίρνουμε **56 DRC Violations** για το **NanoRoute** στάδιο και **46 DRC violations** για το στάδιο μετά το **DRC Verification**. Κατά βάση οφείλονται σε **Geometry** ζητήματα, όπως **Metal_EndOfLine_Spacing**, **Metal Short**, **MinHole** & **Parallel_Run_Length_Spacing**. Ακόμα, υπάρχει κι ένα **Connectivity** issue, το οποίο είναι τύπου **Metal Loop**.

Έπειτα, κάνουμε γέμισμα (Fill) τα μέταλλα με ελάχιστη πυκνότητα στο 10%.



Η τελική εικόνα του design μας είναι η ακόλουθη :



ΑΣΚΗΣΗ 2η

Βήματα 1~6:

Εκτελούμε τα βήματα 1 έως 6 όπως στην Άσκηση 1, με τη μόνη διαφορά ότι προσθέτουμε στις εντολές του Genus τις ακόλουθες εντολές, όπως προτείνει κι ο εργαστηριακός οδηγός, οι οποίες πρέπει να τοποθετηθούν μετά από το διάβασμα των αρχείων βιβλιοθηκών και πριν το διάβασμα και το elaboration του design *picorv32.v*, με τις εντολές να είναι οι εξής :

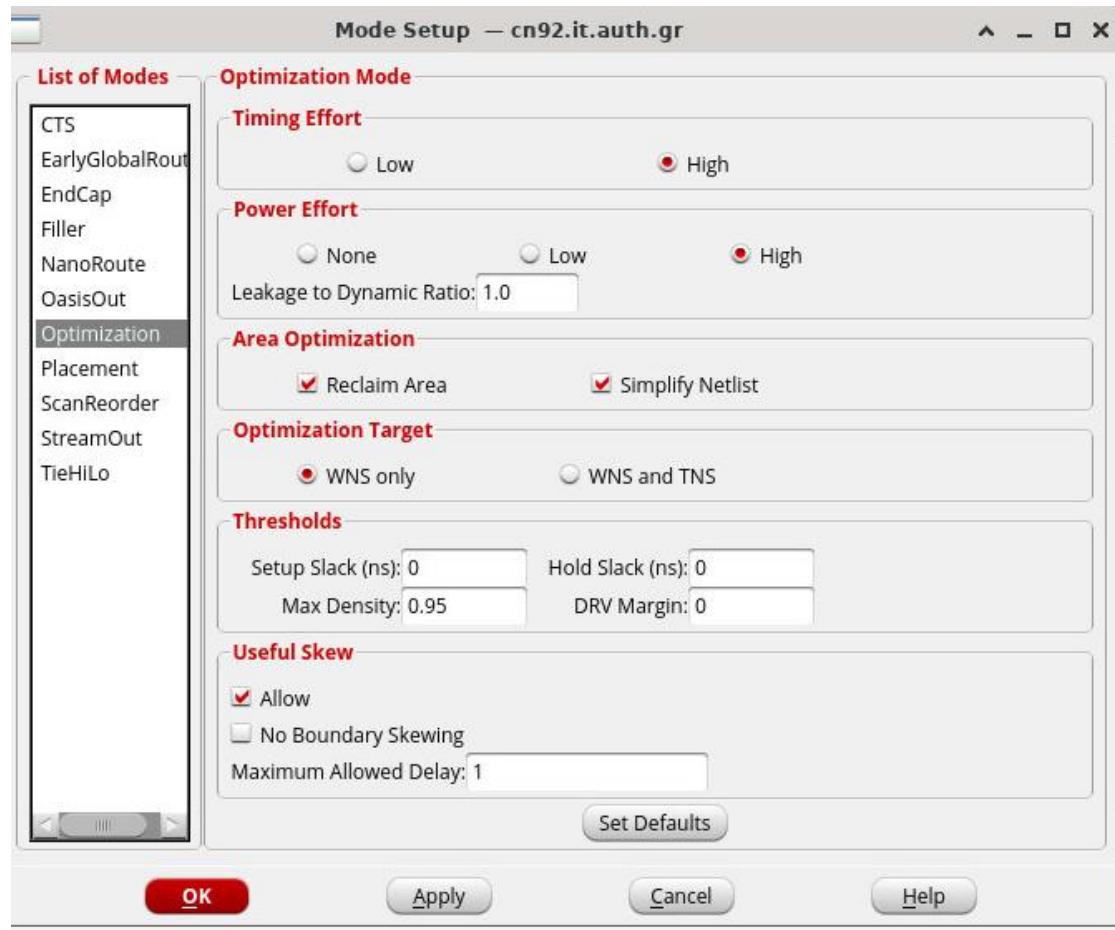
```
#set design power effort to high
set_db design_power_effort high
#set optimization effort to leakage power
set_db opt_leakage_to_dynamic_ratio 1.0
```

Στην ουσία, η 1^η εντολή μεγιστοποιεί τη βελτιστοποίηση ως προς την ισχύ, προσπαθώντας να ελαχιστοποιήσει την τελευταία. Η 2^η εντολή θέτει την ελαχιστοποίηση πλήρως ως προς την **ισχύ διαρροής** έναντι της **δυναμικής ισχύος**. Στη συνέχεια εξάγουμε αναφορές ως προς την ισχύ, επιφάνεια και χρονισμό. Έτσι, έχουμε :

Άσκηση(#)	Leakage Power (W)	Internal Power (W)	Switching Power (W)	Total Power (W)	Area (μm ²)	Slack (ns)
1 ^η	8.46420e-07	1.65971e-03	3.72066e-03	5.38122e-03	48527.595	0
2 ^η	7.35587e-07	1.29599e-03	3.46415e-03	4.76087e-03	52005.151	0
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Αυξήθηκε	Σταθερό

Βήματα 7~11:

Στη συνέχεια, ακολουθούμε την διαδικασία που ακολουθήσαμε και στην 1^η Άσκηση, με τη διαφοροποίηση πως στις ρυθμίσεις της τοποθέτησης ρυθμίζουμε την επιλογή *Power Effort* στο *High*, ενώ αφήνουμε το *Leakage to Dynamic Ratio* στο 1.0 για τους λόγους που αναφέραμε παραπάνω.



Κατόπιν εκτελούμε την τοποθέτηση κι εξάγουμε όπως πριν αναφορές σχετικά με τα στοιχεία του design. Έτσι, έχουμε :

Άσκηση(#)	Leakage Power (mW)	Internal Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	0.000854	1.527	2.269	3.797	34619.976	0.014
2 ^η	0.0006782	1.209	2.256	3.466	37684.638	0.007
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Αυξήθηκε	Μειώθηκε

Bήμα 14:

Μετά τη σύνθεση ρολογιού πραγματοποιούμε ξανά αποτίμηση των χαρακτηριστικών :

Άσκηση(#)	Leakage Power (mW)	Internal Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	0.0008784	1.691	2.485	4.176	35165.124	0.048
2 ^η	0.0007074	1.383	2.478	3.862	38199.690	- 0.020
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Αυξήθηκε	Μειώθηκε

Bήμα 15:

Με το πέρας του Nano Route και το Post Route optimization, αποτιμάμε για μια τελευταία φορά χαρακτηριστικά :

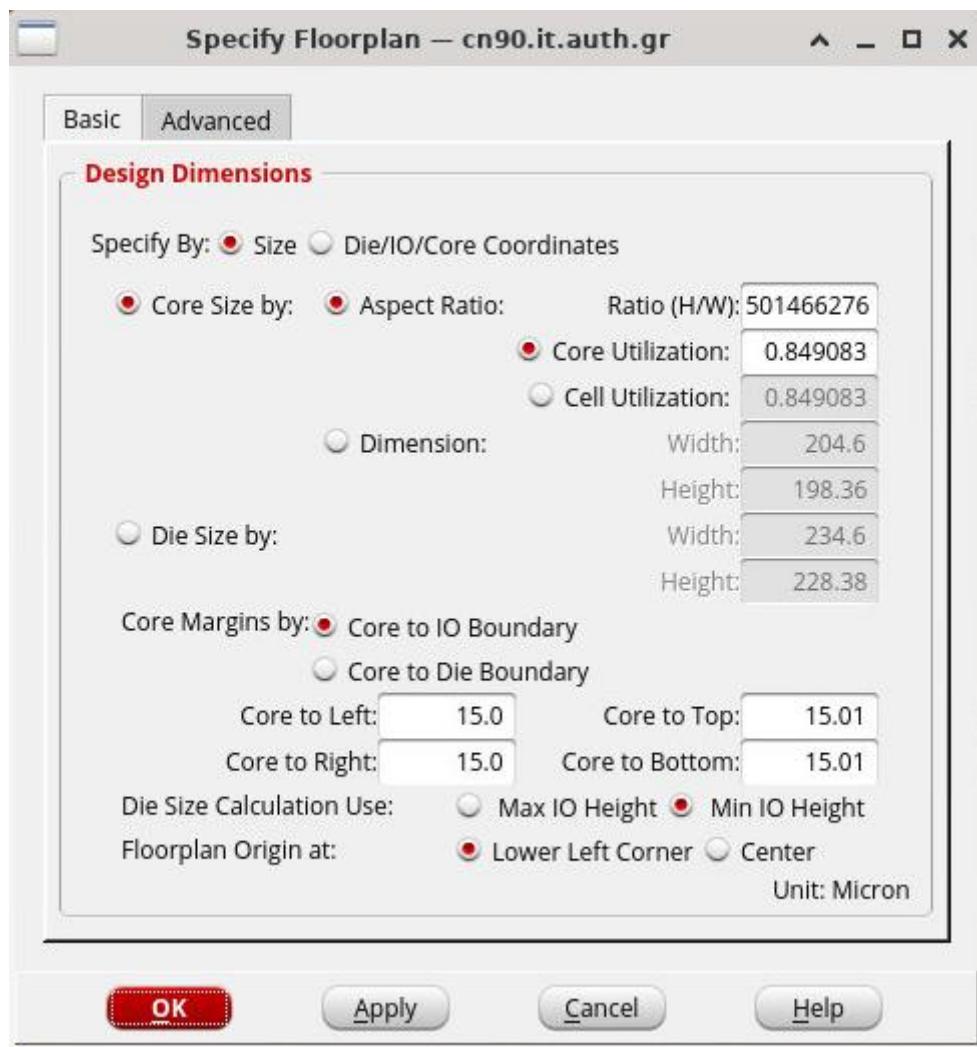
Άσκηση(#)	Leakage Power (mW)	Internal Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	0.0008784	1.691	2.451	4.143	35165.124	0.077
2 ^η	0.0007098	1.385	2.447	3.832	38232.522	0.109
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Αυξήθηκε	Αυξήθηκε

Συνοψίζοντας, βελτιστοποιώντας ως προς την κατανάλωση ισχύος και ιδίως ως προς την ισχύ διαρροής, σε κάθε βήμα είχαμε μείωση ως προς κάθε τύπο ισχύος άρα και της συνολικής ισχύος, αλλά αυτό είχε αρνητική επίπτωση με την αύξηση της επιφάνειας και την μείωση του slack. Ωστόσο, σε αυτό το τελευταίο βήμα, δεν είχαμε μείωση του slack, αλλά αύξησή του σε σχέση με την 1^η Άσκηση, όμως αυτό αποτελεί εξαίρεση σε σχέση με τα προηγούμενα βήματα.

ΑΣΚΗΣΗ 3^η

Βήματα 9~11:

Εκτελούμε όλα τα προηγούμενα βήματα όπως στην 1^η Άσκηση. Στο βήμα 9 επιλέγουμε ποσοστό χρήσης πυρήνα ίσο με 85%.



Μετά το βήμα της τοποθέτησης, εξάγουμε αναφορές σχετικά με τα χαρακτηριστικά :

Άσκηση(#)	Leakage Power (mW)	Internal Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	0.000854	1.527	2.269	3.797	34619.976	0.014
3 ^η	0.00085	1.528	2.266	3.795	34569.018	0.001

Παρατήρηση	Μειώθηκε	Αυξήθηκε	Μειώθηκε	Μειώθηκε	<u>Μειώθηκε</u>	Μειώθηκε
------------	----------	----------	----------	----------	-----------------	----------

Bήματα 13~14:

Μετά τη σύνθεση του ρολογιού κι αφού έχουμε εκτελέσει Early Global Route, έχουμε :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm^2)	Slack (ns)	Wirelength (μm)	Number of Vias
1 ^η	1.691	0.0008784	2.485	4.176	35165.124	0.048	217370	77104
3 ^η	1.69	0.0008751	2.473	4.164	35125.452	0.0320	203040	78121
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	<u>Μειώθηκε</u>	Μειώθηκε	Μειώθηκε	<u>Αυξήθηκε</u>

Bήμα 15:

Τέλος, στη φάση του Post Route και μετά την βελτιστοποίηση του κυκλώματος, αποτιμάμε για μια τελευταία φορά τα χαρακτηριστικά :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm^2)	Slack (ns)
1 ^η	1.691	0.0008784	2.451	4.143	35165.124	0.077
3 ^η	1.69	0.0008752	2.443	4.135	35127.162	0.076
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	<u>Μειώθηκε</u>	Μειώθηκε

Συμπερασματικά, αυξάνοντας το ποσοστό χρήσης του πυρήνα σημαίνει μείωση της επιφάνειας που δεν βρίσκεται σε χρήση από το design κι αυτό μπορεί να επιτευχθεί με αύξηση της επιφάνειας των κελιών και μείωση της συνολικής επιφάνειας του πυρήνα. Πρακτικά, αυτό έχει ως αποτέλεσμα την πυκνότερη τοποθέτηση των κελιών, συνεπώς και μικρότερη μεταξύ τους απόσταση. Επομένως είναι λογικό το ότι μειώθηκε η συνολική επιφάνεια. Ακόμα, λόγω της

μικρότερης απόστασης των συνδεδεμένων κελιών είναι λογικό ότι λάβαμε μικρότερο wirelength, αλλά και αυξημένο αριθμό vias, καθώς η συμφόρηση στο κύκλωμα θα μπορούσε να οδηγήσει σε επικάλυψη κι έτσι το εργαλείο χρησιμοποιεί περισσότερα vias για να επιτύχει τις προδιαγραφές του κυκλώματος και την ενδοεπικοινωνία ανάμεσα στις στρώσεις μετάλλων.

ΑΣΚΗΣΗ 4^η

Bήματα 1~5:

Κάνουμε τα προηγούμενα βήματα όπως στην Άσκηση 1 και στο βήμα παραγωγής των περιορισμών του αρχείου *.sdc (**Βήμα 4**), κάνουμε τις εξής διαφοροποιήσεις για να επιτύχουμε ρολόι συχνότητας 400 MHz , δηλαδή περιόδου 2.5 ns :

```
#1)create a clock named 'clk' with 50% duty cycle and 2.5ns period
```

```
create_clock [get_ports clk] -name clk -period 2.5 -waveform {0 1.25}
```

...

```
#4)set clock transition @ 1% of total period
```

```
set_clock_transition 0.025 [get_clocks clk]
```

...

Bήμα 6:

Αποτιμάμε τα αρχικά χαρακτηριστικά του κυκλώματος :

Άσκηση(#)	Number of Cells	Total Power (W)	Total Area (μm ²)	Slack (ns)
1 ^η	10302	5.38122e-03	48527.595	0

4 ^η	13611	1.31226e-02	65694.604	- 0.969
Παρατήρηση	Aυξήθηκε	Aυξήθηκε	Aυξήθηκε	Μειώθηκε

Με μια πρώτη ματιά, βλέπουμε πως ο υποδιπλασιασμός της περιόδου ρολογιού έχει πολύ δραματικές συνέπειες σε όλα τα βασικά χαρακτηριστικά *PPA* (Power – Performance – Area) του κυκλώματος. Τουλάχιστον σε πρώτη φάση...

Bήματα 7~11:

Μετά το πέρας της τοποθέτησης, εξάγουμε αναφορές :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.527	0.000854	2.269	3.797	34619.976	0.014
4 ^η	4.04	0.001313	5.404	9.445	46560.906	- 0.170
Παρατήρηση	Aυξήθηκε	Aυξήθηκε	Aυξήθηκε	Aυξήθηκε	Aυξήθηκε	Μειώθηκε

Bήμα 14:

Στη συνέχεια αφού συνθέσουμε το δέντρο ρολογιού και κάνουμε *PostCTS optimization* εξάγουμε πάλι αναφορές :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.485	4.176	35165.124	0.048
4 ^η	4.495	0.001388	5.986	10.48	48333.492	- 0.163
Παρατήρηση	Aυξήθηκε	Aυξήθηκε	Aυξήθηκε	Aυξήθηκε	Aυξήθηκε	Μειώθηκε

Bήμα 15:

Μετά το *NanoRoute* και το *PostRoute optimization*, αποτιμάμε μια τελευταία φορά τα χαρακτηριστικά του κυκλώματος :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.451	4.143	35165.124	0.077
4 ^η	4.497	0.001389	5.866	10.36	48354.354	- 0.140
Παρατήρηση	Αυξήθηκε	Αυξήθηκε	Αυξήθηκε	Αυξήθηκε	Αυξήθηκε	Μειώθηκε

Συμπερασματικά, ο υποδιπλασιασμός της περιόδου δημιουργεί μια σειρά από αρνητικές συνέπειες στο κύκλωμα. Αρχικά, δεν επιτυγχάνονται οι στόχοι χρονισμού, αφού σε όλα τα βήματα λαμβάνουμε αρνητικό slack (και μάλιστα από το αρχείο *Quality of Results* στο Βήμα 6) βλέπουμε ότι έχουμε 149 *violating paths*. Ακόμα, ο διπλασιασμός της συχνότητας του κυκλώματος είχε ως αρνητική συνέπεια την τεράστια αύξηση κάθε τύπου ισχύος σε όλα τα βήματα και την αύξηση της συνολικής επιφάνειας. Περισσότερο αναμενόμενη ήταν η αύξηση της μεταβατικής ισχύος, λόγω των αυξήσεων των μεταβάσεων στο ίδιο χρονικό διάστημα. Λαμβάνοντας υπόψη όλες τις παραπάνω παρατηρήσεις, ο σχεδιασμός ρολογιού **με συχνότητα 400 MHz** δε συνίσταται για το υπάρχον design με τις συγκεκριμένες σχεδιαστικές επιλογές.

ΑΣΚΗΣΗ 5^η

Βήματα 1~6:

Για τα πρώτα 6 βήματα, τροποποιούμε το *.tcl αρχείο, ώστε να συμπεριλάβουμε τις παρακάτω εντολές :

```
#start of file

...
#adding slow_vdd1v0_multibitsDFF.lib
set_db library {./timing/slow_vdd1v0_basicCells.lib
./timing/slow_vdd1v0_multibitsDFF.lib}

#adding gsclib045_multibitsDFF.lef
set_db lef_library {./lef/gsclib045_tech.lef
./lef/gsclib045_macro.lef ./lef/gsclib045_multibitsDFF.lef}

read_qrc ./qrc/qx/gpdk045.tch

#use multibit cells
set_db use_multibit_cells true
read_hdl picorv32.v

...
#end of file
```

Στο τέλος του 6^{ου} βήματος εξάγουμε αναφορές σχετικές με τα αρχικά χαρακτηριστικά του κυκλώματος. Έτσι, έχουμε :

Άσκηση(#)	Number of Cells	Total Power (W)	Total Area (μm^2)	Slack (ns)
1 ^η	10302	5.38122e-03	48527.595	0
5 ^η	10166	5.09122e-03	49711.559	0
Παρατήρηση	Μειώθηκε	Μειώθηκε	Αυξήθηκε	Σταθερό

Bήματα 7~11:

Μετά την εισαγωγή του design στο Innovus και στο πέρας του Placement αποτιμάμε πάλι τα χαρακτηριστικά :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.527	0.000854	2.269	3.797	34619.976	0.014
5 ^η	1.259	0.0008754	2.302	3.562	34458.894	0.023
Παρατήρηση	Μειώθηκε	Αυξήθηκε	Αυξήθηκε	Μειώθηκε	Μειώθηκε	Αυξήθηκε

Bήμα 14:

Εκτελούμε τη σύνθεση του δέντρου ρολογιού κι έπειτα αποτιμάμε τα PPA χαρακτηριστικά αλλά και το *Clock Tree Report* του design.

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.485	4.176	35165.124	0.048
5 ^η	1.32	0.0008857	2.378	3.699	34692.138	0.002
Παρατήρηση	Μειώθηκε	Αυξήθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε

Άσκηση(#)	Number of Buffers	Buffering Area (μm ²)	Min Depth	Max Depth	Trunk Wirelength (μm)	Leaves Wirelength (μm)
1 ^η	245	546.858	6	6	2210.140	7218.685
5 ^η	91	202.122	5	5	1251.010	3374.450
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε

Bήμα 15:

Μετά το *NanoRoute* και το *PostRoute optimization*, αποτιμάμε μια τελευταία φορά τα PPA χαρακτηριστικά του κυκλώματος :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.451	4.143	35165.124	0.077
5 ^η	1.321	0.0008857	2.353	3.674	34692.138	0.009
Παρατήρηση	Μειώθηκε	Αυξήθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε

Συμπερασματικά, με την εισαγωγή των βιβλιοθηκών ***slow_vdd1vo_multibitsDFF.lib*** & ***gsclibo45_multibitsDFF.lef*** στο design , τελικώς μειώθηκε δραστικά η συνολική ισχύς του κυκλώματος, ενώ επίσης μειώσεις υπήρξαν ελαφρώς στην συνολική επιφάνεια αλλά και στο slack του κυκλώματος. Ακόμη, η σύγκριση στο **Bήμα 14** ως προς το δέντρο ρολογιού είχε αναμενόμενα αποτελέσματα, καθώς με τη χρήση των *multibits* βιβλιοθηκών, περιμέναμε την ένωση και χρήση συγχωνευμένων καταχωρητών, γεγονός που αποτυπώθηκε στον μειωμένο αριθμό των ***buffers*** που χρησιμοποιήθηκαν στο δέντρο ρολογιού αλλά και στο μικρότερο ***wirelength*** που χρειάστηκε για τη διασύνδεσή τους. Τέλος, παρατηρήσαμε πως αυτό προκάλεσε και μείωση στο μέγιστο κι ελάχιστο λογικό βάθος του δέντρου, μειώνοντας το κατά 1 σε σχέση με αυτό της 1^{ης} Άσκησης.

ΑΣΚΗΣΗ 6η

Βήματα 1~6:

Για τα πρώτα 6 βήματα, τροποποιούμε το *.tcl αρχείο, ώστε να συμπεριλάβουμε τις παρακάτω εντολές :

```
#start of file

...
read_hdl picorv32.v

#insert clock gating to design
set_db lp_insert_clock_gating true
#continue with elaboration
elaborate "picorv32"

...
#create reports for clock gating
report_clock_gating >
/home/n/nikolaoac/Desktop/Reports/Clock_Gating

...
#end of file
```

Σε πρώτη φάση, παραθέτουμε ως συνήθως τα αρχικά κύρια χαρακτηριστικά του κυκλώματος ύστερα από την εξαγωγή αναφορών, με εμφάνιση όλων των τύπων ισχύος :

Άσκηση(#)	Leakage Power (W)	Internal Power (W)	Switching Power (W)	Total Power (W)	Area (μm ²)	Slack (ns)
1 ^η	8.46420e-07	1.65971e-03	3.72066e-03	5.38122e-03	48527.595	0
6 ^η	8.03244e-07	6.98557e-04	2.28945e-03	2.98881e-03	43224.905	0
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Σταθερό

Επίσης δημιουργήσαμε την αναφορά για το Clock Gating :

```

└ Summary
  └ -----
    └ -----
      ┌ Category          Number   %     Average Toggle Saving %
      └ -----
      ┌ Total Clock Gating Instances      58  100.00
      └ -----
      ┌ RC Clock Gating Instances        58  100.00           79.43
      ┌ Non-RC Clock Gating Instances    0   0.00            0.00
      └ -----
      ┌ RC Gated Flip-flops             1706 87.00           83.57
      ┌ Non-RC Gated Flip-flops         0   0.00            0.00
      └ -----
      ┌ Total Gated Flip-flops         1706 87.00
      ┌ Total Ungated Flip-flops       255  13.00
      ┌ Enable not found              238  93.33
      ┌ Register bank width too small 17   6.67
      └ -----
      ┌ Total Flip-flops               1961 100.00
      └ -----
      ┌ Multibit Flip-flop Summary
      └ -----
      ┌ Width    Number   Bits   RC Gated   Ungated
      └ -----
      ┌ 1-bit    1961     1961   1706 (87.00%) 255 (13.00%)
      └ -----
    └ -----
  └ -----

```

Παρατηρούμε πως δημιουργούνται **58 στοιχεία φραγής ρολογιού** ενώ φράζονται τα **1706 flip flops** από τα συνολικά **1961** που υπάρχουν.

Βήματα 7~11:

Μετά την τοποθέτηση, αποτιμάμε τα χαρακτηριστικά :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.527	0.000854	2.269	3.797	34619.976	0.014
6 ^η	1.005	0.0008035	2.185	3.192	30437.658	0.013

Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε
------------	----------	----------	----------	----------	----------	----------

Ας ρίξουμε μια ματιά στην αναφορά για την **επιφάνεια** που δημιουργήθηκε για το βήμα 11 :

Hinst Name	Module Name	Inst Count	Total Area
picorv32		10356	30437.658
RC_CG_HIER_INST0	RC_CG_MOD	1	6.498
RC_CG_HIER_INST1	RC_CG_MOD_1	1	6.498
RC_CG_HIER_INST10	RC_CG_MOD_10	1	6.498
RC_CG_HIER_INST11	RC_CG_MOD_11	1	6.498
RC_CG_HIER_INST12	RC_CG_MOD_12	1	6.498
RC_CG_HIER_INST13	RC_CG_MOD_13	1	6.498
RC_CG_HIER_INST14	RC_CG_MOD_14	1	6.498
RC_CG_HIER_INST15	RC_CG_MOD_15	1	6.498
RC_CG_HIER_INST16	RC_CG_MOD_16	1	6.498
RC_CG_HIER_INST17	RC_CG_MOD_17	1	6.498
RC_CG_HIER_INST18	RC_CG_MOD_18	1	6.498

Παρατηρούμε πως δημιουργήθηκαν διακόπτες φραγής ρολογιού, οι οποίοι καταλαμβάνουν συγκεκριμένη επιφάνεια.

Ας πάμε, τώρα, να ρίξουμε μια ματιά στην αναφορά για την **ισχύ** που δημιουργήθηκε :

Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)
Sequential	0.4379	0.07672	0.0002436	0.5148	16.13
Macro	0	0	0	0	0
IO	0	0	0	0	0
Combinational	0.5379	2.109	0.0005532	2.647	82.94
Clock (Combinational)	0	0	0	0	0
Clock (Sequential)	0.0296	0	6.738e-06	0.0296	0.9275
Total	1.005	2.185	0.0008035	3.192	100

Παρατηρούμε πως η χρήση φραγής ρολογιού, δημιουργεί την **ακολουθιακή ισχύ ρολογιού** του κυκλώματος. Η **συνδυαστική ισχύς ρολογιού** θα δημιουργηθεί με το **Βήμα 14** και το δίκτυο διανομής ρολογιού.

Βήμα 14:

Προχωρώντας σε αποτίμηση των χαρακτηριστικών στο στάδιο PostCTS, έχουμε :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.485	4.176	35165.124	0.048
6 ^η	1.096	0.0008384	2.265	3.361	31192.110	0.024
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε

Bήμα 15:

Τέλος, στο PostRoute στάδιο, εξάγουμε για τελευταία φορά τα χαρακτηριστικά του κυκλώματος :

Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.451	4.143	35165.124	0.077
6 ^η	1.097	0.0008388	2.245	3.343	31197.582	0.101
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Αυξήθηκε

Συμπερασματικά, μας δόθηκε η ευκαιρία να επαληθεύσουμε πως η **Φραγή Ρολογιού** (Clock Gating) είναι μια πολύ ισχυρή τεχνική μείωσης ισχύος. Σκοπό της αποτελεί η μείωση της **δυναμικής** και πιο συγκεκριμένα της **Ισχύος Μετάβασης** (Switching Power). Παρατηρούμε πως, σε όλα τα βήματα, υπήρξε μια δραστική μείωση σε όλους του τύπους ισχύος κι όχι μόνο στην ισχύ μετάβασης. Ακόμη, υπήρξε σημαντική μείωση της συνολικής επιφάνειας σε όλα τα βήματα, ενώ μονάχα στο PostRoute στάδιο (Βήμα 15) υπήρξε το μεγαλύτερο slack της 5^{ης} Άσκησης συγκρινόμενο με τον εαυτό της αλλά και με αυτά της Άσκησης 1.

ΑΣΚΗΣΗ 7η

Βήματα 1~6:

Για τα πρώτα 7 βήματα, τροποποιούμε το *.tcl αρχείο, ώστε να συμπεριλάβουμε τις παρακάτω εντολές :

```
...
elaborate "picorv32"
#Verification rtl vs elab step
write_netlist -lec > elab.v
write_do_lec -top picorv32 -golden_design rtl -revised_design
elab.v -log_file rtl_elab.lec.log > rtl_elab.do
...
syn_generic
# Verification rtl vs generic step
write_netlist -lec > generic.v
write_do_lec -top picorv32 -golden_design rtl -revised_design
generic.v -log_file elab_generic.lec.log > elab_generic.do
syn_map
# Verification rtl vs fv_map step
write_do_lec -top picorv32 -golden_design rtl -revised_design
fv_map -log_file generic_fvmap.lec.log > generic_fvmap.do
syn_opt
...
```

Στη συνέχεια, τρέχουμε σε διαφορετικό *command line* τις παρακάτω εντολές :

```
lec -XL -nogui -dofile rtl_elab.do  
lec -XL -nogui -dofile generic_fvmap.do
```

Στη συνέχεια παίρνουμε αναφορές, οι οποίες συγκρίνουν το design σε μορφή RTL κώδικα κι επιπέδου λειτουργικότητας (*synthesized netlist*) μέσω του εργαλείου *Conformal Logic Equivalence Checking* της Cadence. Πιο συγκεκριμένα, αρχικά συγκρίνουμε τον RTL κώδικα με τη μορφή του design που πήραμε μετά το βήμα του *elaboration* το οποίο επιτυγχάνεται με την 1^η από τις δύο παραπάνω εντολές που διατυπώσαμε. Πηγαίνουμε στα αντίστοιχα sections των commands : **report_verification -hier -verbose & report_statistics** κι έτσι λαμβάνουμε το ακόλουθο περιεχόμενο :

=====		Verification Report	=====
Category	Count		
1. Non-standard modeling options used:	0		
Tri-stated output:	checked		
Revised X signals set to E:	yes		
Floating signals tied to Z:	yes		
Command "add clock" for clock-gating:	not used		
2. Incomplete verification:	0		
All primary outputs are mapped:	yes		
Not-mapped DFF/DLAT is detected:	no		
All mapped points are added as compare points:	yes		
All compared points are compared:	yes		
User added black box:	no		
Black box mapped with different module name:	no		
Empty module is not black boxed:	no		
Command "add ignore outputs" used:	no		
Always false constraints detected:	no		
Verified pin-equivalent outputs are unmapped:	no		
3. User modification to design:	0		
Change gate type:	no		
Change wire:	no		
Primary input added by user:	no		
4. Conformal Constraint Designer clock domain crossing checks recommended: 1			
RTL5.4 Partial case items in full case statement:	used *		
Multiple clocks in the design:	no		
5. Design ambiguity:	0		
Duplicate module definition:	no		
Black box due to undefined cells:	no		
Golden design has abnormal ratio of unreachable gates:	no		
Ratio of golden unreachable gates:	0%		
Revised design has abnormal ratio of unreachable gates:	no		
Ratio of revised unreachable gates:	0%		
All primary input bus ordering is consistent:	yes		
All primary output bus ordering is consistent:	yes		
DFF/DLAT not compared due to disabled clock port(s):	0		
Always X compared point is detected:	not checked		
6. Compare Results:	PASS		
	Total Equivalent modules = 3		
=====			

	Compare Result	Golden	Revised
Root module name		picorv32	picorv32
Primary inputs		102	102
Mapped		102	102
Undriven key points		0	565
Unmapped		0	565
Extra		0	565
Primary outputs		307	307
Mapped		307	307
Equivalent	307		
Black-box key points		2	2
Mapped		2	2
Equivalent	2		
State key points		1755	1755
Mapped		1753	1753
Equivalent	1753		
Unmapped		2	2
Unreachable		2	2

Παρατηρούμε ότι το *Verification Report* δίνει ως αποτέλεσμα σύγκρισης **PASS** σε 3 equivalent (ισοδύναμα) modules, ενώ το section *Report Statistics* δίνει σύνολο **2062 EQ (equivalent) points**. Πιο συγκεκριμένα είναι **307 Primary Outputs**, **1753 DFFs** και **2 Black Box Modules (BBOX)**.

Στη συνέχεια, συγκρίνουμε τον RTL κώδικα με τη μορφή του design που πήραμε μετά το βήμα του *mapping* το οποίο επιτυγχάνεται με την 2^η από τις δύο παραπάνω εντολές που διατυπώσαμε. Πηγαίνουμε στα αντίστοιχα sections των commands : **report_verification -verbose & report_statistics** κι έτσι λαμβάνουμε το ακόλουθο περιεχόμενο :

===== Verification Report =====		
Category	Count	
1. Non-standard modeling options used:		0
Tri-stated output:	checked	
Revised X signals set to E:	yes	
Floating signals tied to Z:	yes	
Command "add clock" for clock-gating:	not used	
2. Incomplete verification:		0
All primary outputs are mapped:	yes	
Not-mapped DFF/DLAT is detected:	no	
All mapped points are added as compare points:	yes	
All compared points are compared:	yes	
User added black box:	no	
Black box mapped with different module name:	no	
Empty module is not black boxed:	no	
Command "add ignore outputs" used:	no	
Always false constraints detected:	no	
3. User modification to design:		0
Change gate type:	no	
Change wire:	no	
Primary input added by user:	no	
4. Conformal Constraint Designer clock domain crossing checks recommended:	0	
Multiple clocks in the design:	no	
5. Design ambiguity:		0
Duplicate module definition:	no	
Black box due to undefined cells:	no	
Golden design has abnormal ratio of unreachable gates:	no	
Ratio of golden unreachable gates:	0%	
Revised design has abnormal ratio of unreachable gates:	no	
Ratio of revised unreachable gates:	0%	
All primary input bus ordering is consistent:	yes	
All primary output bus ordering is consistent:	yes	
DFF/DLAT not compared due to disabled clock port(s):	0	
Always X compared point is detected:	not checked	
6. Compare Results:		PASS
Number of EQ compare points:	2268	
Number of NON-EQ compare points:	0	
Number of Aborted compare points:	0	
Number of Uncompared compare points :	0	

```
// Command: report_statistics
Mapping and compare statistics
=====
Compare Result      Golden      Revised
-----
```

Root module name	picorv32	picorv32
Primary inputs	102	102
Mapped	102	102
Primary outputs	307	307
Mapped	307	307
Equivalent	307	
State key points	1961	1967
Mapped	1961	1961
Equivalent	1961	
Unmapped	0	6
Unreachable	0	6

Παρατηρούμε ότι αυτή η σύγκριση δίνει **2268 EQ Points** με τη σήμανση **PASS**. Πιο συγκεκριμένα είναι **307 Primary Outputs** και **1961 DFFs**.

Άρα, σε κάθε περίπτωση, μετά τη σύνθεση, το εξαγόμενο design σε επίπεδο λειτουργικότητας είναι λογικά ισοδύναμο με τον αρχικό κώδικα σε μορφή RTL. Άρα το εργαλείο μας δίνει το ‘πράσινο φανάρι’ για τη συνέχεια υλοποίησης του κυκλώματος.

ΑΣΚΗΣΗ 8η

Βήματα 8~15:

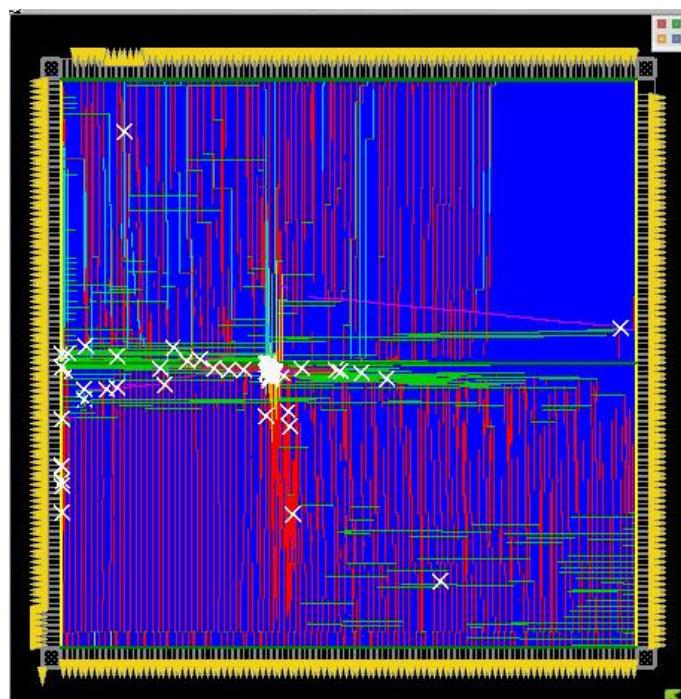
Αρχικά εκτελούμε το *pads.py* script που μας δόθηκε στα πλαίσια του μαθήματος για να πάρουμε την επιθυμητή έξοδο την οποία και θα χρησιμοποιήσουμε στη συνέχεια. Τροποποιούμε το αρχείο εξόδου *genus.v* από το 7^ο Βήμα του Genus σύμφωνα με τον εργαστηριακό οδηγό και την έξοδο από το *pads.py* script. Εισάγουμε το design στο Innovus μαζί με τις βιβλιοθήκες **giolibo45.lef** και **pads_SS_s1vg.lib**, οι οποίες είναι απαραίτητες για τα pads. Αποθηκεύουμε το *.io file που παράγει το Innovus και στη συνέχεια το τροποποιούμε αλλάζοντας τις θέσεις των **pad corners** όπως ορίζει ο εργαστηριακός οδηγός κι έπειτα το εισάγουμε εκ νέου στο εργαλείο. Εκτελούμε όλα τα βήματα μέχρι το Βήμα 15 στο Innovus (όπως κάναμε και στις προηγούμενες ασκήσεις) και εξάγουμε αναφορές σχετικά με τα χαρακτηριστικά του κυκλώματος :

Ασκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.451	4.143	35165.124	0.077
8 ^η	160.4	23.44	4.282	188.1	6196634.172	- 17.155
Παρατήρηση	Αυξήθηκε	Αυξήθηκε	Αυξήθηκε	Αυξήθηκε	Αυξήθηκε	Μειώθηκε

picorv32_chip		18795	6196634.172
example	picorv32	16803	44375.526
iopads	picorv32_pads	1982	6152186.826

Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage (%)
Sequential	0.88	0.0962	0.0002361	0.9764	0.5191
Macro	0	0	0	0	0
IO	158.2	1.778	23.43	183.5	97.54
Combinational	0.8652	2.025	0.001237	2.891	1.537
Clock (Combinational)	0.3747	0.3833	5.668e-05	0.7581	0.4031
Clock (Sequential)	0	0	0	0	0
Total	160.4	4.282	23.44	188.1	100

Είναι σημαντικό να παρατηρήσουμε πως η συντριπτική πλειοψηφία της συνολικής επιφάνειας και συνολικής ισχύος δημιουργείται από τα *IO Pads*. Συνοψίζοντας, η επιφάνεια των pads είναι πολύ μεγαλύτερη από αυτή του πυρήνα του design, όπως υποδεικνύει και *area report* πιο πάνω. Προκειμένου να μην δημιουργηθούν πολύ μεγάλες διασυνδέσεις στο κύκλωμα κι έτσι αυξηθεί η αντίσταση διασυνδέσεων, η ισχύς και οι RC καθυστερήσεις, το Innovus επιλέγει να συνεχίσει να διοχετεύει τα κελιά σε ένα μικρό ποσοστό του χώρου όπως υποδεικνύει και το report. Τέλος, τα IO pads έχοντας μονοπωλήσει την ισχύ και την επιφάνεια, έχουν δημιουργήσει κι ένα αρκετά σημαντικό αρνητικό slack, προκαλώντας παραβάσεις χρονισμού. Ένα πρόβλημα που ακόμα συνάντησα στην άσκηση είναι ότι έλαβα **CoreSiteDouble errors** που εμφανίστηκαν μετά την *preCTS optimization* οι οποίες αναφέρονταν ως errors στην έξοδο του εργαλείου, αλλά δεν επηρέασαν την έκβαση όλων των υπόλοιπων βημάτων. Ισως έφταιγε ότι όρισα δύο φορές το *Core Utilization* από το αντίστοιχο παράθυρο και ίσως υπήρξε διπλή εγγραφή. Η τελική εικόνα του design μετά τη δρομολόγηση είναι η ακόλουθη :



ΑΣΚΗΣΗ 9η

Βήματα 1~6:

Τροποποιούμε το *.tcl αρχείο ακολούθως, προκειμένου να προσθέσουμε **Design For Testability** (DFT) λογική στο design μας :

```
#start of file

...
elaborate "picorv32"

#Adding DFT logic

set_db / .dft_scan_style muxed_scan
set_db / .dft_prefix DFT_
set_db / .dft_identify_top_level_test_clocks true
set_db / .dft_identify_test_signals true
set_db / .dft_identify_internal_test_clocks false
set_db / .use_scan_seqs_for_non_dft false
set_db "design:picorv32" .dft_scan_map_mode tdrc_pass
set_db "design:picorv32"
.set_connect_shift_enable_during_mapping tie_off
set_db "design:picorv32"
.set_connect_scan_data_pins_during_mapping loopback
set_db "design:picorv32" .dft_scan_output_preference auto
set_db "design:picorv32" .dft_lockup_element_type
preferred_level_sensitive
set_db "design:picorv32" .dft_mix_clock_edges_in_scan_chains
true
define_test_clock -name scanclk -period 20000 clk
define_shift_enable -name se -active high -create_port se
```

```

define_test_mode -name test_mode -active high -create_port
test_mode

define_scan_chain -name top_chain -sdi scan_in -sdo scan_out -
shift_enable se -create_ports

check_design -all >
/home/n/nikolaoac/Desktop/Check/Check_Design

#creating sdc file

create_clock [get_ports clk] -name clk -period 5 -waveform {0
2.5}

...
read_sdc Constraints9.sdc

check_timing_intent >
/home/n/nikolaoac/Desktop/Check/Check_Time_Intent

#export DFT reports

check_dft_rules >
/home/n/nikolaoac/Desktop/Reports/Ex9/DFT_Rules

report_scan_registers >
/home/n/nikolaoac/Desktop/Reports/Ex9/Scan_Registers

report_scan_setup >
/home/n/nikolaoac/Desktop/Reports/Ex9/Scan_Setup

syn_generic

#exporting reports after generic

...
syn_map

#exporting reports after mapping

...
syn_opt

#exporting reports after optimization

...

```

```

#exporting DFT reports after synthesis

check_dft_rules >
/home/n/nikolaoac/Desktop/Reports/Ex9/DFT_Rules_afterSyn

report_scan_registers >
/home/n/nikolaoac/Desktop/Reports/Ex9/Scan_Registers_afterSyn

report_scan_setup >
/home/n/nikolaoac/Desktop/Reports/Ex9/Scan_Setup_afterSyn

check_dft_rules -advanced >
/home/n/nikolaoac/Desktop/Reports/Ex9/DFT_Rule_Advanced

connect_scan_chains -auto_create_chains

report_scan_chains >
/home/n/nikolaoac/Desktop/Reports/Ex9/Scan_Chains

#end of file

```

Σε πρώτη φάση, μετά το στάδιο *Check Timing Intent*, εξάγουμε αναφορές που μας δείχνουν αν υπήρχαν παραβάσεις κατά την εισαγωγή των DFT κανόνων στο design. Ανοίγουμε πρώτα το αρχείο *DFT_Rules*:

```

Detected 0 DFT rule violation(s)
Summary of check_dft_rules
*****
Number of usable scan cells: 48
Clock Rule Violations:
-----
    Internally driven clock net: 0
    Tied constant clock net: 0
    Undriven clock net: 0
    Conflicting async & clock net: 0
    Misc. clock net: 0

Async. set/reset Rule Violations:
-----
    Internally driven async net: 0
    Tied active async net: 0
    Undriven async net: 0
    Misc. async net: 0

Total number of DFT violations: 0

Total number of Test Clock Domains: 1
DFT Test Clock Domain: scanclk
    Test Clock 'scanclk' (Positive edge) has 2090 registers
Number of user specified non-Scan registers: 0
    Number of registers that fail DFT rules: 0
    Number of registers that pass DFT rules: 2090
Percentage of total registers that are scannable: 100%

```

Παρατηρούμε πως και οι **2090 Scan καταχωρητές** περνάνε το τεστ των κανόνων DFT. Αυτό μπορεί να συναχθεί κι από το αρχείο *Scan_Registers* :

```
trace_data_reg[31]    PASS; Test clock: clk/rise,
trace_data_reg[32]    PASS; Test clock: clk/rise;
trace_data_reg[33]    PASS; Test clock: clk/rise;
trace_data_reg[34]    PASS; Test clock: clk/rise;
trace_data_reg[35]    PASS; Test clock: clk/rise;
trace_data_reg[3]     PASS; Test clock: clk/rise;
trace_data_reg[4]     PASS; Test clock: clk/rise;
trace_data_reg[5]     PASS; Test clock: clk/rise;
trace_data_reg[6]     PASS; Test clock: clk/rise;
trace_data_reg[7]     PASS; Test clock: clk/rise;
trace_data_reg[8]     PASS; Test clock: clk/rise;
trace_data_reg[9]     PASS; Test clock: clk/rise;
trace_valid_reg       PASS; Test clock: clk/rise;
trap_reg             PASS; Test clock: clk/rise;
porting registers that fail DFT rules
porting registers that are preserved or marked dont-scan
porting registers that are marked Abstract Segment Dont Scan
porting registers that are part of shift register segments
porting registers that are identified as lockup elements
porting registers that are level-sensitive elements
porting misc. non-scan registers
summary:
total registers that pass DFT rules: 2090
total registers that fail DFT rules: 0
total registers that are marked preserved or dont-scan: 0
total registers that are marked Abstract Segment dont-scan: 0
total registers that are part of shift register segments: 0
total registers that are lockup elements: 0
total registers that are level-sensitive: 0
total registers that are misc. non-scan: 0
```

Στη συνέχεια ελέγχουμε τον χρονισμό του κυκλώματος μέσω του αρχείου *Check_Time_Intent* και παρατηρούμε πως είμαστε συνεπείς ως προς το χρονισμό του design :

```

1
2
3 Lint summary
4 Unconnected/logic driven clocks 0
5 Sequential data pins driven by a clock signal 0
6 Sequential clock pins without clock waveform 0
7 Sequential clock pins with multiple clock waveforms 0
8 Generated clocks without clock waveform 0
9 Generated clocks with incompatible options 0
0 Generated clocks with multi-master clock 0
1 Paths constrained with different clocks 0
2 Loop-breaking cells for combinational feedback 0
3 Nets with multiple drivers 0
4 Timing exceptions with no effect 0
5 Suspicious multi_cycle exceptions 0
6 Pins/ports with conflicting case constants 0
7 Inputs without clocked external delays 0
8 Outputs without clocked external delays 0
9 Inputs without external driver/transition 0
0 Outputs without external load 0
1 Exceptions with invalid timing start-/endpoints 0
2
3                                     Total: 0
4

```

Αποτιμάμε τα κύρια χαρακτηριστικά μετά από τα στάδια *syn_map* και *syn_opt* :

Μετά από στάδιο (άσκηση #)	Number of Cells	Total Power (W)	Total Area (μm^2)	Slack (ns)
Σύνθεση (1 ^η)	10302	5.38122e-03	48527.595	0
<i>syn_map</i> (9 ^η)	12028	5.47469e-03	54510.599	0.001
<i>syn_opt</i> (9 ^η)	11599	5.54766e-03	53863.030	0

Είναι λογική η αύξηση του αριθμού των κελιών, της συνολικής ισχύος και της συνολικής επιφάνειας, καθώς εισάγουμε λογική στο design μας με τη μορφή DFT για να μπορέσουμε σε μετέπειτα στάδιο να κάνουμε όλα τα επιθυμητά tests στο design. Επίσης το η *syn_opt* βελτιστοποιεί το κύκλωμα σε σχέση με το *mapping* στάδιο, γεγονός που είναι αναμενόμενο κι αποτυπώνεται στα χαρακτηριστικά, ενώ το slack και η συνολική ισχύς παραμένουν σχετικά σταθερά ανάμεσα σε αυτά τα στάδια.

Με το πέρας της σύνθεσης ανοίγουμε τα αρχεία *DFT_Rule_Advanced* και *Scan_Registers_afterSyn* που εξήγαμε με τις τελευταίες εντολές:

```
-----
    Internally driven clock net: 0
        Tied constant clock net: 0
            Undriven clock net: 0
        Conflicting async & clock net: 0
            Misc. clock net: 0

Async. set/reset Rule Violations:
-----
    Internally driven async net: 0
        Tied active async net: 0
            Undriven async net: 0
            Misc. async net: 0

Advanced DFT Rule Violations:
-----
    Tristate net contention violation: 0
    Potential race condition violation: 0
        X-source violation: 0

Warning: There are a total of 1 undriven pins which may act as

    Total number of DFT violations: 0

Total number of Test Clock Domains: 1
    DFT Test Clock Domain: scanclk
        Test Clock 'scanclk' (Positive edge) has 1961 registers
    Number of user specified non-Scan registers: 0
        Number of registers that fail DFT rules: 0
        Number of registers that pass DFT rules: 1961
    Percentage of total registers that are scannable: 100%
```

```

reg_pc_reg[0]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[1]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[2]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[3]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[4]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[5]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[6]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[7]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[8]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_pc_reg[9]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_sh_reg[0]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_sh_reg[1]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_sh_reg[2]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_sh_reg[3]      PASS; Test clock: clk/rise; Mapped for DFT;
reg_sh_reg[4]      PASS; Test clock: clk/rise; Mapped for DFT;
trap_reg          PASS; Test clock: clk/rise; Mapped for DFT;

Reporting registers that fail DFT rules
Reporting registers that are preserved or marked dont-scan
Reporting registers that are marked Abstract Segment Dont Scan
Reporting registers that are part of shift register segments
Reporting registers that are identified as lockup elements
Reporting registers that are level-sensitive elements
Reporting misc. non-scan registers

Summary:
Total registers that pass DFT rules: 1961
Total registers that fail DFT rules: 0
Total registers that are marked preserved or dont-scan: 0
Total registers that are marked Abstract Segment dont-scan: 0
Total registers that are part of shift register segments: 0
Total registers that are lockup elements: 0
Total registers that are level-sensitive: 0
Total registers that are misc. non-scan: 0

```

Παρατηρούμε πως **οι καταχωρητές** μειώθηκαν σε **1961** σε σχέση με τα προηγούμενα αρχεία, γεγονός που οφείλεται στα στάδια της βελτιστοποίησης. Υπάρχει στο αρχείο *DFT_Rule_Advanced* ένα Warning σχετικά με ένα *Undriven Pin*, αλλά σε γενικές γραμμές, όλοι οι καταχωρητές ικανοποιούν τις DFT συνθήκες, επομένως το design μας είναι ικανό να δεχθεί tests μετά την υλοποίησή του. Τέλος, το αρχείο *Scan_Chains* μας διαβεβαιώνει ότι όλες οι αλυσίδες ανίχνευσης έχουν διασυνδεθεί.

ΑΣΚΗΣΗ 10η

Βήματα 1~7:

Θα επιλέξουμε να υλοποιήσουμε το design μας κάνοντας χρήση **Clock Gating** κι επιλέγοντας **βελτιστοποίηση ως προς την δυναμική ισχύ**. Για το λόγο αυτό τροποποιούμε το *.tcl αρχείο όπως ακολούθως :

#start of file

...

read_qrc ./qrc/qx/gpdk045.tch

#dynamic power optimization

set_db design_power_effort high

set_db opt_leakage_to_dynamic_ratio 0.0

#read design

read_hdl picorv32.v

#insert clock gating

set_db lp_insert_clock_gating true

#elaborate design

elaborate "picorv32"

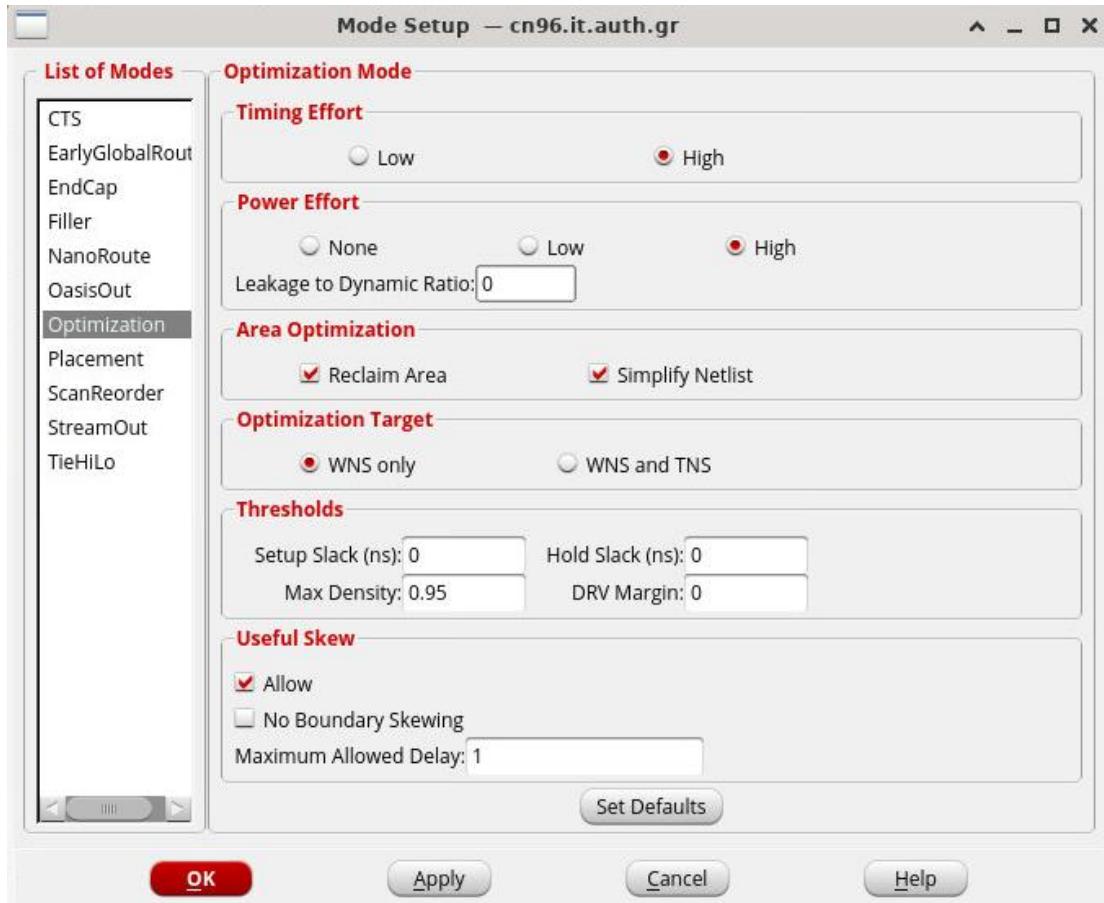
...

#end of file

Δεν τροποποιήσαμε την περίοδο ρολογιού ούτε και το αρχείο sdc !!!

Βήματα 8~15:

Στη συνέχεια εισάγουμε το design στο Innovus κι επιλέγουμε **ποσοστό χρήσης πυρίνα 70%**. Πριν την τοποθέτηση επιλέγουμε **high power effort** και **leakage to dynamic ratio 0.0** στην καρτέλα optimization για να έχουμε βελτιστοποίηση της δυναμικής ισχύος.



Μετά την τοποθέτηση εκτελούμε **optDesign -preCTS !!!**

Εκτελούμε τα υπόλοιπα βήματα όπως στην 1^η άσκηση (ίδιοι Clock Tree περιορισμοί) και μετά το *NanoRoute* εξάγουμε αναφορές ως συνήθως :

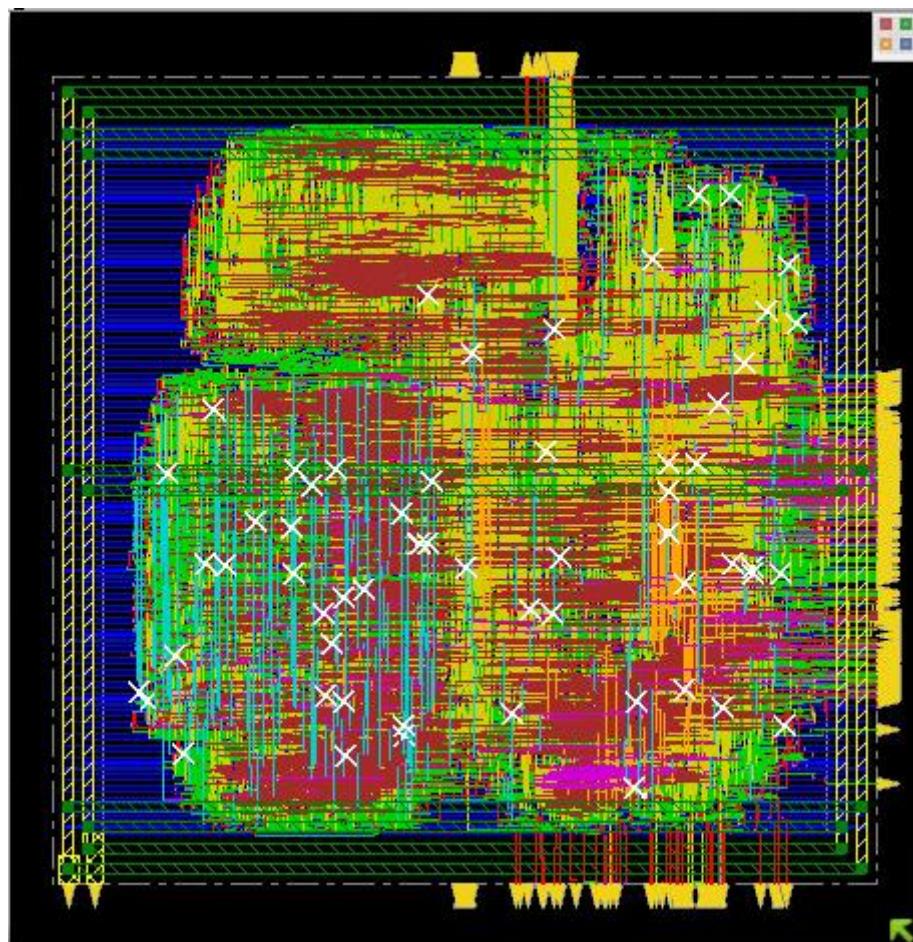
Άσκηση(#)	Internal Power (mW)	Leakage Power (mW)	Switching Power (mW)	Total Power (mW)	Area (μm ²)	Slack (ns)
1 ^η	1.691	0.0008784	2.451	4.143	35165.124	0.077
10 ^η	0.8774	0.0006871	2.189	3.067	31337.802	0.030
Παρατήρηση	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε	Μειώθηκε

Συνοψίζοντας, πετυχαίνουμε δραστική μείωση κάθε τύπου ισχύος και επιφάνειας και το slack παραμένει θετικό !!!

Αν θεωρήσουμε πως η συνολική επιφάνεια ταυτίζεται με την επιφάνεια των κελιών και την μετατρέψουμε σε mm^2 , τότε έχουμε :

$$FOM = \frac{1}{5ns \cdot 3.067mW \cdot 0.031337802mm^2} = 2.0809$$

Τέλος, το design μοιάζει κάπως έτσι :



Ευχαριστίες :

Θα θέλαμε να ευχαριστήσουμε ιδιαιτέρως τον κο Αριστοτέλη Τσεκούρα για το χρόνο και τον κόπο που διέθεσε για να λύσει τις απορίες μας κατά τη διάρκεια της υλοποίησης της εργασίας !!!

ΤΕΛΟΣ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΝΑΦΟΡΑΣ