ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΕΠΑΛΗΘΕΥΣΗ ΨΗΦΙΑΚΟΥ ΚΥΚΛΩΜΑΤΟΣ

Αλέξανδρος Σ. Δημολίτσας

Επιβλέπων: Φώτιος Βαρτζιώτης

Επίκουρος Καθηγητής

Άρτα, Φεβρουάριος, 2021

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΕΠΑΛΗΘΕΥΣΗ ΨΗΦΙΑΚΟΥ ΚΥΚΛΩΜΑΤΟΣ

Αλέξανδρος Σ. Δημολίτσας

Επιβλέπων: Φώτιος Βαρτζιώτης

Επίκουρος Καθηγητής

Άρτα, Φεβρουάριος, 2021

DESIGN AND VERIFICATION OF A DIGITAL CIRCUIT

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Τόπος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής
   Όνομα Επίθετο,
   τίτλος, βαθμίδα

2. Μέλος επιτροπής
   Όνομα Επίθετο,
   τίτλος, βαθμίδα

3. Μέλος επιτροπής
   Όνομα Επίθετο,
   τίτλος, βαθμίδα

Ο/Η Προϊστάμενος/η του Τμήματος

Όνομα Επίθετο,

τίτλος, βαθμίδα

Υπογραφή

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Αλέξανδρος Σ. Δημολίτσας

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία με θέμα «Σχεδίαση και Επαλήθευση Ψηφιακού Κυκλώματος» πραγματοποιήθηκε στο πλαίσιο της πτυχιακής εργασίας του τμήματος Πληροφορικής & Τηλεπικοινωνιών του Πανεπιστημίου Ιωαννίνων το έτος 2021.

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κύριο Βαρτζιώτη Φώτη, για την υπομονή και τις πολύτιμες συμβουλές καθώς επίσης και για τη βοήθεια του καθ' όλη τη διάρκεια της πτυχιακής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου για την αμέριστη κατανόηση, συμπαράσταση και υποστήριξη που έδειξε όλα αυτά τα χρόνια που με βοηθούσαν κατά τη διάρκεια της φοιτητικής μου ζωής.

Τέλος, θα ήθελα να ευχαριστήσω ξεχωριστά την κοπέλα μου Δήμητρα που ήταν το στήριγμα μου όλη αυτή την περίοδο και συνέβαλε τα μέγιστα ώστε να τα καταφέρω.

Αλέξανδρος Σ. Δημολίτσας

Άρτα 2021

ΠΕΡΙΛΗΨΗ

Όπως γίνεται εύκολα αντιληπτό από την πρώτη της κιόλας εμφάνιση μέχρι και σήμερα η τεχνολογία αποτελεί ένα σημαντικό κομμάτι της ζωής του ανθρώπου και τα τελευταία χρόνια της καθημερινότητας του. Στην σημερινή εποχή αμέτρητος αριθμός συσκευών χρησιμοποιεί την τεχνολογία των ψηφιακών κυκλωμάτων αρχίζοντας από ένα τηλεχειριστήριο και έναν υπολογιστή μέχρι και τεράστιες μονάδες παραγωγής.

Σκοπός της παρούσας πτυχιακής εργασίας είναι η σχεδίαση και επαλήθευση ενός ψηφιακού κυκλώματος με τη χρήση ενός εξειδικευμένου λογισμικού για αυτοματοποιημένη σχεδίαση (Electronic design automation - EDA).

Πιο συγκεκριμένα, θα χρησιμοποιήσουμε την EDA σουίτα Alliance σε λειτουργικό σύστημα Linux και την διανομή Ubuntu. Θα ξεκινήσουμε με την ανάλυση και επεξήγηση όλων των εργαλείων της σουίτας και θα υλοποιήσουμε ένα παράδειγμα ενός πολυπλέκτη για την καλύτερη κατανόησή τους. Στην συνέχεια θα προχωρήσουμε στην σχεδίαση του ψηφιακού κυκλώματος ενός πολλαπλασιαστή 4-bit. Στην σχεδίαση του κυκλώματος θα περάσουμε από τα στάδια της σύνθεσης, της τοποθέτησης και δρομολόγησης, ορθότητας σχεδιασμού, τον έλεγχο κανόνων σχεδίασης. Εκτός από τα εργαλεία του Alliance θα χρησιμοποιήσουμε και τη γλώσσα περιγραφής υλικού VHDL, με την οποία θα περιγράψουμε την αρχιτεκτονική του πολυπλέκτη και του πολλαπλασιαστή 4-bit που θα υλοποιήσουμε. Επιπρόσθετα θα δημιουργήσουμε και ένα πρόγραμμα για την αυτοματοποίηση του σχεδιασμού μας για ταχύτερη εκτέλεση, ευκολότερο εντοπισμό λαθών και τυχόν μελλοντικές βελτιώσεις.

Κλείνοντας θα είμαστε σε θέση να κατανοήσουμε τα ψηφιακά κυκλώματα, πως αυτά λειτουργούν αλλά και να τα σχεδιάζουμε.

Λέξεις κλειδιά: Σχεδίαση, Επαλήθευση, Ψηφιακό Κύκλωμα, Πολλαπλασιαστής, Alliance.

ABSTRACT

As it is easily understood from its very first appearance until today, technology is an important part of human life and the last years of his daily life. Today, countless devices use digital circuit technology, from a remote control and a computer to huge production units.

The purpose of this dissertation is to design and verify a digital circuit using a specialized software for automated design (Electronic design automation - EDA).

More specifically, we will use the EDA Alliance suite on a Linux operating system and the Ubuntu distribution. We will start by analyzing and explaining all the tools in the suite and implement an example of a multiplexer to better understand them. Next we will proceed to the design of the digital circuit of a 4-bit multiplier. In the design of the circuit we will go through the stages of composition, installation and routing, design correctness, control of design rules. In addition to the Alliance tools, we will use the VHDL hardware description language, which will describe the architecture of the multiplexer and the 4-bit multiplier that we will implement. In addition we will create a program to automate our design for faster execution, easier error detection and any future improvements.

In closing, we will be able to understand digital circuits, how they work and to design them.

Key words: Design, Verification, Digital Circuit, Multiplier, Alliance

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

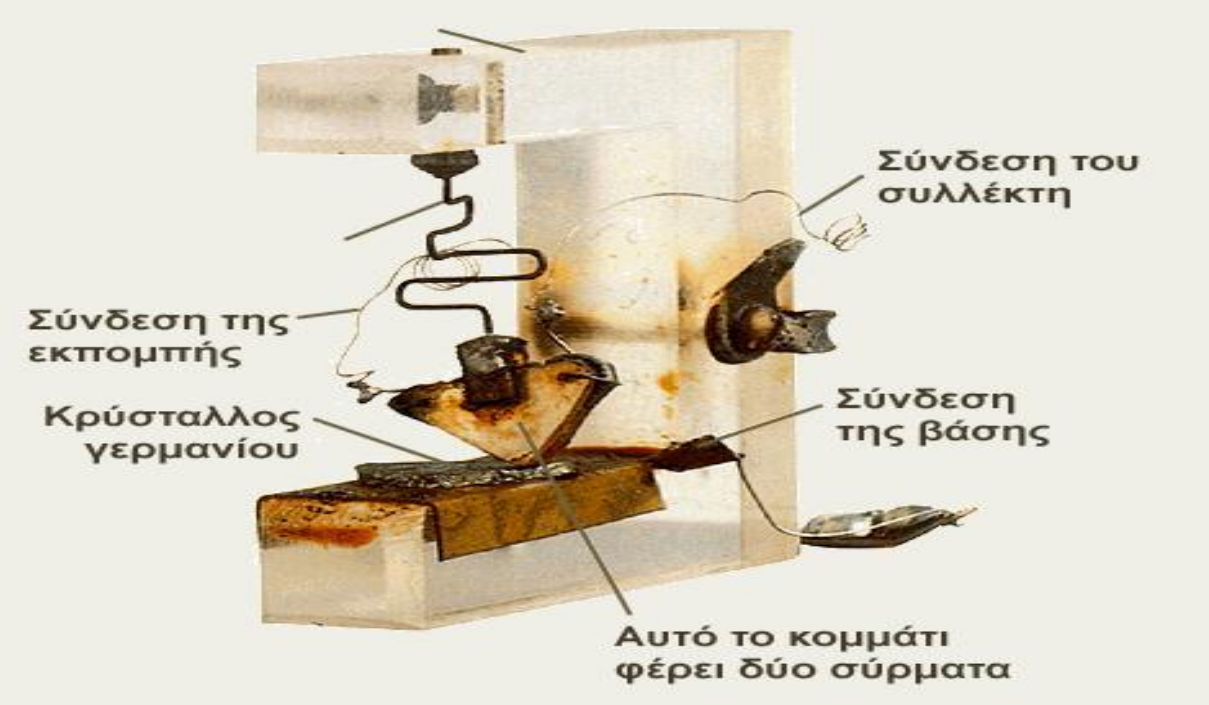
1.ΕΙΣΑΓΩΓΗ

1.1 ΟΛΟΚΛΗΡΩΜΕΝΟ ΚΥΚΛΩΜΑ

1.1.1 ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Ολοκληρωμένο κύκλωμα ονομάζεται ένα κύκλωμα λογικών πυλών το οποίο έχει δημιουργηθεί πάνω σε ένα φύλλο. Τα περισσότερα ολοκληρωμένα κυκλώματα δημιουργούνται πάνω σε φύλλα πυριτίου κυρίως. Αυτό το φύλλο πυριτίου ονομάζεται τσιπ.

Στο πρώτο μισό του 20ού αιώνα, τα ηλεκτρικά κυκλώματα χρησιμοποιούσαν μεγάλες, ακριβές, ενεργοβόρες και αναξιόπιστες λυχνίες κενού. Το 1947, στα εργαστήρια στης Bell, οι John Bardeen και Walter Brattain κατασκεύασαν το πρώτο λειτουργικό τρανζίστορ επαφής σημείου, το οποίο παρουσιάζεται στην Εικόνα 1.1.

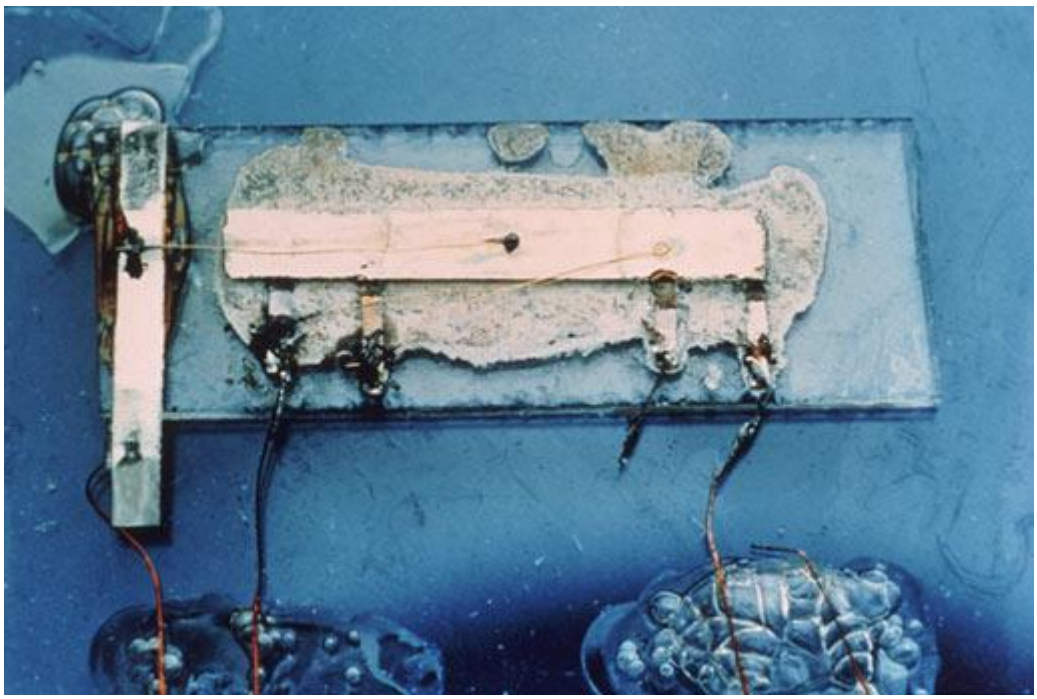


Εικόνα 1.1

Μετά από κάποια χρόνια και πάλι στα εργαστήρια της Bell αναπτύχθηκε το διπολικό τρανζίστορ επαφής, ήταν πιο αξιόπιστο και πιο αποδοτικό από τον προκάτοχό του. Έτσι τα πρώτα ολοκληρωμένα κυκλώματα χρησιμοποιήσαν αυτά τα διπολικά τρανζίστορ.

Δέκα χρόνια αργότερα στην εταιρία Texas Instruments ο Jack Kilby προχώρησε σε μια επαναστατική εφεύρεση το ολοκληρωμένο κύκλωμα. Συνειδητοποίησε ότι αν κατασκευαστούν πολλαπλά τρανζίστορ πάνω σε ένα κομμάτι πυριτίου θα επιτυγχάνονταν δυνατότητες σμίκρυνσης. Στην Εικόνα 1.2 παρουσιάζεται το πρώτο ολοκληρωμένο

κύκλωμα. Η συγκεκριμένη εφεύρεση συγκριτικά με τις προηγούμενες εφευρέσεις ήταν ισχυρότερη, αισθητά πιο οικονομική ως προς την κατασκευή και αρκετά ταχύτερη.



Εικόνα 1.2

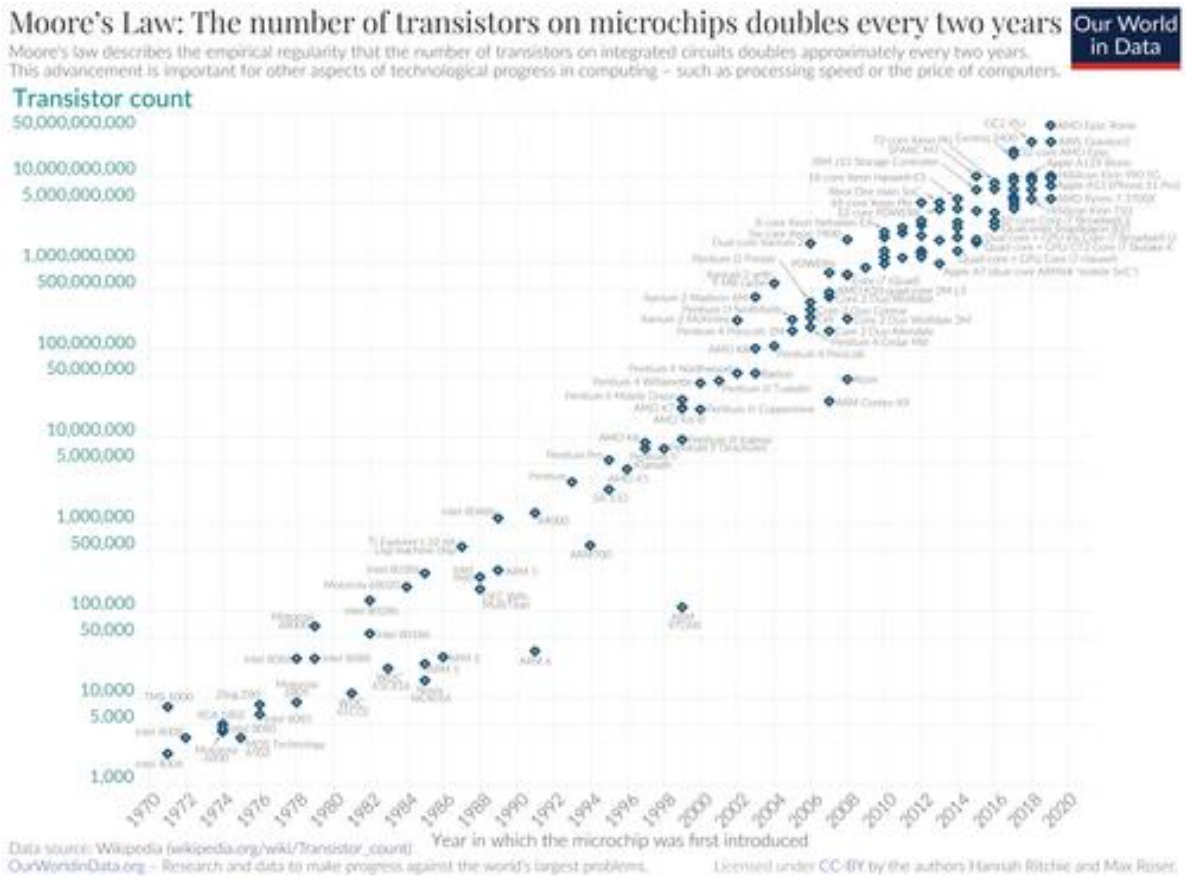
Λίγα χρόνια αργότερα κατά την δεκαετία του ’60, άρχισαν να δημιουργούνται τα τρανζίστορ επίδρασης πεδίου μετάλλου-οξειδίου-ημιαγωγού(Metal Oxide Semiconductor Field Effect Transistor, MOSFET). Το βασικό πλεονέκτημα αυτών των τρανζίστορ είναι ότι λειτουργούν με σχεδόν μηδενικό ρεύμα ελέγχου όταν δεν είναι λειτουργικά. Το MOSFET τρανζίστορ χωρίζεται σε δύο επιμέρους τρανζίστορ το nMOS και το pMOS,όπου το nMOS χρησιμοποιεί πυρίτιο τύπου -n ενώ το pMOS τύπου -p.

Μετά από 3 χρόνια ο Frank Wanlass περιέγραψε τις πρώτες λογικές πύλες κάνοντας χρήση MOSFET. Χρησιμοποίησε pMOS και nMOS τρανζίστορ στις πύλες και έτσι ονομάστηκαν συμπληρωματικοί ημιαγωγοί μετάλλου-οξειδίου(Complementary Metal Oxide Semiconductor, CMOS). Τα κυκλώματα αυτά αν και χρησιμοποιούσαν διακριτά τρανζίστορ είχαν κατά έξι τάξεις μεγέθους μικρότερη κατανάλωση ισχύος από τα αντίστοιχα διπολικά.

Η Intel ήταν πρωτοπόρος στην τεχνολογία nMOS κατασκευάζοντας την στατική μνήμη τυχαίας προσπέλασης SRAM με χωρητικότητα 256 bit. Παρόλο που η τεχνολογία nMOS ήταν πιο φθηνή από αυτή του CMOS, οι λογικές πύλες του nMOS ακόμα και αν ήταν σε αδράνεια κατανάλωναν περισσότερη ισχύ, αυτός ήταν και ένας κυριότερους λόγους που χρησιμοποιήθηκε κατά κόρων η τεχνολογία CMOS σχεδόν σε κάθε εφαρμογή ψηφιακής λογικής.

Το 1965 ο Gordon Moore προέβη στην παρατήρηση ότι αν ο αριθμός των τρανζίστορ που μπορούν να κατασκευαστούν σε ένα ολοκληρωμένο κύκλωμα παρασταθούν γραφικά προκύπτει μια ευθεία γραμμή σε ημιαλγοριθμική κλίμακα και διαπίστωσε ότι ο αριθμός των τρανζίστορ που θα περιέχονται σε ένα ολοκληρωμένο κύκλωμα θα διπλασιάζονται περίπου κάθε 18 μήνες. Η συγκεκριμένη παρατήρηση έμεινε στην ιστορία ως “Νόμος του Moore”.

Ο συγκεκριμένος νόμος βρίσκει βάση και όπως φαίνεται και στην Εικόνα 1.3 μέχρι και το 2012, από και μετά όπως διακρίνεται υπάρχει μια επιβράδυνση με την εισαγωγή ημιαγωγών 22 νανομέτρων. Παρόλο την συγκεκριμένη επιβράδυνση οι επιστήμονες συνέχισαν την εξέλιξη και το 2018 κυκλοφόρησαν ολοκληρωμένα κυκλώματα 7 μανομέτρων, και φτάνοντας στο σήμερα, το 2020, έχουμε ολοκληρωμένα κυκλώματα των 5 νανομέτρων. Ωστόσο η εξέλιξη δεν φαίνεται να σταματάει εκεί καθώς οι επιστήμονες υποστηρίζουν ότι μέχρι και το 2022 θα είναι σε θέση να δημιουργήσουν ολοκληρωμένα κυκλώματα των 2 νανομέτρων. [1][4][5][21]



Εικόνα 1.3

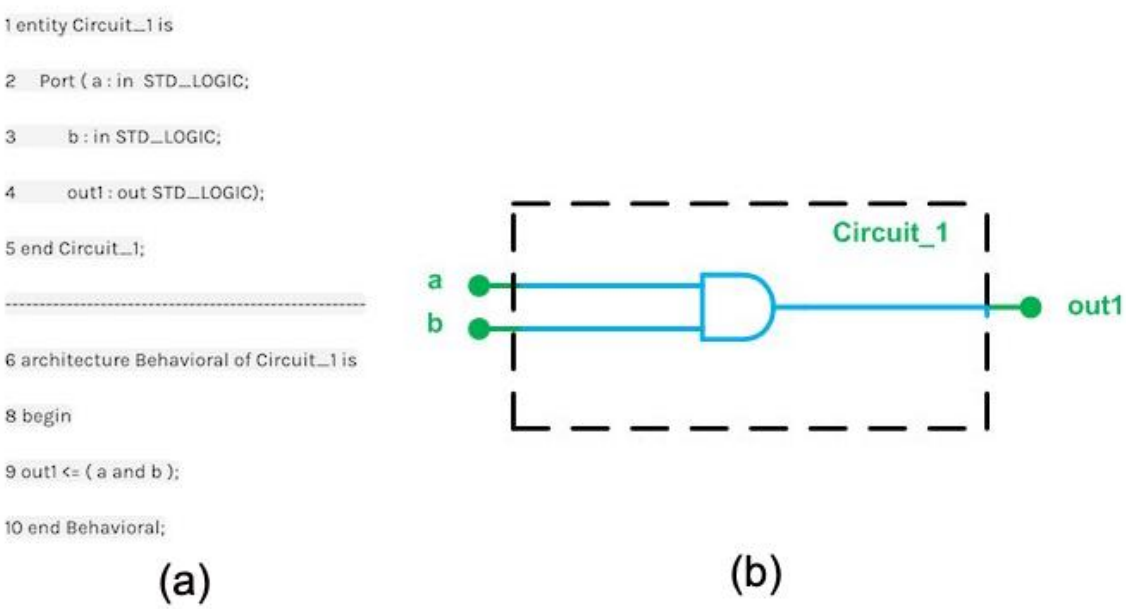
1.2 ΣΧΕΔΙΑΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

Με την πάροδο των χρόνων τα σχέδια ολοκληρωμένων κυκλωμάτων, όπως κυκλώματα VLSI(Very large-scale integration) , έχουν γίνει αρκετά πολύπλοκα. Αυτό έχει οδηγήσει τους σχεδιαστές τέτοιων κυκλωμάτων να χρησιμοποιούν μεθόδους CAD για να τα απλοποιήσουν. Ένα σύστημα CAD(Computer-aided design) επαληθεύει την σωστή σχεδίαση ενός κυκλώματος, προσομοιώνει την συμπεριφορά ενός ολοκληρωμένου κυκλώματος ενώ ταυτόχρονα ο χρήστης μπορεί να προχωράει με τον σχεδιασμό του. Ένα ακόμα χαρακτηριστικό αυτών των συστημάτων είναι ότι μπορούν και αποθηκεύουν μεγάλα διαγράμματα.

Η φυσική σχεδίαση ενός ολοκληρωμένου κυκλώματος υποστηρίζεται από τέσσερις τύπους εργαλείων CAD. Η πρώτη είναι η γεωμετρική προσέγγιση, όπου για την κατασκευή του ακριβούς σχήματος μιας δομής(structure) σε μια μάσκα ολοκληρωμένου κυκλώματος, χρησιμοποιείται ένα γραφικό σύστημα γενικής χρήσης. Το δεύτερο είναι η συμβολική προσέγγιση, όπου το σύστημα μετατρέπει τις πληροφορίες από την συμβολική προσέγγιση σε ακριβείς γεωμετρίες. Σαν τρίτος τύπος εργαλείων θεωρείται η προσέγγιση που βασίζεται σε κελιά(cells). Τα κελιά λειτουργίας και τα δεδομένα απόδοσης μπορούν χαρακτηριστούν πλήρως και οι προδιαγραφές τους αποθηκεύονται σε βιβλιοθήκες κελιών(cell library). Στην τέταρτη προσέγγιση, τα κελιά τοποθετούνται σε μια διάταξη(layout) ολοκληρωμένου κυκλώματος η οποία χρησιμοποιεί μια διαδικαστική γλώσσα για την περιγραφή της τοποθέτησης (placement) και των διασυνδέσεων τους.[1][6][8]

1.2.1 ΣΥΝΘΕΣΗ ΚΑΙ ΕΠΑΛΗΘΕΥΣΗ: ΠΕΡΙΓΡΑΦΗ ΥΛΙΚΟΥ ΓΛΩΣΣΑ ΚΑΙ ΛΕΙΤΟΥΡΓΙΚΗ ΕΠΑΛΗΘΕΥΣΗ

Τα μπλοκ με περιγραφές συμπεριφοράς που δημιουργήθηκαν στις πρώτες φάσεις του ψηφιακού σχεδιασμού πρέπει να μεταφραστούν σε γλώσσα περιγραφής υλικού HDL, όπως VHDL ή Verilog. Η φάση αυτή της μετάφρασης είναι γνωστή ως RTL(Register Transfer Level) και περιλαμβάνει επαλήθευση για να εξασφαλιστεί ότι η υλοποίηση της λογικής πληροί τις προϋποθέσεις σε high-level. Στην Εικόνα 1.4 που ακολουθεί στο σχήμα (a) παρουσιάζεται ένα παράδειγμα κώδικα HDL και στο σχήμα (b) το κύκλωμα που αυτός ο κώδικας περιγράφει.



Εικόνα 1.4

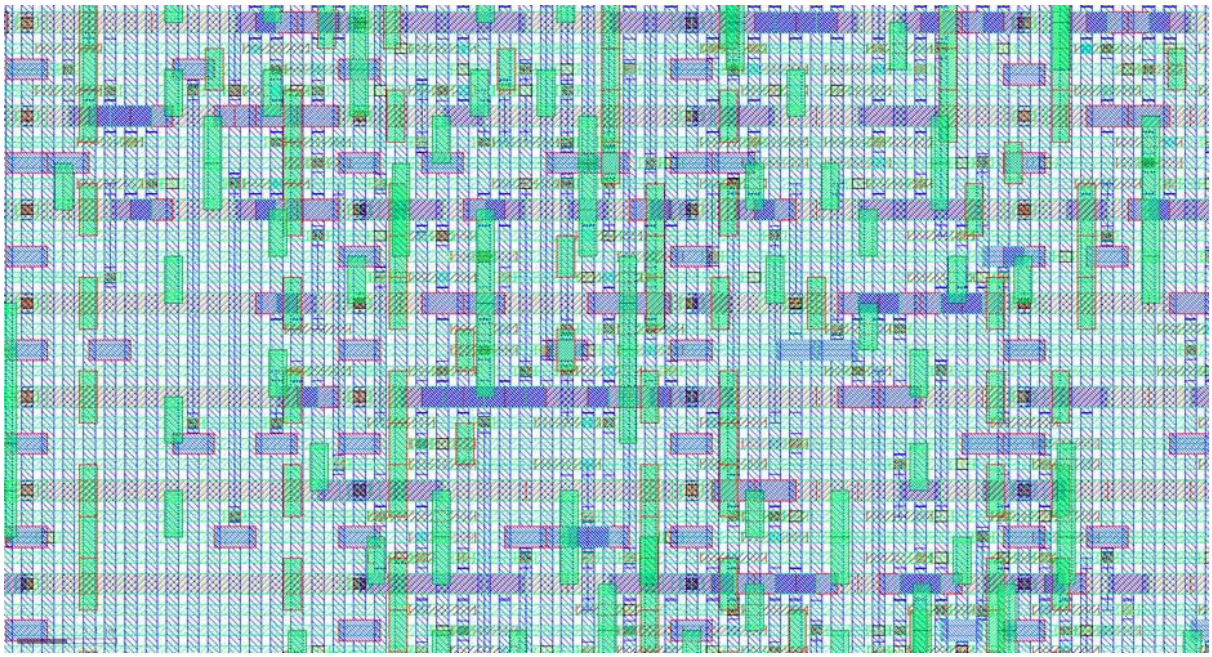
Στην συνέχεια η περιγραφή του υλικού μετατρέπεται σε netlist επιπέδου πύλης. Σε αυτό το στάδιο με την βοήθεια αλγορίθμων βελτιστοποίησης επιτυγχάνουν σημαντικές βελτιώσεις σε ισχύ, ταχύτητα, αξιοπιστία. [7][9]

1.2.2 ΦΥΣΙΚΗ ΔΙΑΤΑΞΗ (LAYOUT) ΟΛΟΚΛΗΡΩΜΕΝΟΥ ΚΥΚΛΩΜΑΤΟΣ

Αφού έχει ολοκληρωθεί και η σύνθεση και η επαλήθευση, η netlist μετατρέπεται σε φυσικό layout. Το φυσικό αυτό layout είναι μια γεωμετρική αναπαράσταση των επιπέδων και της φυσικής δομής του ολοκληρωμένου κυκλώματος.

1.2.3 ΕΠΑΛΗΘΕΥΣΗ ΚΑΙ ΠΡΟΣΟΜΟΙΩΣΗ

Με την ολοκλήρωση του πιο πάνω βήματος πραγματοποιείται επαλήθευση και προσομοίωση λαμβάνοντας υπόψιν την τοποθέτηση (placement) και τα φυσικά χαρακτηριστικά του φυσικού layout. Αφού ολοκληρωθεί επιτυχώς και αυτό το βήμα το αποτέλεσμα θα είναι ένα αρχείο εξόδου το οποίο θα είναι ένα έτοιμο ολοκληρωμένο σύστημα. Στη Εικόνα 1.5 που ακολουθεί παρουσιάζεται το layout ενός ολοκληρωμένου κυκλώματος μετά την τοποθέτηση (placement) και δρομολόγηση(route). Το συγκεκριμένο ολοκληρωμένο σύστημα έχει υλοποιηθεί με το Cadence Design System.[9]



Εικόνα 1.5

1.3 ΑΝΤΙΚΕΙΜΕΝΟ ΠΤΥΧΙΑΚΗΣ

Σκοπός της παρούσας πτυχιακής είναι η σχεδίαση και η δημιουργία ενός πολλαπλασιαστή με την χρήση της γλώσσας περιγραφής υλικού HDL και η αξιολόγηση της σωστής λειτουργίας με την βοήθεια των εργαλείων που παρέχονται από το Alliance CAD tool. Το Alliance μας παρέχει ένα σύνολο εργαλείων που μας επιτρέπει να σχεδιάσουμε και να επαληθεύσουμε το κύκλωμα που επιθυμούμε με βάσης της δικές μας προδιαγραφές. Για την υλοποίηση του κυκλώματος φορτώθηκε το περιβάλλον Alliance σε λειτουργικό σύστημα Linux, τα εργαλεία του Alliance χρησιμοποιήθηκαν ως εργαλεία γραμμής εντολών. Χρησιμοποιήθηκε η γλώσσα περιγραφής υλικού VHDL για να περιγραφεί ο πολυπλέκτης που χρησιμοποιήθηκε ως παράδειγμα για την κατανόηση των εργαλείων καθώς επίσης και για την υλοποίηση του πολλαπλασιαστή που είναι και το αντικείμενο της πτυχιακής. Χρησιμοποιήθηκε μια τυπική βιβλιοθήκη κυττάρων του Alliance καθώς και αρκετά κελιά ειδικού σκοπού για την μνήμη και λογική διαδρομή δεδομένων. Με την βοήθεια των εργαλείων εκτός από τον σχεδιασμό επιτεύχθηκαν ελαχιστοποιήσεις στις καθυστερήσεις και της περιοχής διασύνδεσης. Τέλος αφού μελετήθηκαν όλα τα εργαλεία της σουίτας δημιουργήθηκε ένα πρόγραμμα με σκοπό να αυτοματοποιηθεί η διαδικασία σχεδίασης και να είναι ευκολότερη η τυχόν τροποποίηση στο σχέδιο μας.

1.3.1 ΜΕΘΟΔΟΛΟΓΙΑ

Η μεθοδολογία της πτυχιακής μπορεί να συνοψιστεί ως εξής:

- Αναπτύχθηκε ένα παράδειγμα πολυπλέκτη για την κατανόηση όλων των εργαλείων του Alliance που θα χρησιμοποιήσουμε. Γίνεται εκτενής περιγραφή του κάθε εργαλείου καθώς και των αποτελεσμάτων του καθενός από αυτά πάνω στο παράδειγμα του πολυπλέκτη. Τα εργαλεία αυτά χωρίζονται σε κατηγορίες όπως εργαλεία για RTL σύνθεση, τοποθέτηση και δρομολόγηση, εξαγωγή netlist και παρασιτικών πληροφοριών, σύγκριση netlist, έλεγχο κανόνων σχεδιασμού, μετατροπή συμβολικού layout σε πραγματικό καθώς και εργαλεία αναπαράστασης αποτελεσμάτων.
- Υλοποίηση πολλαπλασιαστή με χρήση εργαλείων του Alliance. Χρησιμοποιήθηκαν γλώσσες περιγραφής υλικού που με την χρήση των κατάλληλων εργαλείων μας δίνουν την τελική μορφή του πολλαπλασιαστή.
- Η κατασκευή του πολλαπλασιαστή χωρίζεται σε τρία μπλοκ : την γεννήτρια μερικών γινομένων, το δέντρο αθροιστών για να προσθέσει τα μερικά γινόμενα και τοn αθροιστή εξόδου.

1. Το πρώτο μπλοκ είναι υπεύθυνο για την παραγωγή μερικών γινομένων. Το δεύτερο μπλοκ για το δέντρο αθροιστών και χρησιμοποιεί τον αλγόριθμο Wallace. Ένα δέντρο Wallace είναι μία αποδοτική υλοποίηση ενός ψηφιακού κυκλώματος που πολλαπλασιάζει δύο ακέραιους αριθμούς.
2. Με την ολοκλήρωση και των τριών αυτών μπλοκ δημιουργούμε τον πολλαπλασιαστή χρησιμοποιώντας δομική περιγραφή αρχικά και με την χρήση των απαραίτητων εργαλείων φτάνουμε στην τελική του μορφή που είναι ένας τεσσάρων bit πολλαπλασιαστής.
3. Δημιουργούμε ένα πρόγραμμα με όλες τι απαραίτητες μεταβλητές περιβάλλοντος και κάθε εργαλείο που χρησιμοποιήθηκε στην σύνθεση του πολλαπλασιαστή με σκοπό να είναι πιο εύκολη η τροποποίηση του σχεδίου και ταχύτερη η υλοποίησή του κάθε φορά.

1.4 ΟΡΓΑΝΩΣΗ ΚΕΙΜΕΝΟΥ

Στο δεύτερο κεφάλαιο παρουσιάζονται συνοπτικά παρόμοιες σουίτες με την σουίτα Alliance πάνω στην οποία βασίστηκε η υλοποίηση του κυκλώματός μας καθώς επίσης γίνεται και μία πρώτη παρουσίαση του Alliance. Στο κεφάλαιο 3 αναλύεται λεπτομερώς το Alliance. Πιο συγκεκριμένα γίνεται ανάλυση των εργαλείων ως προς την λειτουργία τους, την χρήση και τις επιλογές που θα πρέπει να εφαρμόζονται ανάλογα με τις εκάστοτε απαιτήσεις, μέχρι και την παρουσίαση των αποτελεσμάτων τους. Επιπρόσθετα στο κεφάλαιο αυτό αναλύεται σε κάθε στάδιο του ο πολλαπλασιαστής. Περιλαμβάνει μια θεωρητική ανάλυση για κάθε ένα από τα τρία μπλοκ από τα οποία απαρτίζεται ο πολλαπλασιαστής και ιδιαίτερα στο μπλοκ του δέντρου αθροιστών και του αλγορίθμου Wallace. Παρουσιάζονται βήμα-βήμα όλοι οι κώδικες, όλα τα εργαλεία που απαιτούνται και όλα τα αποτελέσματα από την χρήση αυτών μέχρι και το τελικό πραγματικό layout του πολλαπλασιαστή. Στο κεφάλαιο 4 γίνεται παρουσίαση του κυκλώματος και πραγματοποιούνται οι απαραίτητες επεξηγήσεις και αναλύσεις των αποτελεσμάτων. Στο έβδομο και τελευταίο κεφάλαιο παραθέτονται τα συμπεράσματα που προκύπτουν από την υλοποίηση της εργασίας.

2.ΤΟ ΠΡΟΓΡΑΜΜΑ ΣΧΕΔΙΑΣΗΣ VLSI- ALLIANCE VSLI/CAD SYSTEM ΚΑΙ ΠΡΟΪΟΝΤΑ ΑΝΤΑΓΩΝΙΣΜΟΥ

Ένα καλό VLSI CAD SYSTEM πρέπει να υποστηρίζει τα εξής στοιχεία: λογικό σχεδιασμό, σχεδιασμό κυκλωμάτων, δημιουργία διάταξης(layout) και έλεγχο σχεδιασμού. Στην σημερινή εποχή τα περισσότερα συστήματα αυτού του είδους βασίζονται σε πλατφόρμες Unix ή Linux. Στο παρών κεφάλαιο θα αναλυθούν συνοπτικά το πρόγραμμα Alliance πάνω στο οποίο βασίστηκε η παρούσα εργασία και επιπλέον κάποια από τα σημαντικότερα προϊόντα ανταγωνισμού.

2.1 ALLIANCE VSLI/CAD SYSTEM

Το Alliance είναι ένα πλήρες σύνολο εργαλείων CAD και βιβλιοθηκών για έρευνα και εκπαίδευση στην σχεδίαση VLSI που διανέμεται δωρεάν. Βασίζεται σε πλατφόρμες Unix ή Linux κατά συνέπεια η εγκατάστασή του μπορεί να πραγματοποιηθεί σε οποιοδήποτε λειτουργικό σύστημα Unix ή Linux. Για την υλοποίησης της συγκεκριμένης εργασίας το Alliance εγκαταστάθηκε σε λειτουργικό σύστημα Linux και πιο συγκεκριμένα στην διανομή Ubuntu 14.04.

Το Alliance CAD System ως ένα ολοκληρωμένο λογισμικό σχεδίασης περιλαμβάνει VHDL μεταγλωττιστή και προσομοιωτή, εργαλεία σύνθεσης λογικής, αυτόματη τοποθέτηση σχεδιασμού, και δρομολόγηση, έλεγχο κανόνων VLSI, σύγκρισης και μετατροπής συμβολικών layout σε πραγματικά. Το σύνολο αυτών των εργαλείων μας επιτρέπουν να σχεδιάσουμε ένα κύκλωμα με τις δικές μας προδιαγραφές.

Το Alliance περιέχει μια συμβολική βιβλιοθήκη κελιών που καθιστά ανεξάρτητο τον εκάστοτε σχεδιασμό κυκλωμάτων από την τεχνολογία που χρησιμοποιείται στον στάδιο της κατασκευής τους. Οι βιβλιοθήκες κελιών περιλαμβάνουν μια τυπική βιβλιοθήκη κελιών και διάφορα κελιά ειδικού σκοπού για την μνήμη και την λογική της διαδρομής δεδομένων.

Πολλά από τα εργαλεία που παρέχει το Alliance μπορούν να χρησιμοποιηθούν και ως εργαλεία γραμμής εντολών ανεξάρτητα από το σχεδιασμό που τρέχει ο χρήστης. Όπως θα αναλύσουμε και παρακάτω εάν παρέχουμε ένα κατάλληλο αρχείο τεχνολογίας (RDS file) ο σχεδιασμός που έχουμε υλοποιήσει με το Alliance μπορεί να μετατραπεί σε μορφή CIF ή GDSII για να τυπωθεί πάνω σε πυρίτιο.[8]

2.2 ΠΡΟΪΟΝΤΑ ΑΝΤΑΓΩΝΙΣΜΟΥ

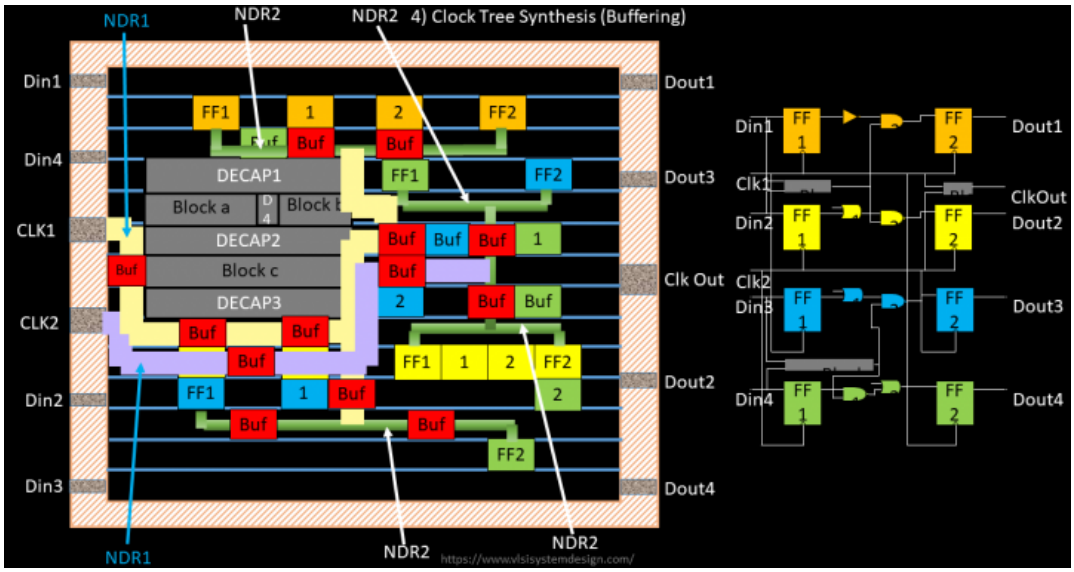
Όπως αναφέρθηκε πιο πάνω υπάρχει ποικιλία λογισμικών για σχεδίαση ολοκληρωμένων κυκλωμάτων VSLI για διάφορα λειτουργικά συστήματα τα οποία διατίθενται δωρεάν ή κατόπιν πληρωμής. Καθώς το λογισμικό που υλοποιήσαμε την συγκεκριμένη εργασία διανέμεται δωρεάν έτσι και τα άλλα λογισμικά της ίδιας οικογένειας που επιλέχθηκαν να αναλυθούν διατίθενται και αυτά δωρεάν. Λόγο και της μεγάλης ποικιλίας σε αυτή την κατηγορία επιλέχθηκαν δύο από τα ίσως ποιο γνωστά λογισμικά σχεδίασης κυκλωμάτων το Qflow και το Icarus Verilog.

2.2.1 Qflow

Το Qflow δημιουργεί μια ροή ψηφιακής σύνθεσης κάνοντας χρήση εργαλείων ανοιχτού κώδικα EDA . Όπως το Alliance έτσι και το Qflow είναι ένα εργαλείο γραμμής εντολών με την χρήση τερματικού. Το συγκεκριμένο εργαλείο τρέχει σε λογισμικό GUI πράγμα που το κάνει δύσκολο στην χρήση καθώς οι χρήστες δεν είναι τόσο εξοικειωμένοι με το συγκεκριμένο λογισμικό και έτσι τους αποτρέπει από την χρήση του.

Το Qflow διαθέτει πλήρη και ολοκληρωμένη γκάμα εργαλείων για την σύνθεση ψηφιακών κυκλωμάτων με αρχή από ένα κώδικα σε μορφή Verilog και κατάληξη σε ένα πραγματικό layout. Είναι ένας συνδυασμός πολλών εξαρτήσεων για την σύνθεση, τοποθέτηση, διομολόγηση και προβολή layout. Ο βασικός σκοπός για τα εργαλεία σύνθεσης είναι ότι θα πρέπει να έχουν την δυνατότητα να χαρτογραφούν ένα λογικό κύκλωμα σε μια τυπική βιβλιοθήκη cell. [10][11]

Στην Εικόνα 2.1 που ακολουθεί παρουσιάζεται ένα στιγμιότυπο από το layout ενός σχεδιασμού.



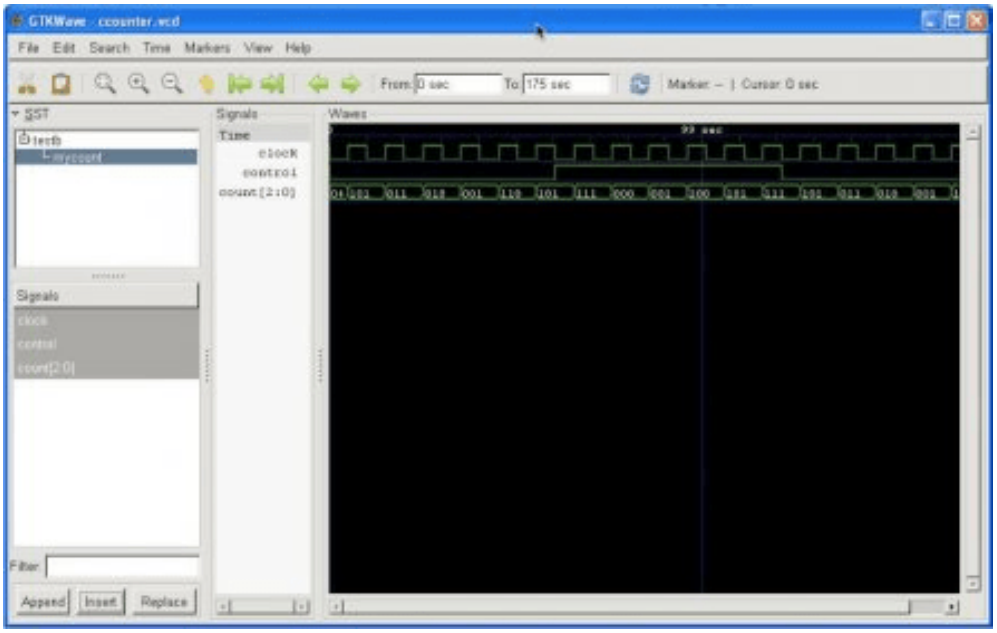
Εικόνα 2.1

2.2.2 ICARUS VERILOG

Το Icarus Verilog είναι ένα εργαλείο προσομοίωσης και σύνθεσης Verilog και διανέμεται δωρεάν. Είναι διαθέσιμο για διάφορα λειτουργικά συστήματα μερικά από τα οποία είναι Linux, Windows,Mac OS X., OpenSolaris. Το συγκεκριμένο εργαλείο διαθέτει διάφορες εκδόσεις αλλά πιο συγκεκριμένα από την έκδοση 0.9 λειτουργεί ως μεταγλωττιστής κώδικα που γράφτηκε σε Verilog για ένα σχεδιασμό που επιθυμεί ο χρήστης. Για σύνθεση ο μεταγλωττιστής του εργαλείου δημιουργεί netlist στην μορφή που επιθυμεί ο χρήστης. Ο μεταγλωττιστής έχει σαν σκοπό να αναλύει και να επεξεργάζεται περιγραφές σχεδίασης γραμμένες στο πρότυπο IEEE Standard IEEE Std 1364-2005.

Ο συγκεκριμένος μεταγλωττιστής γράφτηκε από τον Stephen Williams. Είναι ένα εργαλείο που βρίσκεται ακόμα σε εξέλιξη .[12][13]

Στην Εικόνα 2.2 που ακολουθεί φαίνεται το περιβάλλον του Icarus Verilog καθώς και το πως εμφανίζει κάποια από τα αποτελέσματα στην έξοδο.



Εικόνα 2.2

3. ΠΑΡΟΥΣΙΑΣΗ ΚΑΙ ΑΝΑΛΥΣΗ ΕΡΓΑΛΕΙΩΝ

Στο παρόν κεφάλαιο θα πραγματοποιηθεί λεπτομερή παρουσίαση των εργαλείων που παρέχει το Alliance. Το σύνολο των εργαλείων αυτών μας επιτρέπει να σχεδιάσουμε και να δοκιμάσουμε ένα κύκλωμα από τα πρώτα του στάδιο δηλαδή τις προδιαγραφές που εμείς έχουμε ορίσει μέχρι και την τελική του μορφή το layout. Βέβαια το Alliance μας παρέχει την δυνατότητα εκτός από την ολοκληρωμένη μορφή ενός σχεδιασμού να το δοκιμάσουμε και σε πολλές ακόμα ενδιάμεσες μορφές. Για να γίνει ακόμα πιο κατανοητή η λειτουργία και η χρήση του κάθε εργαλείου θα χρησιμοποιήσουμε ως παράδειγμα έναν πολυπλέκτη 2 εισόδων.

3.1 ASIMUT

Το ASIMUT(A SIMUlation Tool) είναι εργαλείο γραμμής εντολών το οποίο μπορεί να χρησιμοποιηθεί για να ελεγχθεί η περιγραφή ενός κυκλώματος σε behavioral ή structural μορφή της γλώσσας VHDL. Όπως δηλώνει και το όνομά του μπορεί να χρησιμοποιηθεί για προσομοίωση και παραγωγή εξόδων από ένα κύκλωμα για ένα δεδομένο σύνολο εισόδων. Τρέχουμε το ASIMUT στην γραμμή εντολών ως εξής:

asimut [options] [target_file] [input_pattern_file] [output_pattern_file]

- **Options**: Σε αυτό το πεδίο βάζουμε διάφορες επιλογές με τις οποίες το ASIMUT λειτουργεί ανάλογα.
- **Target_file**: Εδώ μπαίνει το αρχείο που θέλουμε να θέλουμε να δημιουργήσουμε
- **Input_pattern_file**: Είναι ένα αρχείο εισόδου pattern. Αυτό το αρχείο που περιέχει όλους τους πιθανούς συνδυασμούς των εισαγόμενων τιμών.
- **Output_pattern_file**: Είναι ένα αρχείο εισόδου pattern. Αυτό το αρχείο που περιέχει όλους τους πιθανούς συνδυασμούς των εισαγόμενων τιμών αλλά αυτή την φορά έχει και τις εξόδους των εισαγόμενων τιμών.

3.2 B2F

Το εργαλείο B2F(Behavior to Finite state machine format abstractor) είναι ένας μεταφραστής από περιγραφή συμπεριφοράς σε μορφή FSM(Finite State Machine). Ως είσοδο χρησιμοποιεί μια περιγραφή RTL(Register Transfer Level) της γλώσσας VHDL σε μορφή .**vbe**. Το B2F χρησιμοποιεί ένα συμβολικό αλγόριθμο προσομοίωσης για να δημιουργεί (build) ένα γράφημα ισοδύναμο με την μηχανή πεπερασμένων καταστάσεων που δίνετε ως έξοδος.

Τρέχουμε το B2F στην γραμμή εντολών ως εξής:

b2f [options] input_file output_flle

- **Options**: Σε αυτό το πεδίο βάζουμε διάφορες επιλογές με τις οποίες το B2F λειτουργεί ανάλογα.
- **Input_file**: Είναι το αρχείο που περιέχει την περιγραφή συμπεριφοράς(.vbe)
- **Output file**: Το αρχείο που θα μετατραπεί σε μορφή γραφήματος(.fsm)

Για να εξηγήσουμε καλύτερα πως λειτουργεί το συγκεκριμένο εργαλείο θα χρησιμοποιήσουμε την ακόλουθη περιγραφή vhdl ενός απλού controller Εικόνα 3.1.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity RWFSM is
   port(
      CLK : in  std_logic;
      START : in  std_logic;
      RW    : in  std_logic;
      NE    : in  std_logic;
      BUSY  : out std_logic);
end RWFSM;

architecture RTL of RWFSM is
   type STATE_TYPE is (IDLE, READING, WRITING, WAITING);
   signal CURRENT_STATE, NEXT_STATE: STATE_TYPE;
begin
   comb : process(CURRENT_STATE, START, RW, NE)
   begin
      BUSY  <= '0';
      case CURRENT_STATE is
         when IDLE =>
            if START = '0' then
               NEXT_STATE <= IDLE;
            elsif RW = '1' then
               NEXT_STATE <= READING;
            else
               NEXT_STATE <= WRITING;
            end if;
         when READING =>
            BUSY      <= '1';
            if NE = '0' then
               NEXT_STATE <= READING;
            else
               NEXT_STATE <= WAITING;
            end if;
         when WRITING =>
            BUSY      <= '1';
            if NE = '0' then
               NEXT_STATE <= WRITING;
            else
               NEXT_STATE <= WAITING;
            end if;
         when WAITING =>
            BUSY       <= '1';
            NEXT_STATE <= IDLE;
      end case;
   end process;

   seq : process
   begin
      wait until CLK'event and CLK = '1';
      CURRENT_STATE <= NEXT_STATE;
   end process;
end RTL;
```

Εικόνα 3.1

Αρχικά θα μετατρέψουμε αυτό το αρχείο σε behavioral περιγραφή (.vbe). Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το εργαλείο VASY του Alliance. Περισσότερη ανάλυση για την χρήση και τις ιδιότητες του εργαλείου αυτού θα ακολουθήσουν στο αντίστοιχο υποκεφάλαιο.

```
vasy -Vaop -I vhd rwfsm
```

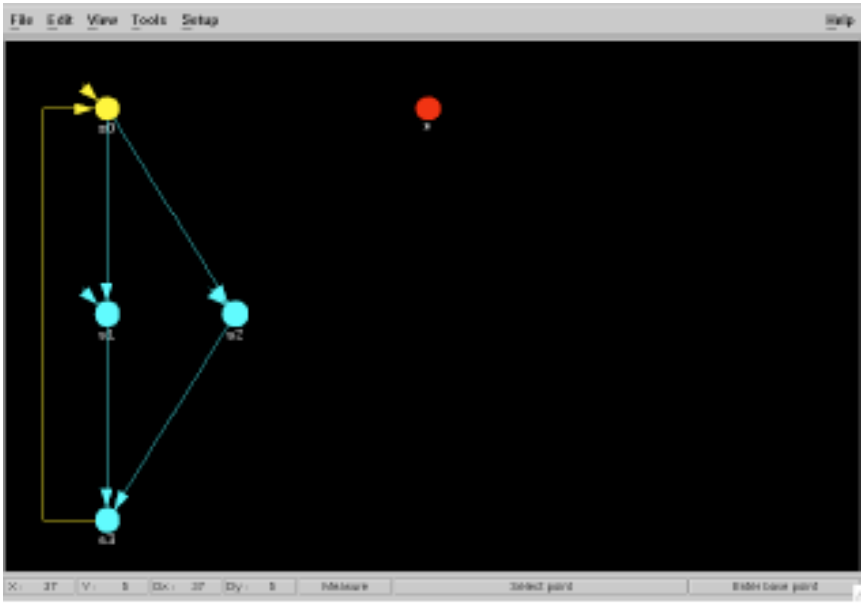
- Επιλογή **V**: Η λειτουργία Verbose είναι ενεργοποιημένη. Κάθε βήμα της ανάλυσης εμφανίζεται στην τυπική έξοδο.
- Επιλογή **a**:Οδηγεί μια ισοδύναμη περιγραφή σε μορφή **.vbe(**behavioral) ή **.vst**(structural)
- Επιλογή **o**: Επιτρέπει την αντικατάσταση υπαρχόντων αρχείων.
- Επιλογή **p**: Προσθέτει βύσματα τροφοδοσίας(power connectors) για την πηγή(vdd) και την γείωση(vss).

Έχοντας μετατρέψει το αρχείο μας σε μορφή behavioral, χρησιμοποιώντας το εργαλείο B2F το μετατρέπουμε εκ νέου σε μορφή γραφήματος.

```
b2f -V rwfsm rwgraph
```

- Επιλογή **V**: Η λειτουργία Verbose είναι ενεργοποιημένη. Κάθε βήμα της ανάλυσης εμφανίζεται στην τυπική έξοδο.

Έχοντας δημιουργήσει το αρχείο σε μορφή γραφήματος θα το απεικονίσουμε με ένα από τα εργαλεία απεικόνισης που διαθέτει το Alliance το XFSM το οποίο θα αναλυθεί στο αντίστοιχο υποκεφάλαιο. Γράφοντας την εντολή xfsm στο τερματικό και αφού την τρέξουμε εμφανίζεται ένα παράθυρο. Από το File επιλέγουμε το αρχείο με όνομα rwgraph.fsm και θα εμφανιστεί το γράφημα που έχουμε δημιουργήσει όπως φαίνεται στην Εικόνα 3.2 [14].



Εικόνα 3.2

3.3 BOOM

Το BOOM(BOOlean Minimization) στο λογισμικό σχεδίασης Alliance χρησιμοποιείται στα πρώτα στάδια της διαδικασίας σύνθεσης. Βελτιστοποιεί μια behavioral περιγραφή χρησιμοποιώντας μια αναπαράσταση της λογικής λειτουργίας του με χρήση ενός διαγράμματος RBDD(Reduced Ordered Binary Diagram) δηλαδή δυαδικό διάγραμμα μειωμένης διάταξης. Ένα μειονέκτημα του BOOM είναι ότι δίνει καλά αποτελέσματα μόνο για μικρή τυχαία λογική, συνεπώς δεν μπορεί να χρησιμοποιηθεί για βελτιστοποίηση Datapath.

Χρησιμοποιούμε το BOOM στην γραμμή εντολών με την εξής εντολή:

boom [-VTOAP] [-l num] [-d num] [-i num] [-a num] [-sjbgpwtmorn] filename [outname]

- Επιλογή **V**: Η λειτουργία Verbose είναι ενεργοποιημένη. Κάθε βήμα της ανάλυσης εμφανίζεται στην τυπική έξοδο.
- Επιλογή **T**: Η λειτουργία ανίχνευσης είναι ενεργοποιημένη. Κάποιες από τις πληροφορίες εντοπισμού σφαλμάτων εμφανίζονται στην τυπική έξοδο.
- Επιλογή **O**: Αντιστρέφει την αρχική σειρά μεταβλητών Bdd.
- Επιλογή **A**: Το BOOM εκτελεί μια τοπική βελτιστοποίηση και διατηρεί την αρχιτεκτονική της αρχικής περιγραφής αποθηκεύοντας τα περισσότερα από τα ενδιάμεσα σήματα. Η συγκεκριμένη λειτουργία είναι κατάλληλη για μεγάλα ή κανονικά κυκλώματα όπως πολλαπλασιαστές (multipliers) και αθροιστές (adders).Το BOOM ως προεπιλογή εκτελεί

καθολική βελτιστοποίηση και αφαιρεί τα περισσότερα από τα ενδιάμεσα σήματα με σκοπό οι έξοδοι να εκφράζονται σε όρους εισόδου ή εσωτερικών καταχωρητών. Επίσης αυτή η επιλογή είναι κατάλληλη για κυκλώματα όπως FSM δηλαδή κυκλώματα μηχανών πεπερασμένων καταστάσεων.

- Επιλογή **l num**: Καθορίζει το επίπεδο βελτιστοποίησης. Παίρνει τιμές [0-3] και η προεπιλεγμένη τιμή είναι 0).
- Επιλογή **d num**: Καθορίζει το ποσοστό βελτιστοποίησης καθυστέρησης. Προεπιλεγμένη τιμή είναι 0% για την καθυστέρηση και για επιφάνεια είναι 100%.
- Επιλογή **i num**: Καθορίζει τον αριθμό επαναλήψεων για τον επιλεγμένο αλγόριθμο βελτιστοποίησης. Συνιστάται η χρήση του μόνο για πιο εξειδικευμένους χρήστες.
- Επιλογή **a num**: Καθορίζει το πλάτος κατά την διάρκεια της αναδιάρθρωσης Bdd. Συνιστάται η χρήση του μόνο για πιο εξειδικευμένους χρήστες.
- Επιλογή **sjbgpwtmorn**: Καθορίζει ποιος αλγόριθμος πρέπει να χρησιμοποιηθεί για την βελτιστοποίηση boolean.
- Στο filename μπαίνει το αρχείο που θα επεξεργαστεί το BOOM
- Στο outname το όνομα που επιθυμούμε να έχει το ελαχιστοποιημένο αρχείο που θα δημιουργηθεί.

3.4 BOOG

Το εργαλείο BOOG(Binding and Optimizing On Gates) χαρτογραφεί μια περιγραφή behavioral σε ένα προκαθορισμένο πρότυπο cell βιβλιοθήκης. Το BOOG στις περισσότερες των περιπτώσεων χρησιμοποιείται στο αμέσως επόμενο βήμα μετά το BOOM και όπως και το BOOM αποτελούν εργαλεία της διαδικασίας σύνθεσης. Το BOOG δημιουργεί ένα δυαδικό δίκτυο ισοδύναμο με την περιγραφή που αποκτήθηκε ακριβώς στο προηγούμενο βήμα από το BOOM. Έτσι το εργαλείο για κάθε μια δυαδική συνάρτηση του κάθε κόμβου του δικτύου προσπαθεί να βρει ένα cell ή ακόμα και ένα σύνολο από cells για να υλοποιήσει την συνάρτηση. Το αποτέλεσμα αυτής της υλοποίησης θα είναι μια structural περιγραφή δηλαδή ένα αρχείο σε μορφή .**vst**.

Υλοποιούμε το BOOG ως εξής:

boog [-hmxold] input_file output_file [lax_file]

- Επιλογή **h**: Λειτουργία βοήθειας. Με την επιλογή αυτή εμφανίζονται οι επιλογές βοήθειας του εργαλείου.

- Επιλογή **m(**optim_mode): Λειτουργία βελτιστοποίησης. Ορίζεται σε ένα αρχείο lax, και είναι μόνο συντόμευση που ορίζεται στην γραμμή εντολών. Αυτός ο αριθμός ορίζεται μεταξύ του 0 και του 4 και είναι στον ευχέρεια του χρήστη τι αριθμό θα επιλέξει ανάλογα με την βελτιστοποίηση που θέλει να επιτύχει στο κύκλωμά του. Με την επιλογή 0 η περιοχή του κυκλώματος θα βελτιωθεί, με την επιλογή 4 θα βελτιωθούν καθυστερήσεις οι καθυστερήσεις του κυκλώματος. Με την επιλογή 2 επιτυγχάνεται μια μέση βελτιστοποίηση.
- Επιλογή **x**(xsch_mode): Δημιουργεί ένα αρχείο .xsch. Το αρχείο αυτό είναι ένας χρωματικός χάρτης για κάθε σήμα που περιέχεται στο αρχείο που παίρνει σαν είσοδο το συγκεκριμένο εργαλείο.
- Επιλογή **o**(output file): Ένας διαφορετικός τρόπος για να πάρει το αρχείο που θα δημιουργηθεί την κατάληξη .**vst** δηλαδή structural περιγραφή.
- Επιλογή **l**: Ένας ακόμη τρόπος για να χρησιμοποιηθεί το αρχείο lax.
- Επιλογή **d**(debug_file): Από τον εσωτερικό αλγόριθμο αποτελεσμάτων δημιουργείται ένα αρχείο με όνομα VBEdebug file.
- **Input_file**: Το όνομα του αρχείου που έχει ελαχιστοποιήσει το BOOM και θα βελτιστοποιήσει το BOOG.
- **Output_file**: Το όνομα που αρχείου που θα δημιουργηθεί με την εκτέλεση του BOOG.

Ένα πολύ σημαντικό βήμα για να λειτουργήσει σωστά το εργαλείο αυτό είναι ότι ο χρήστης πρέπει να έχει ρυθμίσει σωστά τις μεταβλητές περιβάλλοντος που χρησιμοποιεί το BOOG. Η μεταβλητή MBK_CATAL_LIB δίνει τις διαδρομές των βοηθητικών αρχείων εισόδου, behavioral αρχεία. Η μεταβλητή MBK_TARGET_LIB καθορίζει τη διαδρομή του καταλόγου της επιλεγμένης τυπικής cell βιβλιοθήκης. Καθώς και η μεταβλητή MBK_OUT_LO είναι αυτή που θέτει την μορφή εξόδου της structural περιγραφής δηλαδή την κατάληξη **.vst**.

3.5 COUGAR

Το COUGAR είναι ένας ¨εξωλκέας¨ που εφαρμόζεται σε ένα συμβολικό layout. Βέβαια μπορεί και σε ένα πραγματικό layout με την προϋπόθεση να παρέχεται η αντίστοιχη τεχνολογία από ένα αρχείο RDS. Το COUGAR βρίσκει χρήση μόνο σε τρανζίστορ CMOS και δεν μπορεί να υποστηρίξει άλλες τεχνολογίες. Ένα ακόμα χαρακτηριστικό είναι ότι μπορεί να εξάγει πληροφορίες για παρασιτική χωρητικότητα αλλά χωρίς μεγάλη αξιοπιστία.

Το COUGAR λειτουργεί στην γραμμή εντολών ως εξής:

cougar [-v] [-c] [-f] [-t] [-ar] [-ac] input_name [output_name]

- Επιλογή **v**: Η λειτουργία Verbose είναι ενεργοποιημένη. Κάθε βήμα της ανάλυσης εμφανίζεται στην τυπική έξοδο, με μερικά στατιστικά στοιχεία.
- Επιλογή **c**:
- Επιλογή **f**: Το cell συμβολικού layout ισοπεδώνεται στο επίπεδο καταλόγου πριν από την εξαγωγή. Χρησιμοποιείται το man catal για λεπτομέρειες στο συγκεκριμένο επίπεδο καταλόγου. Αν δεν υπάρχει ο κατάλογος ή είναι κενός, η netlist είναι μια διασύνδεση των τρανζίστορ, αλλιώς είναι μια διασύνδεση πυλών των οποίων τα ονόματα ορίζονται στον κατάλογο.
- Επιλογή **t**: Ειδοποιεί εξαγωγή σε επίπεδο τρανζίστορ, το cell του συμβολικού layout ισοπεδώνεται σε τρανζίστορ layout πριν την εξαγωγή.
- Επιλογή **ar**: Αποσυνδέει την αντίσταση διασύνδεσης και την χωρητικότητα στην γείωση. Η τιμή της αντίστασης για κάθε layer μπορεί να αλλάξει στο αρχείο RDS.
- Επιλογή **ac**: Αποσυνδέει την χωρητικότητα από την γείωση σε περίπτωση απώλειας.
- **Input_name**: Το όνομα του αρχείου που παίρνει ως είσοδο το COUGAR
- **Output_name**: Το όνομα του αρχείου που θα έχει ως έξοδο.

3.6 DREAL

Το DREAL είναι ένας ιεραρχικός επεξεργαστής πραγματικού layout. Χρησιμοποιούμαι το εργαλείο αυτό ως εξής:

dreal [-l file_name] [-xor] [-debug] [-install] [-force]

- Επιλογή **l file_name**: Φορτώνει το όνομα του αρχείου.
- Επιλογή **xor**: Χρησιμοποιεί την μέθοδο γραφικού xor. Η προεπιλεγμένη επιλογή είναι γραφικό invert.
- Επιλογή **install**: Εναλλάσσει σε ένα ιδιωτικό χρωματικό χάρτη.
- Επιλογή **force**: Όλα τα γραφικά αντικείμενα εμφανίζονται.

3.7 DRUC

DRUC (Design Rule Checker) είναι ένας παραμετροποιημένος έλεγχος κανόνων της VLSI. Όλοι οι απαραίτητοι για τον έλεγχο κανόνων παρέχονται από την μεταβλητή RDS_TECHNO_NAME. Το DRUC ελέγχει αν υπάρχει παραβίαση των καθορισμένων κανόνων σχεδιασμού, οπότε προϋποθέτει κατά την εφαρμογή να βεβαιωθεί ο χρήστης ότι το root και στιγμιαία cells βρίσκονται στον ίδιο φάκελο. Το DRUC ορίζεται στην γραμμή εντολών με τον εξής τρόπο:

```
druc -v muxoor
```

- Επιλογή **v**: Η λειτουργία Verbose είναι ενεργοποιημένη. Κάθε βήμα της ανάλυσης εμφανίζεται στην τυπική έξοδο.
- Επιλογή **h**: Έλεγχος κανόνα ιεραρχικού σχεδιασμού, δημιουργία τοπικών αρχείων για μελλοντική χρήση.

3.8 FlatBeh

Το εργαλείο FlatBeh(Flattens to Behavioral) δημιουργεί μια behavioral περιγραφή συμπεριφορά από ένα structural. Αυτό μπορεί να χρησιμοποιηθεί για να ελεγχθεί εάν η περιγραφή που λαμβάνεται στην διαδικασία σύνθεσης του σχεδίου είναι σωστή. Συνδέει την structural περιγραφή του σχεδιασμού στόχου μέχρι τα στοιχεία behavioral των φύλλων. Τότε, είναι σε θέση να χρησιμοποιήσει μια behavioral περιγραφή του σχεδιασμού στόχου.

Τρέχουμε το FlatBeh στην γραμμή εντολών ως εξής:

```
flatbeh root_structural_file [ output_file ]
```

Το συγκεκριμένο εργαλείο δεν έχει καμία επιλογή λειτουργίας. Ως μόνοι παράμετροι μπορούν να θεωρηθούν τα ονόματα των αρχείων που θα χρησιμοποιήσει δηλαδή τα structural και behavioral αρχεία που το τελευταίο θα είναι και το αρχείο εξόδου. Πρέπει ο χρήστης να προσέξει τις μεταβλητές περιβάλλοντος . Το MBK_CATAL_LIB πρέπει να έχει ρυθμιστεί στο path που έχουν δίνονται οι περιγραφές του εκάστοτε σχεδιασμού, η μεταβλητή MBK_IN_LO να έχει οριστεί σε vst επειδή το αρχείο που θα πάρει ως είσοδο το εργαλείο είναι structural.

3.9 FLATLO

Το εργαλείο με όνομα FLATLO(FLATtens Logical) εισάγει ένα ιεραρχικό netlist και ισοπεδώνει(το κάνει επίπεδο) σε ένα δεδομένο επίπεδο vst ή σε λογική μορφή al του Alliance.

Χρησιμοποιείται στο τερματικό με την ακόλουθη εντολή:

```
flatlo [option] logical_figure [instance] output_name
```

- Επιλογή **r**: Ισοπεδώνει τον target file στον κατάλογο
- Επιλογή **t**: Ισοπεδώνει τον target file σε επίπεδο τρανζίστορ.

3.10 FLATPH

Το FLATPH(FLATtens Physical) ένα ιεραρχικό συμβολικό layout σε ένα δεδομένο επίπεδο.

Το εργαλείο μπορεί να χρησιμοποιηθεί ως εξής:

```
flatph [option] logical_figure [instance] output_name
```

Το FLATPH έχει ακριβώς τις ίδιες επιλογές με το εργαλείο FLATLO.

3.11 FMI

Το εργαλείο FMI(Finite state machine Minimizer) παίρνει μία περιγραφή FMI και την ελαχιστοποιεί σε ένα ισοδύναμο μειωμένο, όσο αφορά των αριθμό των καταστάσεων, FSM.

Με την ακόλουθη εντολή ο χρήστης μπορεί να τρέξει εργαλείο αυτό:

```
fmi [-V] input_file output_file
```

Το FMI έχει μόνο μία επιλογή **V** η οποία ενεργοποιεί το Verbose mode και κάθε βήμα της ελαχιστοποίησης εμφανίζεται.

3.12 FSP

Το FSP(Finite State machine Proof) ελέγχει αν δύο περιγραφές FSM είναι πανομοιότυπες.

Χρησιμοποιείται με την ακόλουθη εντολή:

```
fsp [-V] format1 format2 file1 file2
```

Το FSP όπως και το FSM έχει μία μόνο επιλογή **V** η οποία ενεργοποιεί το Verbose mode και κάθε βήμα της ελαχιστοποίησης εμφανίζεται.

3.13 GENPAT

Το GENPAT(GENerator of PATterns) είναι ένα εργαλείο που χρησιμοποιείται για την δημιουργία αρχείων εισόδου pattern. Είναι αρχείο σε γλώσσα C,σύνολο από συναρτήσεις, που διευκολύνουν την δημιουργία pattern εισόδου. Αυτό το εργαλείο χρησιμοποιεί ένα αρχείο C που περιγράφει τα patterns εισόδου και δημιουργεί ένα αρχείο εξόδου **.pat**. Για μικρά κυκλώματα η επεξεργασία ενός αρχείο pattern είναι σχετικά απλή για κάποιο πιο περίπλοκο κύκλωμα η επεξεργασία γίνεται αρκετά πιο δύσκολη, έτσι το GETPAT αυτοματοποιεί την επεξεργασία αυτή.

Χρησιμοποιούμαι το GENPAT ως εξής:

```
genpat [-v] [-k] [file]
```

- Επιλογή **V**: Ενεργοποιεί το Verbose mode.
- Επιλογή **k**: Μετά την ολοκλήρωση διατηρεί το εκτελέσιμο και το μεταγλωττισμένο Makefile.

3.14 GRAAL

Το GRAAL είναι ένας editor για συμβολικό layout. Ουσιαστικά είναι ένα γραφικό εργαλείο που απαιτεί την εγκατάσταση X11 και του Motif.

Με την ακόλουθη εντολή συντάσσεται το GRAAL:

graal [-l file_name] [-scale n] [-debug] [-xor] [-install] [-force]

- Επιλογή **l file_name**: Φορτώνει το αρχείο με το αντίστοιχο όνομα
- Επιλογή **scale n**: Κλίμακα του ονόματος αρχείου
- Επιλογή **xor**: Χρησιμοποιούνται δύο μέθοδοι γραφικών η προεπιλεγμένη invert και η xor.
- Επιλογή **install**: Εναλλάσσει σε ένα ιδιωτικό χρωματικό χάρτη.
- Επιλογή **force**: Εμφανίζονται όλα τα γραφικά όλων των στοιχείων που χρησιμοποιούνται.

3.15 K2F

Το συγκεκριμένο εργαλείο του Alliance μεταφράζει μια περιγραφή FSM σε μορφή Berkeley (.kiss2) ή αντίστροφα. Η μορφή kiss μπορεί να χρησιμοποιηθεί για να απλοποιήσει μια περιγραφή FSM με τα εργαλεία SIS ή VIS τα οποία έχουν αναπτυχθεί στο πανεπιστήμιο Berkeley της Καλιφόρνια.

Με την ακόλουθη εκτελείται στην γραμμή εντολών:

k2f In_format Out_format Input_name [Output_name]

Το συγκεκριμένο εργαλείο δεν διαθέτει επιλογές.

3.16 L2P

Το L2P δημιουργεί ένα αρχείο PostScript(.ps) από ένα αρχείο εισόδου συμβολικού ή πραγματικού layout. Το PostScript είναι ένα λεξικό που απαρτίζεται από σύνολο μακροεντολών οι οποίες χρησιμοποιούνται κατά την διάρκεια της ερμηνείας ενός αρχείου PostScript. Ο χρήστης με τη τροποποίηση των μακροεντολών στα λεξικά του PostScript μπορεί να ελέγχει ποιο στρώμα θα εξάγει, πως να σχεδιάσει τα ορθογώνια, να καθορίσει τα χρώματα στα ορθογώνια και πολλά ακόμα. Αν το εργαλείο εκτελεστεί χωρίς επιλογές θα δημιουργήσει ένα τυπικό αρχείο εξόδου, το οποίο για μικρά κυκλώματα είναι αρκετό για σχέδια φύλλων cell.

Το L2P εκτελείται ως εξής με την παρακάτω εντολή:

l2p file_name

Για τα δικά μας απλά μικρά κυκλώματα θα χρησιμοποιούμαι το εργαλείο χωρίς κάποια από τις επιλογές του.

3.17 LOON

Το LOON(Light Optimizing On Nets) χρησιμοποιείται κατά την διαδικασία σύνθεσης μετά το BOOM και BOOG. Με την ακόλουθη εντολή χρησιμοποιείται το εργαλείο:

loon loon <input_file> <output_file> [-l <lax_file>] [-x <xsch_mode>] [-m <optim_mode>]

- Επιλογή **l_laxfile**: Ένας άλλος τρόπος για να δειχτεί το όνομα αρχείου παραμέτρου LAX
- Επιλογή **x xsch_mode**: Δημιουργεί ένα αρχείο **.xsch**, τα οποία είναι ένας χρωματικός χάρτης για τα σήματα εξόδου που περιέχονται στο αρχείο εξόδου. Το συγκεκριμένο αρχείο χρησιμοποιείται από το xsch κατά την προβολή του netlist. Υπάρχουν δύο επίπεδα για την λειτουργία xsch 0 ή 1 και μπορεί ο χρήστης να χρωματίσει αντίστοιχα την κρίσιμη διαδρομή ή όλα τα σήματα με διαβάθμιση καθυστέρησης.
- Επιλογή **m optmode**: Λειτουργία βελτιστοποίησης. Λαμβάνει τιμές μεταξύ 0 και 4 και υποδεικνύει τον τύπο βελτιστοποίησης. Αν επιλεχθεί 0 η περιοχή του κυκλώματος θα βελτιωθεί. Αν επιλεχθεί 4 θα βελτιωθούν οι καθυστερήσεις του κυκλώματος. Η επιλογή 2 είναι μία μέση τιμή για βελτιστοποίηση.

3.18 LVX

Το εργαλείο LVX(Logical Versus eXtracted) συγκρίνει δύο netlist σε επίπεδο gate-level ή block-level. Το εργαλείο αυτό χρησιμοποιείται για την σύγκριση μίας λογικής netlist με μια netlist που εξάγεται από μια φυσική netlist. Επίσης με το LVX μπορούμε να επαληθεύσουμε αν η τοποθέτηση(placement) και η δρομολόγηση(route) ενός λογικού σχεδιασμού έχουν γίνει σωστά. Η χρήση του LVX ακολουθεί του COUGAR καθώς χρησιμοποιεί την εξαγόμενη netlist του COUGAR για σύγκριση. Χρησιμοποιούμε το LVX με την ακόλουθη εντολή:

lvx format1 format2 filename1 filename2 [-a] [-o] [-f]

- Επιλογή **format1,format2**: Το είδος της μορφής που θα έχουν τα αρχεία που θα συγκριθούν. Το format1 θα είναι ένα αρχείο με κατάληξη **.vst** δηλαδή structural που αποκτήθηκε από το την εκτέλεση του LOON και το format2 θα είναι ένα αρχείο με κατάληξη **.al** που αποκτήθηκε από την εκτέλεση του COUGAR.

- Επιλογή **a**: Κάποιο δρομολογητές δημιουργούν layout με πολλούς φυσικούς συνδέσμους(connectors) για πηγή(VDD) και γείωση(VSS). Υπάρχει περίπτωση αυτοί οι σύνδεσμοι να μην είναι εσωτερικά συνδεδεμένοι μεταξύ τους με αποτέλεσμα να έχουν διαφορετικά ονόματα στην εξαγομένη netlist. Έτσι με την επιλογή αυτή μειώνονται αυτοί οι σύνδεσμοι πριν από την σύγκριση.
- Επιλογή **f**: Οι δυο netlist ισοπεδώνονται στα φύλλα των cells που περιέχοντα στο αρχείο καταλόγου.

3.19 MOKA

Με το εργαλείο MOKA(Model checker Ancestor) ελέγχουμε μια περιγραφή FSM ή RTL χρησιμοποιώντας μια λίστα από τύπους που περιγράφονται σε ένα αρχείο CTL(Control Temporal Logic).

Χρησιμοποιούμε το MOKA με την ακόλουθη εντολή:

moka [-VDB] fsmfile ctlfile

- Επιλογή **V**: Λειτουργία Verbose ενεργοποιημένη.
- Επιλογή **D**: Ενεργοποιείται η λειτουργία εντοπισμού σφαλμάτων. Κάθε βήμα του μοντέλου ελέγχου περιγράφεται με λεπτομέρεια στην έξοδο.
- Επιλογή **D**: Το αρχείο εισαγωγής είναι μια περιγραφή VHDL.

3.20 NERO

Το NERO(Negotiating ROuter) είναι ένας απλός δρομολογητής πάνω από το cell και μπορεί να επεξεργαστεί σχέδια ακόμα και σε πύλες με μέγεθος 4K. Ως είσοδο παίρνει μια netlist και μια τοποθέτηση(placement) του σχεδίου.

Το NERO χρησιμοποιεί καθολική δρομολόγηση(global routing) αν η μισή περίμετρος του σχεδίου είναι μεγαλύτερη από 800 λάμδα. Όταν χρησιμοποιείται η συγκεκριμένη δρομολόγηση τα μεγαλύτερα δίκτυα θα δρομολογούνται πρώτα χρησιμοποιώντας πρώτα τα επίπεδα 3 και 4 και στην συνέχεια όλα τα υπόλοιπα δίκτυα θα δρομολογηθούν χρησιμοποιώντας τα εναπομείναντα διαθέσιμα επίπεδα. Στην καθολική δρομολόγηση χρησιμοποιούνται το λιγότερο 4 επίπεδα δρομολόγησης και τα δίκτυα δρομολογούνται από το συντομότερο στο μακρύτερο με τον ίδιο αλγόριθμο δρομολόγησης.

Χρησιμοποιούμε το NERO με την ακόλουθη εντολή:

nero [options] netlist layout

3.21 OCP

Το OCP είναι ένα εργαλείο που χρησιμοποιείται για τοποθέτηση(placement) του σχεδίου. Ως είσοδο το OCP δέχεται μια netlist τυπικών κελιών. Η netlist μπορεί να είναι της μορφής structural, EDIF. Για να μπορεί το εργαλείο να αποκτήσει την structural μορφή, δηλαδή ένα αρχείο με κατάληξη **.vst** που χρησιμοποιούμε στις δικές μας περιπτώσεις, πρέπει να οριστεί η μεταβλητή περιβάλλοντος MBK_IN_LO. Ακόμα ο χρήστης αν το θεωρεί απαραίτητο μπορεί να παρέχει στην τοποθέτηση ένα αρχείο **.ioc** που δείχνει την τοποθέτηση των συνδέσμων. Σαν έξοδος της τοποθέτησης θα είναι ένα αρχείο φυσικού layout με τοποθετημένα κελιά και συνδέσμους. Με την εντολή περιβάλλοντος MBK_OUT_LO καθορίζεται η μορφή της εξόδου που συνήθως είναι ένα αρχείο με κατάληξη .**ap**.

Χρησιμοποιούμε το OCP με την παρακάτω εντολή:

```
ocp [options] netlist outputname
```

- Επιλογή **c**: Το OCP θα τοποθετήσει τους connectors τυχαία σε κάθε πλευρά του κουτιού στήριξης(abutment).
- Επιλογή **ioc Name**: Η τοποθέτηση των connectors θα γίνει ανάλογα με όσα αναφέρονται στο αρχείο NAME.ioc.
- Επιλογή **v**: Λειτουργία Verbose είναι ενεργοποιημένη.

3.22 PROOF

Το PROOF συγκρίνει μια εξαγόμενη περιγραφή συμπεριφοράς από ένα layout με την αντίστοιχη behavioral συμπεριφορά του σχεδιασμού. Το εργαλείο yagle χρησιμοποιείται για να εξαχθεί μια behavioral συμπεριφορά από ένα layout, αλλά πλέον το εργαλείο αυτό δεν διατίθεται από το Alliance.

Χρησιμοποιούμε το εργαλείο αυτό με την ακόλουθη εντολή:

```
proof [-a] [-d] file1 file2
```

- Επιλογή **a:** Με την επιλογή αυτή καθιστάτε δυνατή η διατήρηση των κοινών βοηθητικών σημάτων. Με το εργαλείο PROOF διατηρούνται όλα τα ενδιάμεσα σήματα τα οποία έχουν το ίδιο όνομα και στις δύο περιγραφές, ένα κοινό σήμα θεωρείται ως είσοδος αλλά και έξοδος κάθε περιγραφής. Η επιλογή είναι χρήσιμη για μεγάλες περιγραφές που περιέχουν μεγάλες εξισώσεις. Χρησιμοποιείται επιτοπλείστον για εντοπισμό σφαλμάτων βήμα προς βήμα στις δύο περιγραφές ή ακόμα χρησιμοποιείται όταν το εργαλείο δεν βγάζει κάποιο αποτέλεσμα

- Επιλογή **d**: Όταν οι behavioral συμπεριφορές είναι διαφορετικές το PROOF εμφανίζει τα σφάλματα.

3.23 RING

Το εργαλείο RING είναι ένας pad ring δρομολογητής(router), που δεν έχει επιλογές. Ένα παράδειγμα αρχείου(. **rin)** που περιγράφει τον προσανατολισμό των εισόδων και εξόδων πολυπλέκτη που θα χρησιμοποιήσουμε παρακάτω ως παράδειγμα. Για να μπορέσει ο χρήστης να χρησιμοποιήσει το παρακάτω αρχείο πρέπει να παρέχει και ένα αρχείο structural του σχεδιασμού που δημιουργεί και περιλαμβάνει pads Εικόνα 3.3.

```
north(q,c)
south(b,a)
east(p_vssb)
west(p_vddb)
width(vss 50 vdd 80)
```

Εικόνα 3.3

Το εργαλείο RING χρησιμοποιείται με την εντολή στην γραμμή εντολών με τον ακόλουθο τρόπο:

ring source result [stat]

3.24 S2R

Το S2R(Symbolic to Layout) είναι ένα εργαλείο που μας επιτρέπει να μεταφράσουμε τον συμβολικό μας layout σε ένα πραγματικό. Για να επιτευχθεί αυτό το S2R χρησιμοποιεί ένα αρχείο τεχνολογίας το οποίο ορίζεται από την μεταβλητή περιβάλλοντος RDS_TECHNO_NAME. Με το κατάλληλο αρχείο τεχνολογίας η έξοδος θα είναι ένα αρχείο layout πλήρως ολοκληρωμένο. Ορίζουμε την μεταβλητή περιβάλλοντος RDS_OUT με σκοπό να προσδιορίσουμε τον τύπο του αρχείου εξόδου, ο προεπιλεγμένος τύπος είναι cif αλλά μπορεί να οριστεί και σε μορφή gds. Το όνομα καταλόγου που χρησιμοποιείται στην αντικατάσταση pad ορίζεται από την μεταβλητή περιβάλλοντος MBK_CATAL_NAME. Με την εντολή περιβάλλοντος RDS_IN καθορίζεται η μορφή των cells που χρησιμοποιούνται στην αντικατάσταση. Τέλος για να χρησιμοποιηθεί σωστά το εργαλείο πρέπει να ρυθμιστεί σωστά και μία ακόμα μεταβλητή περιβάλλοντος η MBK_IN_PH. Με την μεταβλητή αυτή καθορίζεται η μορφή του συμβολικού layout.

Τρέχουμε το S2R στο τερματικό με την εντολή :

s2r [-tc1rv] source [result]

- Επιλογή **t**: Είναι μια επιλογή με την οποία το S2R αποφεύγει τα DRC σφάλματα, αλλά είναι μία χρονοβόρα επιλογή.
- Επιλογή **c**: Διαγράφει όλους τους connectors και τα ονόματα κόμβων σε όλα τα επίπεδα ιεραρχίας.
- Επιλογή **l**: Δημιουργεί το ανώτερο επίπεδο cell με την κάθε περίπτωση να παρουσιάζονται ως μαύρα κουτιά. Με τον τρόπο αυτό μπορεί να επιτευχθεί σημαντική μείωση στο μέγεθος του αρχείου.
- Επιλογή **r**: Τα μαύρα κουτιά δεν αντικαθιστούνται και τα cells επισημαίνονται με το γράμμα G στο αρχείο catal. Δεν θα υπάρξει κάποια αντικατάσταση σε περίπτωση που ο χρήστης εισάγει κάποιο layout από κάποια εξωτερική συσκευή.
- Επιλογή **v**: Λειτουργία Verbose είναι ενεργοποιημένη.

3.25 SCAPIN

Το εργαλείο SCAPIN(SCAn Path Insertion) χρησιμοποιείται για να εισάγει διαδρομή σάρωσης στο netlist του εκάστοτε σχεδιασμού. Η διαδρομή σάρωσης που έχει εισαχθεί θα περιέχει όλους τους καταχωρητές που καθορίζονται στο αρχείο το οποίο καθορίζεται από ένα αρχείο pathfile(.**path).** Το συγκεκριμένο εργαλείο προσθέτει ακόμα τρεις νέους connectors στην netlist που είναι οι εξής εξωτερική σάρωση, σάρωση, και δοκιμή σάρωσης. Όπως στα περισσότερα εργαλεία έτσι και στο SCAPIN πρέπει να οριστούν ορισμένες μεταβλητές περιβάλλοντος για να λειτουργήσει το εργαλείο. Η μεταβλητή MBK_IN_LO και MBK_OUT_LO καθορίζουν τις μορφές των αρχείων εισόδου και εξόδου αντίστοιχα. Η μεταβλητή SCAN_PARAM_NAME που καθορίζει την θέση του αρχείου παραμέτρων, είναι αρχείο με κατάληξη .**scapin** και περιέχει ιδιότητες των cells που απαιτούνται κατά την εισαγωγή διαδρομής σάρωσης όπως ονόματα θυρών, ονόματα μοντέλων και άλλα πολλά.

Τρέχουμε το εργαλείο SCAPIN με την παρακάτω εντολή:

scapin [-VRB] [-P file] infile pathfile outfile

- Επιλογή **V**: Λειτουργία Verbose είναι ενεργοποιημένη.
- Επιλογή **R**: Κάθε καταχωρητής της διαδρομής σάρωσης αντικαθιστούνται από ένα αντίστοιχο cell με δυνατότητα εγγραφής και ονομάζεται reg-mux.
- Επιλογή **B**: Προσθέτει έναν buffer πριν από την σάρωση του scan_out connector εξόδου.
- Επιλογή **P file**:Καθορίζει ένα αρχείο παραμέτρου .**scapin** το οποίο περιέχει ιδιότητες από όλα τα απαιτούμενα cell για την εισαγωγή της διαδρομής σάρωσης.

3.26 SYF

Το SYF(Synthesizer of Finite state machine) χρησιμοποιείται για την δημιουργία περιγραφών ροής δεδομένων VHDL από μια περιγραφή FSM, η περιγραφή αυτή χρησιμοποιεί μια εσωτερική στοίβα.

Χρησιμοποιούμε το παραπάνω εργαλείο με την ακόλουθη εντολή:

syf -a|j|m|u|o|r [-CDEOPRSTV] input [output]

- Επιλογή **a**: Χρησιμοποιεί το "ASP" ως αλγόριθμο κωδικοποίησης.
- Επιλογή **j**: Χρησιμοποιεί το "Jedi" ως αλγόριθμο κωδικοποίησης.
- Επιλογή **m**: Χρησιμοποιεί το "Mustang" ως αλγόριθμο κωδικοποίησης.
- Επιλογή **u**: Χρησιμοποιεί κωδικοποίηση που δίνεται από το αρχείο **input.enc**. Ένα σωστό κομμάτι του αρχείου αυτού περιέχει όνομα κατάστασης που ακολουθείτε από τον δεκαεξάρικό κώδικά της.
- Επιλογή **o**: Χρησιμοποιεί αλγόριθμο κωδικοποίησης **one hot.** Το one-hot είναι μια ομάδα από bit μεταξύ των οποίων οι αποδεκτοί συνδυασμοί τιμών είναι μόνο εκείνοι με ένα μόνο high (1) bit και όλοι οι άλλοι να είναι low (0) bit.
- Επιλογή **r**: Χρησιμοποιεί διακριτούς τυχαίους αριθμούς για κωδικοποίηση κατάστασης.
- Επιλογή **C**: Ελέγχει την συνοχή των μεταβάσεων.
- Επιλογή **D**: Με την συγκεκριμένη επιλογή το εργαλείο βελτιστοποιεί μόνο τους κώδικες που έχουν χρησιμοποιηθεί.
- Επιλογή **E**: Αποθηκεύει το αποτέλεσμα της κωδικοποίησης σε αρχείο με κατάληξη **.inc**. Το συγκριμένο αρχείο έχει ακριβώς την ίδια σύνταξη με το αρχείο που χρησιμοποιείται σαν είσοδο στην πιο πάνω επιλογή **u**.
- Επιλογή **O**: Το εργαλείο χρησιμοποιεί καταχωρητές στις εξόδους.
- Επιλογή **P**: Εφαρμόζει μια διαδρομή σάρωσης για τους καταχωρητές στοίβας, καταχωρητές κατάστασης και τους καταχωρητές εξόδου. Ο χρήστης πρέπει να χρησιμοποιήσει το εργαλείο SCAPIN που είδαμε παραπάνω για μια σωστή εισαγωγή διαδρομής σάρωσης σε μια netlist.
- Επιλογή **I**: Δεν αντιστρέφεται η πολικότητα των εξόδων.
- Επιλογή **R**: Είναι μια επιλογή που είναι διαθέσιμη μόνο για MOORE FSM δεν θα αναλυθεί καθώς δεν την χρησιμοποιούμε στους δικούς μας σχεδιασμούς.
- Επιλογή **S**: Το SYF δεν λαμβάνει υπόψη του το κόστος μεταβάσεων για τον υπολογισμό μιας κωδικοποίησης.

- Επιλογή **V**: Λειτουργία Verbose είναι ενεργοποιημένη.
- Επιλογή **F**: Ορίζει την μορφή της εξόδου.

3.27 VASY

Το εργαλείο VASY(VHDL Analyzer for sYnthesis) μετατρέπει μια περιγραφή VHDL σε μια ισοδύναμη συνθετική περιγραφή η οποία μπορεί να χρησιμοποιηθεί από το Alliance. Το VASY έχει την δυνατότητα να δημιουργεί αρχεία εξόδου σε μορφή VHDL ή Verilog.

Χρησιμοποιείται το εργαλείο VASY με την παρακάτω εντολή:

vasy [-VpavsoipSHL] [-C num] [-E num] [-I format] [-P file] filename [outname]

- Επιλογή **V**: Λειτουργία Verbose είναι ενεργοποιημένη, κάθε βήμα της ανάλυσης εμφανίζεται στην έξοδο.
- Επιλογή **p**: Προσθέτει βύσματα τροφοδοσίας στην πηγή και την γείωση.
- Επιλογή **a**: Μετατρέπει μια περιγραφή VHDL σε μορφή behavioral (vbe) ή structural(vst). Με την συγκεκριμένη επιλογή όλοι οι αριθμητικοί τελεστές επεκτείνονται σε ένα ισοδύναμο σύνολο από boolean εκφράσεις.
- Επιλογή **v**: Οδηγεί μια ισοδύναμη περιγραφή σε μορφή Verilog.
- Επιλογή **s**: Οδηγεί μια ισοδύναμη περιγραφή VHDL(.vhd).
- Επιλογή **o**:Εξουσιοδοτεί την αντικατάσταση υπαρχόντων αρχείων.
- Επιλογή **i**:Οδηγεί τις αρχικές τιμές σήματος, αλλά μόνο αυτές που προέρχονται από την επιλογή **s**.
- Επιλογή **p**: Προσθέτει βύσματα τροφοδοσίας στην πηγή και την γείωση όπως και η επιλογή **p** αλλά η συγκεκριμένη επιλογή χρησιμοποιείται πιο πολύ στο Alliance.
- Επιλογή **S**: Χρησιμοποιεί Std λογική αντί για bit, αλλά μόνο για τιμές που προέρχονται από την επιλογή **s**.
- Επιλογή **H**: Σε μια structural περιγραφή, όλα τα μοντέλα περιπτώσεων αναλύονται αναδρομικά. Τα φύλλα των cells καθορίζονται από ένα αρχείο που ονομάζεται CATAL.
- Επιλογή **L**: Δημιουργείται ένα αρχείο **.lax** το οποίο περιέχει λίστα όλων των σημάτων που πρέπει να διατηρείται κατά το στάδιο της σύνθεσης με την χρήση του BOOM.
- Επιλογή **E**: Τα εργαλεία σύγκρισης επεκτείνονται σε ένα ισοδύναμο σύνολο Boolean εκφράσεων, όταν το μέγεθός τους είναι μεγαλύτερο από τον αριθμό. Η επιλογή αυτή λαμβάνει υπόψιν μόνο την επιλογή **a**.
- Επιλογή **I format**: Με την επιλογή αυτή καθορίζεται η μορφή της εισόδου, όπως behavioral (vbe), structural(vst).

- Επιλογή **P file**: Καθορίζει ένα αρχείο **file.pkg** το οποίο περιέχει μια λίστα λογικού και φυσικού ονόματος πακέτου.
- Επιλογή **filename(outname)**: Το όνομα του αρχείου εξόδου.

3.28 X2Y

Το εργαλείο X2Y είναι ένας μετατροπέας netlist. Είναι ένα εργαλείο που δεν έχει επιλογές αλλά μόνο τα format που υποστηρίζει.

Το X2Y χρησιμοποιείται στην γραμμή εντολών με την ακόλουθη εντολή:

x2y informat outformat infile outfile

- **Informat** και **Outformat** είναι το format που θα έχουν τα αρχεία εισόδου και εξόδου αντίστοιχα.
- **Infile** και **Outfile** είναι τα αρχεία εισόδου και εξόδου αντίστοιχα.

Στην παρακάτω Εικόνα 3.4 φαίνονται τα format που μπορεί να διαχειριστεί το εργαλείο X2Y.

Option	Description
al	ALLIANCE netlist
ap	ALLIANCE layout
cct	HILO netlist
cp	VTI layout
edi	EDIF netlist or layout
fne	VTI extracted netlist
hns	VTI netlist
spi, sp, cir	SPICE netlist
vlg	VERILOG netlist
vst	VHDL netlist

Εικόνα 3.4

3.29 XFSM

Το XFSM είναι μόνο ένα γραφικό εργαλείο για την απεικόνιση των μηχανών πεπερασμένων καταστάσεων FSM και δεν διαθέτει κάποια επιλογή.

Το καλούμε από την γραμμή εντολών απλά με το όνομα του δηλαδή xfsm.

3.30 XPAT

Το XPAY(X window PAT file visualizer) είναι και αυτό ένα εργαλείο που παρουσιάζει τα αρχεία **.pat**.

Χρησιμοποιείται με την εξής εντολή:

xpat [-l file_name] [-xor] [-install] [-force]

- Επιλογή **l file_name**: Φορτώνει το όνομα του αρχείου.
- Επιλογή **xor**: Χρησιμοποιούνται δύο μέθοδοι γραφικών η προεπιλογή invert ή xor.
- Επιλογή **install**: Εναλλαγή σε ένα ιδιωτικό χρωματικό χάρτη.
- Επιλογή **force**: Εμφανίζονται όλα τα γραφικά αντικείμενα.

3.31 XSCH

Όπως και τα παραπάνω εργαλεία που είδαμε έτσι και το XSCH είναι ένα γραφικό εργαλείο x-window για την απεικόνιση των σχηματικών.

Χρησιμοποιείται με την παρακάτω εντολή:

xsch [-l file_name] [-xor] [-install] [-force] [-I input_format] [-slide file_name ...]

- Επιλογή **l file_name**: Φορτώνει το όνομα του αρχείου.
- Επιλογή **xor**: Χρησιμοποιούνται δύο μέθοδοι γραφικών η προεπιλογή invert ή xor.
- Επιλογή **install**: Εναλλαγή σε ένα ιδιωτικό χρωματικό χάρτη.
- Επιλογή **force**: Εμφανίζονται όλα τα γραφικά αντικείμενα.
- Επιλογή **I input_format**: Καθορίζει την μορφή εισόδου **vbe,vst, al.**
- Επιλογή **slide file_name**: Ενεργοποιεί την λειτουργία διαφανειών. Όλα τα αρχεία απεικονίζονται ένα προς ένα χρησιμοποιώντας τα πλήκτρα + ,-.

3.32 XVPN

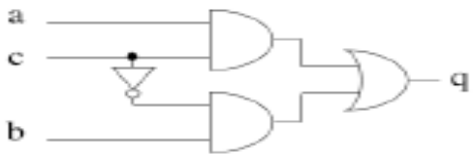
Το XVPN είναι και αυτό ένα εργαλείο x-window που απεικονίζει Petri nets. Το συγκεκριμένο εργαλείο δεν έχει επιλογές.

4. ΠΑΡΑΔΕΙΓΜΑ ΠΟΛΥΠΛΕΚΤΗ

Στο κεφάλαιο αυτό θα κάνουμε χρήση ενός παραδείγματος και πιο συγκεκριμένα ενός πολυπλέκτη με σκοπό την περαιτέρω ανάλυση των εργαλείων που θα χρησιμοποιήσουμε στο δικό μας σχεδιασμό. Θα γίνει ανάλυση των εργαλείων βήμα-βήμα και θα παρουσιαστούν και τα αποτελέσματα από το κάθε εργαλείο ώστε να είναι πιο κατανοητά και να είμαστε σε θέση να υλοποιήσουμε και να αναλύσουμε τον δικό μας πολλαπλασιαστή.

4.1 ΥΛΟΠΟΙΗΣΗ ΠΟΛΥΠΛΕΚΤΗ

Θα χρησιμοποιήσουμε έναν πολυπλέκτη 2 εισόδων του οποίου το σχήμα φαίνεται στην παρακάτω Εικόνα 4.1.



Εικόνα 4.1

Ξεκινώντας θα γράψουμε σε ένα αρχείο κειμενογράφου τον κώδικα που φαίνεται στην Εικόνα 4.2. Όπως βλέπουμε η περιγραφή του κυκλώματος είναι behavioral και έτσι θα αποθηκεύσουμε τον κώδικα με την κατάληξη **.vbe**.

```
-- mux.vbe : a discrete 2-input multiplexer circuit
-- VHDL behavioral description
ENTITY mux IS
PORT(
  a   : IN BIT;
  b   : IN BIT;
  c   : IN BIT;
  q   : OUT BIT;
  vdd : IN BIT;
  vss : IN BIT );
END mux;

ARCHITECTURE vbe OF mux IS
 BEGIN
  q <=((a AND c) OR (NOT(c) AND b));
 END vbe;
```

Εικόνα 4.2

Στο πολυπλέκτη αν η είσοδος **c** πάρει την τιμή 1 τότε η έξοδος **q** θα πάρει την τιμή που έχει η είσοδος **a**, αλλιώς αν η είσοδος **c** πάρει την τιμή 0 τότε η έξοδος **q** θα πάρει την τιμή που έχει η είσοδος **b.**

Είμαστε έτοιμοι να χρησιμοποιήσουμε το πρώτο εργαλείο το ASIMUT για να ελέγξουμε την ορθότητα της σύνταξης της περιγραφής του αρχείου του πολυπλέκτη. Για να μπορέσει όμως το ASIMUT να τρέξει σωστά πρέπει πρώτα να ρυθμίσουμε τις απαραίτητες μεταβλητές περιβάλλοντος. Θα τρέξουμε στην γραμμή εντολών την εντολή env | grep MBK

και το αποτέλεσμα φαίνεται στην Εικόνα 4.3. Η μεταβλητή περιβάλλοντος που είναι σημαντική για το ASIMUT είναι η MBK_IN_LO. Μπορούμε να δούμε ότι η μεταβλητή αυτή είναι ρυθμισμένη σε vst δηλαδή σε structural που είναι και ο προεπιλεγμένος τύπος περιγραφής για το Alliance.

```
[cadence@sonetto alliance]$ env | grep MBK
MBK_IN_PH=ap
MBK_SCALE_X=100
MBK_CATAL_NAME=CATAL
MBK_OUT_PH=ap
MBK_SPI_MODEL=/home/cadence/alliance/install/etc/spimodel.cfg
MBK_OUT_LO=vst
MBK_VSS=vss
MBK_C4_LIB=./cellsC4
MBK_VDD=vdd
MBK_TARGET_LIB=/home/cadence/alliance/install/cells/sxlib
MBK_IN_LO=vst
MBK_WORK_LIB=.
MBK_CATA_LIB=.:/home/cadence/alliance/install/cells/sxlib:/home/cadence/alliance
/install/cells/dp_sxlib:/home/cadence/alliance/install/cells/rflib:/home/cadence
/alliance/install/cells/rf2lib:/home/cadence/alliance/install/cells/ramlib:/home
/cadence/alliance/install/cells/romlib:/home/cadence/alliance/install/cells/pxli
b
```

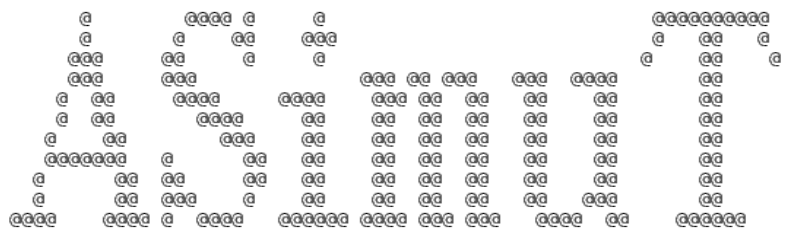
Εικόνα 4.3

Θα πρέπει να αλλάξουμε την τιμή της μεταβλητής MBK_IN_LO από **vst** σε **vbe** για να ελέγξουμε με το ASIMUT την behavioral περιγραφή του κυκλώματος. Όπως είδαμε και στο κεφάλαιο που αναλύσαμε το συγκεκριμένο εργαλείο αυτή η αλλαγή θα γίνει με την επιλογή **b.** Ακόμα θα χρησιμοποιήσουμε και την επιλογή **c** γιατί θέλουμε να εκτελεστεί μόνο το στάδιο της σύνταξης δηλαδή το εργαλείο να μεταγλωττίσει το αρχείο. Με την ακόλουθη εντολή θα ελέγξουμε τον σχεδιασμό μας:

```
asimut -b -c mux
```

Η έξοδος από την εκτέλεση του ASIMUT είναι η ακόλουθη:

```
[cadence@sonetto alliance]$ asimut -b -c mux
```




```
initializing ...
searching `mux` ...
BEH : Compiling `mux.vbe` (Behaviour) ...
making GEX ...
```

Εικόνα 4.4

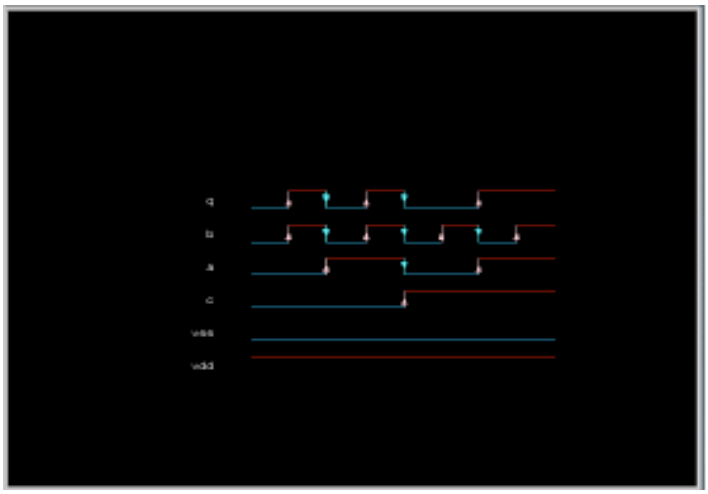
Όπως βλέπουμε από την Εικόνα 4.4 ο έλεγχος της ορθότητας σύνταξης για τον σχεδιασμό μας πραγματοποιήθηκε χωρίς λάθη. Αν υπήρχε κάποιο λάθος στην περιγραφή του κυκλώματος θα μας εμφάνιζε στην οθόνη το ASIMUT μήνυμα λάθους και σε ποια γραμμή μέσα στο αρχείο το είχε εντοπίσει.

Θα χρησιμοποιήσουμε ένα αρχείο εισαγωγής pattern ο κώδικας του οποίου φαίνεται στην Εικόνα 4.5 και θα τον αποθηκεύσουμε σε έναν κειμενογράφο με όνομα mux_in.pat. Το αρχείο pattern περιέχει περιγραφή για τις εισόδους και τις εξόδους του κυκλώματος που χρησιμοποιούμε καθώς και για την πηγή και την γείωση. Όπως βλέπουμε και στην εικόνα δίπλα από τις εισόδους χρησιμοποιούμε το **B** για δυαδικές τιμές. Αν θέλουμε να έχουμε οκταδικές τιμές βάζουμε το **O** και για δεκαεξαδικές τιμές βάζουμε το **X**. Στην συνέχεια παίρνουμε όλες τις πιθανές τιμές που μπορούν να υπάρξουν, ξεκινάμε από το 00 σε χρόνο 0 και σε κάθε 1 ns τροποποιούμε τις τιμές εισόδου για να εξετάσουμε το κύκλωμα για τους πιθανούς συνδυασμούς εισόδου. Με τα σύμβολα ?* δίπλα από κάθε τιμή λέμε στον simulator να υπολογίσει κάθε τιμή.

```
-- terminals of the mux circuit
in  vdd   B;
in  vss   B;
in  c     B;
in  a     B;
in  b     B;
out q   B;
 -- time interval description of the input values and expected (undetermined) output
begin
<0  ns> mux_test           : 10 000 ?*;
<+1 ns> notc_nota_notb     : 10 000 ?*;
<+1 ns> notc_nota_b        : 10 001 ?*;
<+1 ns>                    : 10 001 ?*;
<+1 ns> notc_a_notb        : 10 010 ?*;
<+1 ns>                    : 10 010 ?*;
<+1 ns> notc_a_b           : 10 011 ?*;
<+1 ns>                    : 10 011 ?*;
<+1 ns> c_nota_notb        : 10 100 ?*;
<+1 ns>                    : 10 100 ?*;
<+1 ns> c_nota_b           : 10 101 ?*;
<+1 ns>                    : 10 101 ?*;
<+1 ns> c_a_notb           : 10 110 ?*;
<+1 ns>                    : 10 110 ?*;
<+1 ns> c_a_b              : 10 111 ?*;
<+1 ns>                    : 10 111 ?*;
end;
```

Εικόνα 4.5

Θα χρησιμοποιήσουμε το γραφικό εργαλείο XPAT για να απεικονίσουμε το αρχείο mux_in.pat που μόλις δημιουργήσαμε. Στην Εικόνα 4.6 φαίνονται οι κυματομορφές των εισόδων και της εξόδου.



Εικόνα 4.6

Θα τρέξουμε πάλι το ASIMUT μαζί με το αρχείο pattern που έχουμε δημιουργήσει. Θα κάνουμε compile τον σχεδιασμό μας και το αρχείο εισόδου pattern και θα δημιουργηθεί ένα αρχείο εξόδου mux_out.pat.

Τρέχουμε το ASIMUT ως εξής και τα αποτελέσματά του φαίνονται στην Εικόνα 4.7:

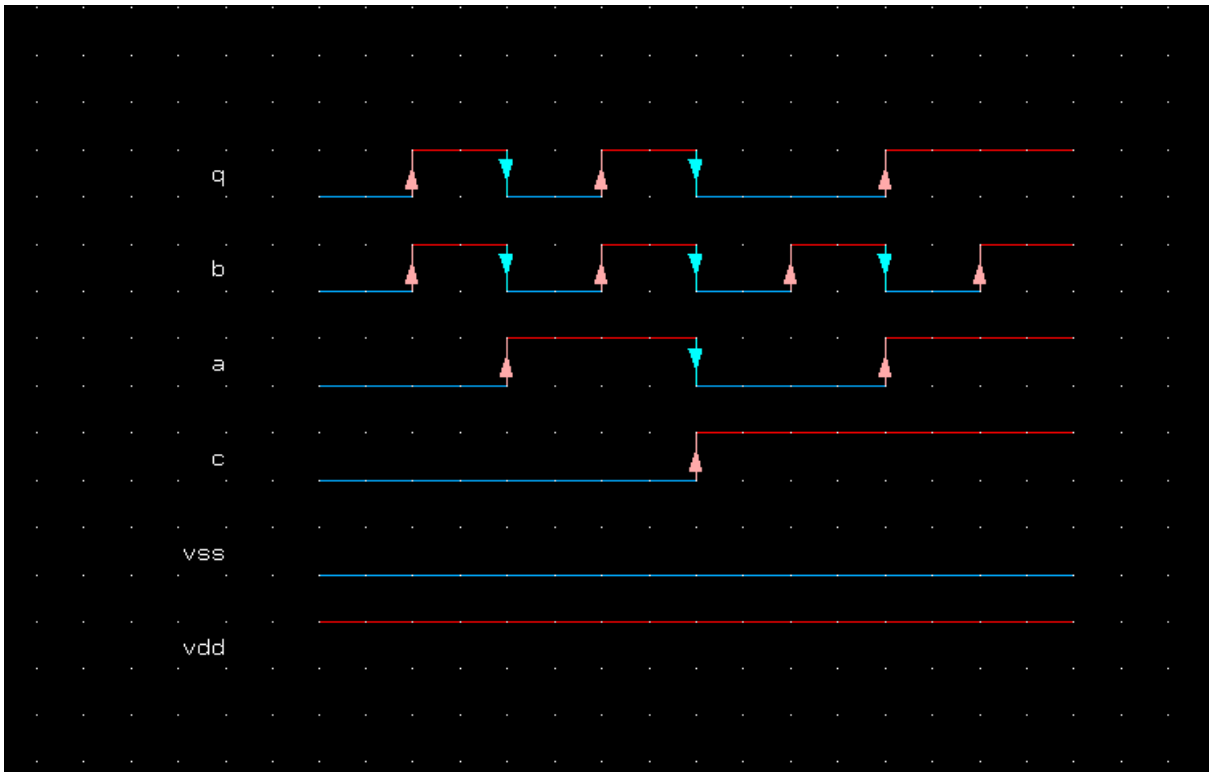
```
asimut -b mux mux_in mux_out
```

```
File  Edit  View  Search  Terminal  Help
       Paris, France, Europe, Earth, Solar system, Milky Way, ...
initializing ...
searching `mux` ...
BEH : Compiling `mux.vbe` (Behaviour) ...
making GEX ...

searching pattern file : `mux_in` ...
restoring ...

linking ...
executing ...
###----- processing pattern 0 : 0 ps -----###
###----- processing pattern 1 : 1000 ps -----###
###----- processing pattern 2 : 2000 ps -----###
###----- processing pattern 3 : 3000 ps -----###
###----- processing pattern 4 : 4000 ps -----###
###----- processing pattern 5 : 5000 ps -----###
###----- processing pattern 6 : 6000 ps -----###
###----- processing pattern 7 : 7000 ps -----###
###----- processing pattern 8 : 8000 ps -----###
###----- processing pattern 9 : 9000 ps -----###
###----- processing pattern 10 : 10000 ps -----###
###----- processing pattern 11 : 11000 ps -----###
###----- processing pattern 12 : 12000 ps -----###
###----- processing pattern 13 : 13000 ps -----###
###----- processing pattern 14 : 14000 ps -----###
###----- processing pattern 15 : 15000 ps -----###
```

Εικόνα 4.7

Γίνεται compile στο αρχείο mux.vbe, το εργαλείο χρησιμοποιεί το αρχείο εισόδου pattern, συσχετίζει τα αρχεία και εκτελεί στα διαστήματα που αναφέρονται στο pattern αρχείο.

Χρησιμοποιούμε πάλι το γραφικό εργαλείο XPAT για να απεικονίζουμε το αρχείο εξόδου mux_out.pat το οποίο φαίνεται στην Εικόνα 4.8.



Εικόνα 4.8

Οι τιμές έχουν δοθεί στις εισόδους a,b,c και τα αποτελέσματα φαίνονται στην έξοδο q. Βλέποντας την εικόνα μπορούμε να ελέγξουμε ότι το κύκλωμα συμπεριφέρεται σωστά γιατί όταν το c έχει την τιμή 0 η έξοδος q παίρνει την τιμή του της εισόδου b και όταν το c έχει την τιμή 1 η έξοδος q παίρνει την τιμή της εισόδου a.

Στη συνέχεια αφού τελειώσαμε με τον έλεγχο της συντακτικής ορθότητας του κυκλώματός μας θα προχωρήσουμε στην κατασκευή των φύλλων του πολυπλέκτη με τις πύλες AND, OR και INVERTER(NOT). Θα χρησιμοποιήσουμε τρείς κώδικες behavioral περιγραφής για κάθε μια από τις τρεις λογικές πύλες, τους οποίους θα γράψουμε σε ένα κειμενογράφο όνομα **invg.vbe, andg.vbe, org.vbe** ξεχωριστά για κάθε πύλη αντίστοιχα. Στην Εικόνα 4.9 φαίνονται οι τρεις κώδικες.

```
-- AND gate entity andg is port(a  : in  bit;
     b  : in  bit;
     q  : out bit;
   vdd  : in  bit;
   vss  : in  bit); end andg;

architecture vbe of andg is begin
  q <= (a AND b); end vbe;
```

```
-- OR gate entity org is port(a: in  bit;
     b: in  bit;
     q: out bit;
   vdd: in  bit;
   vss: in  bit); end org;

architecture vbe of org is begin
  q <= (a OR b); end vbe;
```

```
-- inverter (NOT) gate entity invg is port(a : in bit;
     x : out bit;
   vdd : in bit;
   vss : in bit); end invg;

architecture vbe of invg is begin
  x <= NOT(a); end vbe;
```

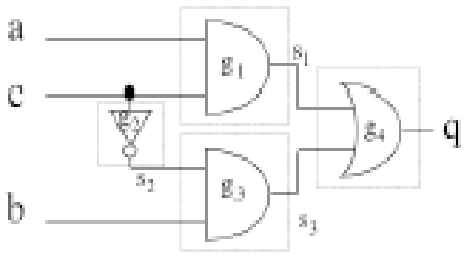
Εικόνα 4.9

Πλέον αφού έχουμε τα τρία φύλλα του σχεδιασμού μας είμαστε σε θέση να δημιουργήσουμε το τελικό αρχείο του πολυπλέκτη το οποίο θα είναι μια structural περιγραφή, που θα ενσωματώνει και τις τρεις λογικές πύλες που συνθέτουν το κύκλωμα. Στην αρχή του κώδικα βλέπουμε να ορίζονται οι είσοδοι, η έξοδος, γείωση και η πηγή του πολυπλέκτη. Στην συνέχεια τα φύλλα του σχεδιασμού μας, οι λογικές πύλες, συνθέτουν τα components στα οποία ορίζονται για κάθε component οι είσοδοι, έξοδοι, η πηγή και η γείωση αντίστοιχα. Ακολούθως πραγματοποιείται συσχετισμός μεταξύ ports του πολυπλέκτη με τα ports του κάθε component. Γράφουμε και αυτόν τον κώδικα όπως ακριβώς και τους προηγούμενους σε ένα αρχείο κειμένου με όνομα **mux.vst**. Στην παρακάτω Εικόνα 4.10 απεικονίζεται η structural περιγραφή του σχεδιασμού μας.

```
entity mux is
   port (
      a   : in   bit;
      b   : in   bit;
      c   : in   bit;
      q   : out  bit;
      vdd : in   bit;
      vss : in   bit
 ); end mux;

architecture structural of mux is

signal s1 : bit; signal s2 : bit; signal s3 : bit;

Component andg
   port (a  : in   bit;
         b  : in   bit;
         q  : out  bit;
        vdd : in   bit;
        vss : in   bit); end component;

Component org
   port (a  : in   bit;
         b  : in   bit;
         q  : out  bit;
        vdd : in   bit;
        vss : in   bit); end component;

Component invg
   port (a  : in   bit;
         q  : out  bit;
        vdd : in   bit;
        vss : in   bit); end component;

begin

g1 : andg port map (a  => a,
          b  => c,
          q  => s1,
          vdd => vdd,
          vss => vss);

g2 : invg port map (a  => c,
          q  => s2,
          vdd => vdd,
          vss => vss);

g3 : andg port map (a  => s2,
          b  => b,
          q  => s3,
          vdd => vdd,
          vss => vss);

g4 : org port map (a  => s1,
          b  => s3,
          q  => q,
          vdd => vdd,
          vss => vss);

end structural;
```

Εικόνα 4.10

Για πιο εύκολη κατανόηση των συσχετίσεων των ports μεταξύ τους μπορούμε να συμβουλευτούμε μαζί με τον κώδικα και την Εικόνα 4.11.



Εικόνα 4.11

Έχοντας έτοιμο το αρχείο με την structural περιγραφή του πολυπλέκτη θα χρησιμοποιήσουμε ένα άλλο εργαλείο του Alliance το BOOM με στόχο να πετύχουμε βελτιστοποίηση της συμπεριφοράς μας. Το BOOM θα εκτελέσει μια δυαδική ελαχιστοποίηση ώστε να επιτευχθεί βελτιστοποίηση στο κύκλωμα. Θα χρησιμοποιήσουμε το BOOM και για τις τρεις λογικές πύλες που έχουμε δημιουργήσει, δηλαδή στα αρχεία invg.vbe, andg.vbe, org.vbe και τα αποτελέσματά τους απεικονίζονται στην Εικόνα 4.12.

Τρέχουμε το εργαλείο βελτιστοποίησης BOOM στην γραμμή εντολών με τις ακόλουθες εντολές και για τις τρεις πύλες:

```
boom -l 3 -d 50 invg

boom -l 3 -d 50 andg

boom -l 3 -d 50 org
```

- Επιλογή **l 3**: Δίνει επίπεδο βελτιστοποίησης στο 3 (0-3 διαθέσιμες τιμές)
- Επιλογή **d 50**: Η καθυστέρηση και η περιοχή θα βελτιστοποιηθούν εξίσου και οι δύο. Τιμή 0% σημαίνει βελτιστοποίηση καθυστέρησης και τιμή 100% σημαίνει βελτιστοποίηση περιοχής.

Εικόνα 4.12

Διακρίνουμε από την εικόνα ότι το εργαλείο BOOM έτρεξε και μας έδωσε βελτιστοποιημένα αρχεία και για τις λογικές πύλες με ονόματα invg_o.vbe, andg_o.vbe και org_o.vbe. Με τα βελτιστοποιημένα σχέδια που έχουμε αποκτήσει θα χρησιμοποιήσουμε το εργαλείο BOOG. Το BOOG θα χαρτογραφήσει την behavioral περιγραφή αλλά θα δημιουργήσει και θα βελτιστοποιήσει μια structural περιγραφή χρησιμοποιώντας την τυπική βιβλιοθήκη cell **sxlib** που παρέχεται μαζί με στο σύστημα Alliance. Και σε αυτό το βήμα

θα τρέξουμε το BOOG και για τα τρία βελτιστοποιημένα αρχεία που έχουμε αποκτήσει από το BOOM

Με τις ακόλουθες εντολές εκτελούμε το BOOG και για τα τρία αρχεία αντίστοιχα:

```
boog invg_o invg -x 1 -m 2

boog andg_o andg -x 1 -m 2

boog org_o org -x 1 -m 2
```

- Επιλογή **x**: Δείχνει ότι το αρχείο που θέλουμε να δημιουργηθεί είναι ένα αρχείο εξόδου **.xsc.** Το αρχείο αυτό δείχνει τα χρησιμοποιούμενα cells και τις διαδρομές καθυστέρησης. Με την τιμή 1 δείχνουμε ότι όλες οι διαδρομές θα χρωματιστούν.
- Επιλογή **m**: Με την τιμή 2 βελτιστοποιείται η περιοχή αλλά και η καθυστέρηση εξίσου. Με την τιμή 0 πραγματοποιείται βελτιστοποίηση στην περιοχή ενώ με την τιμή 4 πραγματοποιείται βελτιστοποίηση στην καθυστέρηση.

  Τα αποτελέσματα μετά την εκτέλεση του εργαλείου παρουσιάζονται στις Εικόνες 4.13,4.14,4.15

```
50% area - 50% delay optimization
Reading file 'invg_o.vbe'...
Controlling file 'invg_o.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'invg_o.vbe'...
Capacitances on file 'invg_o.vbe'...
Unflattening file 'invg_o.vbe'...
Mapping file 'invg_o.vbe'...
Saving file 'invg.vst'...
Quick estimated critical path (no warranty)...116 ps from 'a' to 'q'
Quick estimated area (with over-cell routing)...750 lambda
Details...
        inv_x2: 1
        Total: 1
Saving delay gradient in xsch color file 'invg.xsc'...
```

Εικόνα 4.13

```
50% area - 50% delay optimization
Reading file 'andg_o.vbe'...
Controlling file 'andg_o.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'andg_o.vbe'...
Capacitances on file 'andg_o.vbe'...
Unflattening file 'andg_o.vbe'...
Mapping file 'andg_o.vbe'...
Saving file 'andg.vst'...
Quick estimated critical path (no warranty)...324 ps from 'b' to 'q'
Quick estimated area (with over-cell routing)...1250 lambda
Details...
        a2_x2: 1
        Total: 1
Saving delay gradient in xsch color file 'andg.xsc'...
```

Εικόνα 4.14

```
50% area - 50% delay optimization
Reading file 'org_o.vbe'...
Controlling file 'org_o.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'org_o.vbe'...
Capacitances on file 'org_o.vbe'...
Unflattening file 'org_o.vbe'...
Mapping file 'org_o.vbe'...
Saving file 'org.vst'...
Quick estimated critical path (no warranty)...358 ps from 'a' to 'q'
Quick estimated area (with over-cell routing)...1250 lambda
Details...
        o2_x2: 1
        Total: 1
Saving delay gradient in xsch color file 'org.xsc'...
```

Εικόνα 4.15

Με την εκτέλεση του εργαλείου θα δημιουργηθούν βελτιστοποιημένα αρχεία structural τα οποία θα είναι **invg.vst, andg.vst, org.vst.**

Έχοντας δημιουργήσει όλα τα αρχεία με τα εργαλεία BOOM, BOOG θα χρησιμοποιήσουμε πάλι το εργαλείο ASIMUT για να προσομοιώσουμε και να ελέγξουμε επίσης αν η περιγραφή εξακολουθεί να λειτουργεί σωστά. Σαν είσοδο στο εργαλείο θα βάλουμε το αρχείο structural του πολυπλέκτη mux.vst και το αρχείο εισόδου pattern mux_in.vst. Το αποτέλεσμα από την εκτέλεση του ASIMUT θα είναι ένα αρχείο εξόδου pattern με όνομα **muxst_out.pat**. Το συγκεκριμένο αρχείο θα το συγκρίνουμε με τα αρχεία και τα αποτελέσματά τους που αποκτήθηκαν από την behavioral περιγραφή. Ουσιαστικά θα συγκρίνουμε το αρχείο muxst_out.pat με το αρχείο pattern mux_out.pat. Σε αυτό το κομμάτι δεν θα

χρησιμοποιήσουμε την επιλογή **b** διότι το αρχείο mux είναι στην μορφή που θέλουμε, είναι σε μορφή .vst. Το χρησιμοποιούμε ως είσοδο οπότε δεν απαιτείται κάποια αλλαγή όπως θα δούμε και στην Εικόνα 4.16 η μεταβλητή περιβάλλοντος MBK_IN_LO είναι ήδη ρυθμισμένη να παίρνει ως είσοδο structural περιγραφή. Τρέχουμε την εντολή env | grep MBK και τα αποτελέσματά της παρουσιάζονται στην εικόνα που ακολουθεί.

```
[cadence@sonetto alliance]$ env | grep MBK
MBK_IN_PH=ap
MBK_SCALE_X=100
MBK_CATAL_NAME=CATAL
MBK_OUT_PH=ap
MBK_SPI_MODEL=/home/cadence/alliance/install/etc/spimodel.cfg
MBK_OUT_LO=vst
MBK_VSS=vss
MBK_C4_LIB=./cellsC4
MBK_VDD=vdd
MBK_TARGET_LIB=/home/cadence/alliance/install/cells/sxlib
MBK_IN_LO=vst
MBK_WORK_LIB=.
MBK_CATA_LIB=.:/home/cadence/alliance/install/cells/sxlib:/home/cadence/alliance/install/cells/dp_sxlib:/home/cadence/allia
nce/install/cells/rflib:/home/cadence/alliance/install/cells/rf2lib:/home/cadence/alliance/install/cells/ramlib:/home/caden
ce/alliance/install/cells/romlib:/home/cadence/alliance/install/cells/pxlib
[cadence@sonetto alliance]$
```

Εικόνα 4.16

Χρησιμοποιούμε το ASIMUT στην γραμμή εντολών με την εξής εντολή:

```
asimut mux mux_in muxst_out
```

Στην Εικόνα 4.17 που ακολουθεί παρουσιάζονται τα αποτελέσματα από την εκτέλεση του ASIMUT.

```
          @  @@       @@@@     @@    @@  @@  @@   @@    @@       @@
         @    @@        @@@    @@    @@  @@  @@   @@    @@       @@
         @@@@@@@   @      @@   @@    @@  @@  @@   @@    @@       @@
        @      @@  @@     @@   @@    @@  @@  @@   @@    @@       @@
        @      @@  @@@    @    @@    @@  @@  @@   @@   @@@       @@
       @@@@    @@@@ @  @@@@   @@@@@@ @@@@ @@@ @@@   @@@@  @@    @@@@@@

                              A SIMUlation Tool

                    Alliance CAD System 5.0,         asimut v3.02
                    Copyright (c) 1991...1999-2021, ASIM/LIP6/UPMC
                    E-mail        :    alliance-users@asim.lip6.fr

            Paris, France, Europe, Earth, Solar system, Milky Way, ...
initializing ...
searching `mux` ...
compiling `mux` (Structural) ...

flattening the root figure ...

searching `inv_x2` ...
BEH : Compiling `inv_x2.vbe` (Behaviour) ...
making GEX ...

searching `a2_x2` ...
BEH : Compiling `a2_x2.vbe` (Behaviour) ...
making GEX ...

searching `o2_x2` ...
BEH : Compiling `o2_x2.vbe` (Behaviour) ...
making GEX ...

searching pattern file : `mux_in` ...
restoring ...

linking ...
executing ...
###----- processing pattern 0 : 0 ps -----###
###----- processing pattern 1 : 1000 ps -----###
###----- processing pattern 2 : 2000 ps -----###
###----- processing pattern 3 : 3000 ps -----###
###----- processing pattern 4 : 4000 ps -----###
###----- processing pattern 5 : 5000 ps -----###
###----- processing pattern 6 : 6000 ps -----###
###----- processing pattern 7 : 7000 ps -----###
###----- processing pattern 8 : 8000 ps -----###
###----- processing pattern 9 : 9000 ps -----###
###----- processing pattern 10 : 10000 ps -----###
###----- processing pattern 11 : 11000 ps -----###
###----- processing pattern 12 : 12000 ps -----###
###----- processing pattern 13 : 13000 ps -----###
###----- processing pattern 14 : 14000 ps -----###
###----- processing pattern 15 : 15000 ps -----###
```

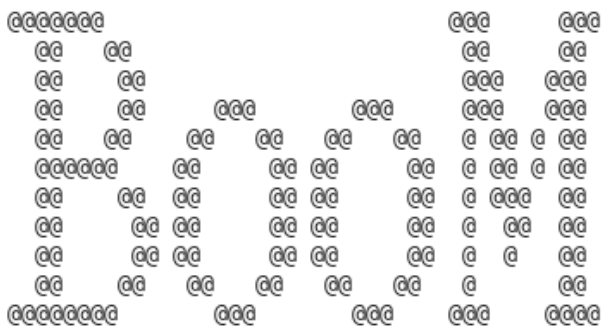
Εικόνα 4.17

Έχοντας ελέγξει με το ASIMUT την ορθότητα του σχεδιασμού μας και το έχουμε προσομοιώσει θα το βελτιστοποιήσουμε με το BOOM όπως ακριβώς κάναμε και πιο πάνω για κάθε μία λογική πύλη. Ακριβώς όπως και στην προηγούμενη χρήση του συγκεκριμένου εργαλείου έτσι και εδώ θα χρησιμοποιήσουμε μέγιστο επίπεδο βελτιστοποίησης καθυστέρησης και περιοχής.

Χρησιμοποιούμε το BOOM με την ακόλουθη εντολή και η έξοδος του παρουσιάζεται στην Εικόνα 4.18.

```
boom -l 3 -d 50 mux
```

```
[cadence@sonetto alliance]$ boom -l 3 -d 50 mux
```




```
--> Parse BEH file mux.vbe
--> Drive BEH file mux_o
```

Εικόνα 4.18

Βλέπουμε ότι από την εκτέλεση του BOOM προέκυψε ένα βελτιστοποιημένο αρχείο mux_o.vbe.

Το αρχείο mux_o.vbe θα χρησιμοποιηθεί ως είσοδο στο εργαλείο BOOG. Και σε αυτό το βήμα τρέχουμε το BOOG ακριβώς με τις ίδιες ιδιότητες όμως κάναμε και στο αντίστοιχο σημείο με τις λογικές πύλες. Θα βελτιστοποιήσουμε και την περιοχή και την καθυστέρηση. Επίσης θα απεικονιστούν όλες οι διαδρομές καθυστέρησης. Τα αποτελέσματα από τη εκτέλεση του BOOG φαίνονται στην Εικόνα 4.19.

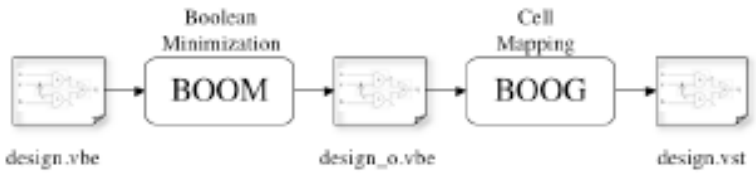
Με την εντολή που ακολουθεί κάνουμε χρήση του BOOG:

```
boog mux_o mux_oo -x 1 -m 2
```

```
[cadence@sonetto alliance]$ boog mux_o mux_oo -x 1 -m 2

                  @@@@@@@                            @@@@ @
                    @@   @@                        @@    @@
                    @@    @@                      @@      @
                    @@    @@     @@@       @@@    @@
                    @@   @@    @@   @@   @@   @@  @@
                    @@@@@@    @@     @@ @@     @@ @@     @@@@@
                    @@    @@  @@     @@ @@     @@ @@     @ @@
                    @@     @@ @@     @@ @@     @@ @@    @  @@
                    @@     @@ @@     @@ @@     @@  @@      @@
                    @@    @@   @@   @@   @@   @@    @@    @@
                   @@@@@@@@      @@@      @@@       @@@@

                          Binding and Optimizing On Gates

                   Alliance CAD System 5.0, boog 5.0 [2003/01/09]
                   Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
                   Author(s):                    Fran\ufffdois Donnet
                   E-mail        :    alliance-users@asim.lip6.fr

        MBK_VDD        : vdd
        MBK_VSS        : vss
        MBK_IN_LO      : vst
        MBK_OUT_LO     : vst
        MBK_WORK_LIB   : .
        MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
50% area - 50% delay optimization
Reading file 'mux_o.vbe'...
Controlling file 'mux_o.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'mux_o.vbe'...
Capacitances on file 'mux_o.vbe'...
Unflattening file 'mux_o.vbe'...
Mapping file 'mux_o.vbe'...
Saving file 'mux_oo.vst'...
Quick estimated critical path (no warranty)...657 ps from 'c' to 'q'
Quick estimated area (with over-cell routing)...3000 lambda\ufffd
Details...
        ao2o22_x2: 1
        inv_x2: 1
        Total: 2
Saving delay gradient in xsch color file 'mux_oo.xsc'...
End of boog...
```

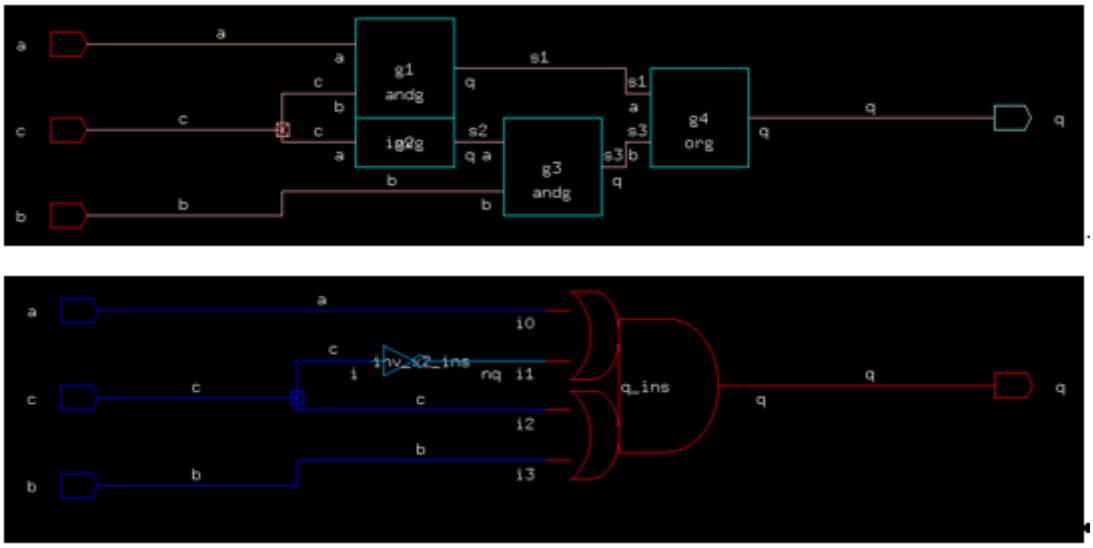
Εικόνα 4.19

Στην Εικόνα 4.20 φαίνεται πως από μια behavioral περιγραφή και από την εκτέλεση του BOOM παράγεται ένα βελτιστοποιημένο αρχείο στο οποίο έχει εφαρμοστεί δυαδική ελαχιστοποίηση, και στην συνέχεια με την εκτέλεση του BOOG παράγεται πάλι ένα βελτιστοποιημένο αρχείο στο οποίο έχει εφαρμοστεί βελτιστοποίηση στην καθυστέρηση, στην περιοχή καθώς και χαρτογράφηση.



Εικόνα 4.20

Χρησιμοποιώντας το γραφικό εργαλείο XSCH απεικονίζουμε τα αρχεία που έχουμε δημιουργήσει. Αρχικά θα απεικονίσουμε τον αρχικό μας structural σχεδιασμό mux.vst και στην συνέχεια τον βελτιστοποιημένο σχεδιασμό που αποκτήθηκε από το BOOG για να δούμε τις διαφορές τους. Στην Εικόνα 4.21 φαίνονται οι δύο σχεδιασμοί αντίστοιχα.



Εικόνα 4.21

Όπως μπορούμε να διακρίνουμε από την εικόνα ο αρχικός σχεδιασμός αποτελείται από πιο απλά και ξεκάθαρα στοιχεία. Ο βελτιστοποιημένος σχεδιασμός από έναν inverter και μία σύνθετη πύλη, που αποτελείται από δύο OR και μια AND πύλες. Το εργαλείο BOOG επέλεξε και τα δύο αυτά στοιχεία από την βιβλιοθήκη cell sxlib.

Σε αυτή την φάση έχοντας βελτιστοποιήσει τον σχεδιασμό μας και αποκτώντας το απαραίτητο αρχείο είμαστε σε θέση να περάσουμε στο επόμενο βήμα αυτό της τοποθέτησης(placement) και δρομολόγησης(route). Για να τοποθετήσουμε τον σχεδιασμό

μας χρησιμοποιήσουμε το εργαλείο τοποθέτησης OCP. Πριν από αυτό όμως πρέπει να ελέγξουμε τις μεταβλητές περιβάλλοντος που σχετίζονται με το OCP. Θα πρέπει η μεταβλητή MBK_IN_LO που καθορίζει τον τύπο του αρχείου εισόδου να έχει την τιμή **.vst** γιατί το αρχείο που θα πάρει ως είσοδο το OCP είναι structural περιγραφή που αποκτήθηκε από το BOOG. Η δεύτερη μεταβλητή που πρέπει να προσέξουμε είναι η MBK_OUT_PH που καθορίζει τον τύπο του αρχείου εξόδου και πρέπει να έχει την τιμή **.ap**. Τρέχουμε την εντολή env | grep MBK στο τερματικό και όπως διακρίνουμε στην Εικόνα 4.22 και οι δύο μεταβλητές είναι σωστά ορισμένες.

```
[cadence@sonetto alliance]$ env | grep MBK
MBK_IN_PH=ap
MBK_SCALE_X=100
MBK_CATAL_NAME=CATAL
MBK_OUT_PH=ap
MBK_SPI_MODEL=/home/cadence/alliance/install/etc/spimodel.cfg
MBK_OUT_LO=vst
MBK_VSS=vss
MBK_C4_LIB=./cellsC4
MBK_VDD=vdd
MBK_TARGET_LIB=/home/cadence/alliance/install/cells/sxlib
MBK_IN_LO=vst
MBK_WORK_LIB=.
MBK_CATA_LIB=.:/home/cadence/alliance/install/cells/sxlib:/home/cadence/alliance/install/cells/dp_sxlib:/home/cadence/allia
nce/install/cells/rflib:/home/cadence/alliance/install/cells/rf2lib:/home/cadence/alliance/install/cells/ramlib:/home/caden
ce/alliance/install/cells/romlib:/home/cadence/alliance/install/cells/pxlib
```

Εικόνα 4.22

Με τις μεταβλητές περιβάλλοντος σωστά ρυθμισμένες μπορούμε να εκτελέσουμε το OCP για να τοποθετήσουμε τον σχεδιασμό μας με την παρακάτω εντολή:

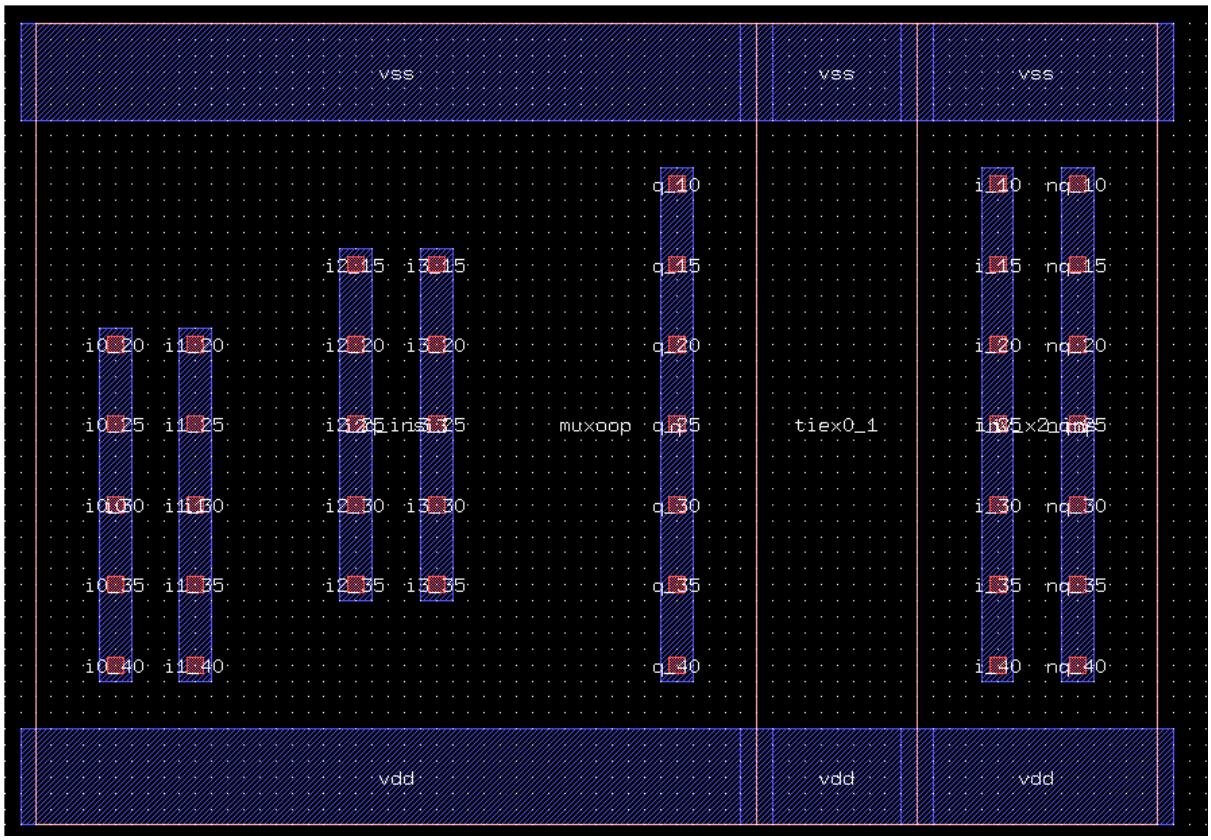
```
ocp mux_oo muxoop
```

Η εκτέλεση του εργαλείου τοποθέτησης παρουσιάζεται στην Εικόνα 4.23

```
[cadence@sonetto alliance]$ ocp mux_oo muxoop

                          @@@         @@@@ @ @@@@@@@
                        @@   @@     @@    @@   @@   @@
                       @@     @@   @@      @   @@    @@
                      @@       @@ @@       @   @@    @@
                      @@       @@ @@           @@   @@
                      @@       @@ @@           @@@@@
                      @@       @@ @@           @@
                       @@     @@   @@      @   @@
                        @@   @@     @@    @@   @@
                          @@@         @@@@   @@@@@@

                            Placer for Standards Cells

                    Alliance CAD System 5.0,            ocp 5.0
                    Copyright (c) 2001-2021,     ASIM/LIP6/UPMC
                    E-mail        : alliance-users@asim.lip6.fr

 o Special Net detected : vdd
 o Special Net detected : vss
 o No More Mouvement Possible .....

Ocp : placement finished
```

Εικόνα 4.23

Με την εκτέλεση του OCP πήραμε ως έξοδο το αρχείο muxoop.ap το οποίο να απεικονίσουμε με το εργαλείο GRAAL. Χρησιμοποιούμε το εργαλείο απεικόνισης GRAAL με την εξής εντολή graal. Στην Εικόνα 4.24 που ακολουθεί φαίνεται ο τοποθετημένος σχεδιασμός μας.



Εικόνα 4.24

Με την ολοκλήρωση της τοποθέτησης σειρά έχει η δρομολόγηση του σχεδιασμού. Η διαδικασία αυτή πραγματοποιείται με το εργαλείο NERO.

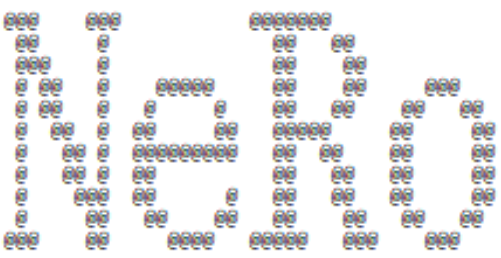
Εκτελούμε το NERO στην γραμμή εντολών ως εξής:

nero -V -p muxoop mux_oo muxoor

- Επιλογή **V**: Πάρα πολύ λεπτομερή διαδικασία.
- Επιλογή **p**: Καθορίζει ένα όνομα για το αρχείο τοποθέτησης διαφορετικό από το όνομα της netlist.
- Χρησιμοποιεί το αρχείο τοποθέτησης muxoop.ap της netlist mux_oo.vst που αποκτήσαμε από το BOOG και θα μας δώσει ως έξοδο ένα αρχείο muxoor.ap

Τα αποτελέσματα από την χρησιμοποίηση του NERO παρουσιάζονται στην Εικόνα 4.25.

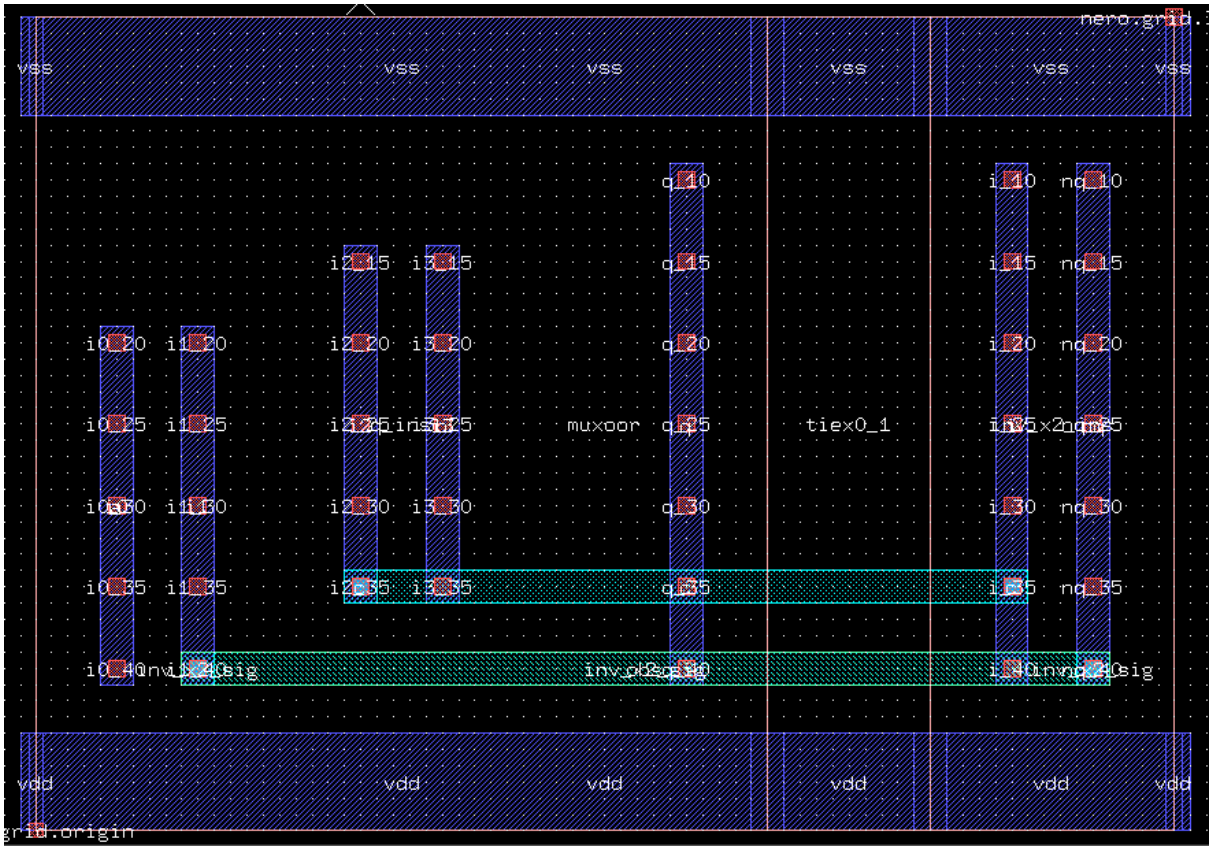
```
[cadence@sonetto alliance]$ nero -V -p muxoop mux_oo muxoor
```



```
                    Negotiating Router

                    Alliance CAD System 5.0,           nero 5.0
                    Copyright (c) 2002-2021,     ASIM/LIP6/UPMC
                    E-mail        : alliance-users@asim.lip6.fr

                                S/N 20120503.1


  o  MBK environment :

     MBK_IN_LO       := vst
     MBK_OUT_LO      := vst
     MBK_IN_PH       := ap
     MBK_OUT_PH      := ap
     MBK_WORK_LIB    := .
     MBK_CATA_LIB    := .
                        /home/cadence/alliance/install/cells/sxlib
                        /home/cadence/alliance/install/cells/dp_sxlib
                        /home/cadence/alliance/install/cells/rflib
                        /home/cadence/alliance/install/cells/rf2lib
                        /home/cadence/alliance/install/cells/ramlib
                        /home/cadence/alliance/install/cells/romlib
                        /home/cadence/alliance/install/cells/pxlib
     MBK_CATAL_NAME  := CATAL
     MBK_VDD         := vdd
     MBK_VSS         := vss
     MBK_SEPAR       := .

  o  Loading netlist "mux_oo"...
  o  Loading layout "muxoop"...
  o  Flattening layout...
  o  Flattening netlist...
  o  Building netlist dual representation (lofigchain)...
  o  Binding logical & physical views...

  o  Loading design into grid...
     o  Using seed cell "inv_x2_ins" (model "inv_x2").
     o  Grid offset : (0,0) [adjust (0,0)]
     o  Small design, global routing disabled.
     o  Allocating grid size [15,11,3].
     o  Loading external terminals.
     o  Finding obstacles.
     o  Loading nets into grid.
     o  Allocating the net scheduler.
     o  Reading power grid.

  o  Local routing stage.
     - [   6] (hp :=     0) "vdd"
     - [   5] (hp :=     0) "vss"
     - [   4] (hp :=     4) "a"
     - [   3] (hp :=     4) "b"
     - [   2] (hp :=     6) "q"
     - [   1] (hp :=    14) "c"
     - [   0] (hp :=    17) "inv_x2_sig"

  o  Routing stats :
     - routing iterations    := 169
     - re-routing iterations := 0
     - ratio                 := 0%.

  o  Dumping routing grid.
  o  Saving MBK figure "muxoor".
  o  Saving layout as "muxoor"...
```

Εικόνα 4.25

Χρησιμοποιώντας πάλι το εργαλείο GRAAL μπορούμε να δούμε το δρομολογημένο αρχείο muxoor.ap στην Εικόνα 4.26.



Εικόνα 4.26

Μπορούμε τώρα να προχωρήσουμε στο επόμενο βήμα που είναι η εξαγωγή από την αντίστοιχη netlist με το εργαλείο COUGAR. Όπως κάνουμε πριν από την χρήση κάθε εργαλείου έτσι και σε αυτό το σημείο θα ελέγξουμε τις μεταβλητές περιβάλλοντος. Η μεταβλητή που μας ενδιαφέρει είναι η MBK_OUT_LO καθώς το αρχείο εξόδου που θα παραχθεί από το COUGAR πρέπει να έχει την κατάληξη **.al**.

Τρέχουμε την εντολή env | grep MBK και όπως θα βλέπουμε στην Εικόνα 4.27 η μεταβλητή MBK_OUT_LO έχει την τιμή .vst την οποία πρέπει να την αλλάξουμε και να εκχωρήσουμε την τιμή .al. για να λειτουργήσει το εργαλείο COUGAR.

```
[cadence@sonetto alliance]$ env | grep MBK
MBK_IN_PH=ap
MBK_SCALE_X=100
MBK_CATAL_NAME=CATAL
MBK_OUT_PH=ap
MBK_SPI_MODEL=/home/cadence/alliance/install/etc/spimodel.cfg
MBK_OUT_LO=vst
MBK_VSS=vss
MBK_C4_LIB=./cellsC4
MBK_VDD=vdd
MBK_TARGET_LIB=/home/cadence/alliance/install/cells/sxlib
MBK_IN_LO=vst
MBK_WORK_LIB=.
MBK_CATA_LIB=.:/home/cadence/alliance/install/cells/sxlib:/home/cadence/alliance/install/cells/dp_sxlib:/home/cadence/allia
nce/install/cells/rflib:/home/cadence/alliance/install/cells/rf2lib:/home/cadence/alliance/install/cells/ramlib:/home/caden
ce/alliance/install/cells/romlib:/home/cadence/alliance/install/cells/pxlib
```

Εικόνα 4.27

Για να αλλάξουμε την τιμή της μεταβλητής γράφουμε στην γραμμή την εντολή export MBK_OUT_LO = al και στην συνέχεια γράφουμε την εντολή echo $MBK_OUT_LO για να δούμε αν άλλαξε η τιμή της μεταβλητής. Στην Εικόνα 4.28 φαίνεται η αλλαγή της μεταβλητής.

```
[cadence@sonetto alliance]$ export MBK_OUT_LO=al
[cadence@sonetto alliance]$ echo $MBK_OUT_LO
al
```

Εικόνα 4.28

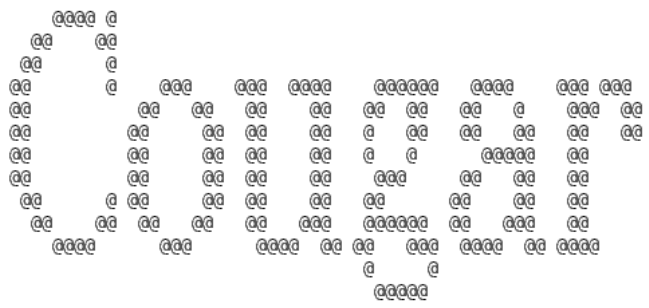
Τώρα είμαστε σε θέση να χρησιμοποιήσουμε το COUGAR για επαληθεύσουμε τον σχεδιασμό μας. Το COUGAR θα πάρει ως είσοδο το ήδη δρομολογημένο αρχείο συμβολικού layout muxoor.ap και θα παράγει το εξαγόμενο netlist muxooc.al.

Χρησιμοποιούμε το COUGAR με την παρακάτω εντολή:

```
cougar muxoor muxooc
```

Τα αποτελέσματα από την χρησιμοποίηση του COUGAR παρουσιάζονται στην Εικόνα 4.29.

```
[cadence@sonetto alliance]$ cougar muxoor muxooc
```



```
                Netlist extractor ... formerly Lynx

          Alliance CAD System 5.0,         cougar 1.21
          Copyright (c) 1998-2021,      ASIM/LIP6/UPMC
          Author(s): Ludovic Jacomme and Gregoire Avot
          Contributor(s):             Picault Stephane
          E-mail        :  alliance-users@asim.lip6.fr


      ---> Extract symbolic figure muxoor
      <--- done !

      ---> Total extracted capacitance
      <--- 0.0DF
```

Εικόνα 4.29

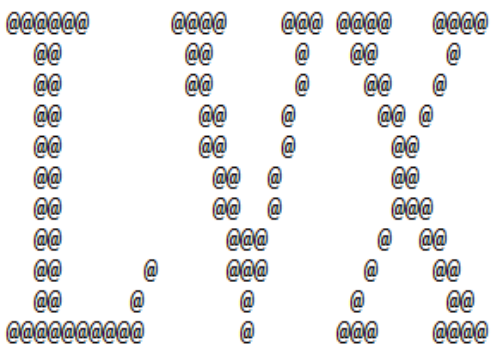
Με την ολοκλήρωση και του εργαλείου COUGAR και έχοντας επαληθεύσει τον σχεδιασμό θα χρησιμοποιήσουμε το LVX για να συγκρίνουμε το structural αρχείο mux_oo.vst και το netlist muxooc.al που αποκτήθηκε από το COUGAR. Γίνεται έλεγχος και σε αυτό το βήμα αν η τοποθέτηση και η δρομολόγηση ενός λογικού σχεδιασμού έχουν γίνει σωστά.

Χρησιμοποιούμε το εργαλείο LVX με την ακόλουθη εντολή:

```
lvx vst al mux_oo muxooc
```

Τα αποτελέσματα από την υλοποίηση του εργαλείου σύγκρισης φαίνονται στην Εικόνα 4.30.

```
[cadence@sonetto alliance]$ lvx vst al mux_oo muxooc
```



```
                    Gate Netlist Comparator

          Alliance CAD System 5.0,            lvx 1.5
          Copyright (c) 1992-2021,     ASIM/LIP6/UPMC
          E-mail       : alliance-users@asim.lip6.fr


***** Loading mux_oo (vst)...

***** Loading muxooc (al)...


***** Compare Terminals ...........
***** O.K.      (0 sec)

***** Compare Instances ..........
***** O.K.      (0 sec)

***** Compare Connections ..........
***** O.K.      (0 sec)

===== Terminals .......... 6
===== Instances .......... 2
===== Connectors ......... 17


***** Netlists are Identical. *****     (0 sec)
```

Εικόνα 4.30

Όπως διακρίνουμε αφού φορτωθούν τα δύο αρχεία και αφού περνάει από 3 στάδια σύγκρισης μας βγάζει το μήνυμα ότι οι δυο netlist είναι πανομοιότυπες. Δεδομένου αυτού συμπεραίνουμε ότι ο σχεδιασμός έχει τοποθετηθεί και δρομολογηθεί σωστά.

Το COUGAR έχει επίσης την δυνατότητα να εξάγει και spice netlist από τον σχεδιασμό που είναι σε μορφή **ap**. Μια spice netlist είναι μια αναπαράσταση ενός κυκλώματος που βασίζεται σε κείμενο. Με την προβολή της netlist μας βοηθά να μάθουμε για την σύνταξη και την προσομοίωση του spice. Μας βοηθά ακόμα σε αρκετές περιπτώσεις στον εντοπισμό σφαλμάτων προσομοίωσης. Μια netlist επιπέδου spice είναι η network netlist RC(

Resistance and Capacitance) που περιέχει πληροφορίες για τις τάσεις τροφοδοσίας και την γείωση. Το spice netlist χρησιμοποιείται για σχεδίαση γραφήματος ρεύματος αντί για γραφήματος τάσης. Το spice είναι παρά πολύ συσχετισμένο με την πραγματική συμπεριφορά του κυκλώματος.[7][8]

Για να χρησιμοποιήσουμε το COUGAR και να εξάγουμε spice netlist θα πρέπει να αλλάξουμε τις μεταβλητές περιβάλλοντος MBK_OUT_LO και MBK_SPI_MODEL.

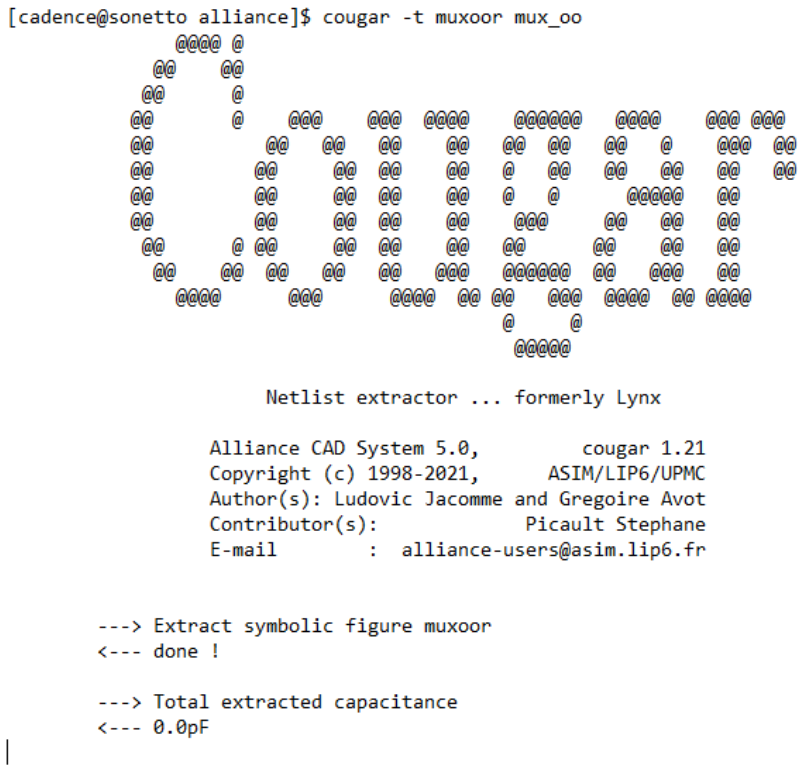
Για να αλλάξουμε την τιμή της μεταβλητής γράφουμε στην γραμμή την εντολή export MBK_OUT_LO = spi και στην συνέχεια γράφουμε την εντολή echo $MBK_OUT_LO για να δούμε αν άλλαξε η τιμή της μεταβλητής. Την ίδια διαδικασία εκτελούμε και για μεταβλητή MBK_SPI_MODEL. Γράφουμε στην γραμμή εντολών export MBK_SPI_MODEL= $ALLIANCE_TOP/etc/spimodel.cfg και στην συνέχεια γράφουμε την εντολή για να δούμε αν άλλαξε η τιμή της μεταβλητής echo $MBK_SPI_MODEL.

Χρησιμοποιούμε πάλι το COUGAR για να εξάγουμε την spice netlist αυτή την φορά με την ακόλουθη εντολή γραμμής εντολών:

```
cougar -t muxoor mux_oo
```

- Επιλογή **t**: Εξαγωγή επιπέδου τρανζίστορ.
- Αρχείο muxoor.ap το αρχείο που δέχεται ως είσοδο και είναι το τοποθετημένο αρχείο που αποκτήθηκε από το OCP και το αρχείο mux_oo.spi είναι το αρχείο εξόδου.

Στην Εικόνα 4.31 παρουσιάζεται το αποτέλεσμα από την υλοποίηση του COUGAR για την spice netlist.



Εικόνα 4.31

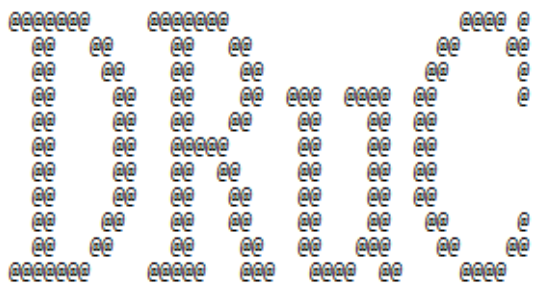
Έχοντας συγκρίνει τα αρχεία μας με το εργαλείο LVX και είδαμε ότι είναι πανομοιότυπα καθώς και ο σχεδιασμός μας είναι σωστά τοποθετημένος και δρομολογημένος προχωράμε στο επόμενο βήμα που είναι ο έλεγχος κανόνων VLSI χρησιμοποιώντας το εργαλείο DRUC στο αρχείο muxoor.ap που αποκτήθηκε από τον OCP.

Χρησιμοποιούμε την εντολή που ακολουθεί για την υλοποίηση του DRUC και τα αποτελέσματά του φαίνονται στις Εικόνες 4.32, 4.33, 4,34, 4.35, 4.36, 4.37.

druc -v muxoor

- Επιλογή **v**: Λειτουργία Verbose ενεργοποιημένη. Κάθε βήμα από την υλοποίηση του DRC εμφανίζεται στην έξοδο.

```
[cadence@sonetto alliance]$ druc -v muxoor
```



```
                    Design Rule Checker

                  Alliance CAD System 5.0,           druc 5.0
                  Copyright (c) 1993-2021,     ASIM/LIP6/UPMC
                  E-mail        : alliance-users@asim.lip6.fr

Flatten DRC on: muxoor
Delete MBK figure : muxoor
Load Flatten Rules : /home/cadence/alliance/install/etc/cmos.rds

layer RDS_NWELL 4.;
layer RDS_NTIE 2.;
layer RDS_PTIE 2.;
layer RDS_NDIF 2.;
layer RDS_PDIF 2.;
layer RDS_ACTIV 2.;
layer RDS_CONT 1.;
layer RDS_VIA1 1.;
layer RDS_VIA2 1.;
layer RDS_VIA3 1.;
layer RDS_VIA4 1.;
layer RDS_VIA5 1.;
layer RDS_POLY 1.;
layer RDS_POLY2 1.;
layer RDS_ALU1 1.;
layer RDS_ALU2 2.;
layer RDS_ALU3 2.;
layer RDS_ALU4 2.;
layer RDS_ALU5 2.;
layer RDS_ALU6 2.;
layer RDS_USER0 1.;
layer RDS_USER1 1.;
layer RDS_USER2 1.;

regles

# Note : ``min'' is different from ``>=''.
# min is applied on polygons and >= is applied on rectangles.
# There is the same difference between max and <=.
# >= is faster than min, but min must be used where it is
# required to consider polygons, for example distance of
# two objects in the same layer
#
# There is no rule to check NTIE and PDIF are included in NWELL
# since this is necessarily true
#-------------------------------------------------------------

# Check the NWELL shapes
#-----------------------
caracterise RDS_NWELL (
  regle 1 : largeur >= 4. ;
  regle 2 : longueur_inter min 4. ;
  regle 3 : notch >= 12. ;
);
relation RDS_NWELL , RDS_NWELL (
  regle 4 : distance axiale min 12. ;
);
```

Εικόνα 4.32

```
# Check RDS_PTIE is really excluded outside NWELL
#-----------------------------------------------
relation RDS_PTIE , RDS_NWELL (
  regle 5 : distance axiale >= 7.5;
  regle 6 : enveloppe longueur_inter < 0. ;
  regle 7 : marge longueur_inter < 0. ;
  regle 8 : croix longueur_inter < 0. ;
  regle 9 : intersection longueur_inter < 0. ;
  regle 10 : extension longueur_inter < 0. ;
  regle 11 : inclusion longueur_inter < 0. ;
);

# Check RDS_NDIF is really excluded outside NWELL
#-----------------------------------------------
relation RDS_NDIF , RDS_NWELL (
  regle 12 : distance axiale >= 7.5;
  regle 13 : enveloppe longueur_inter < 0. ;
  regle 14 : marge longueur_inter < 0. ;
  regle 15 : croix longueur_inter < 0. ;
  regle 16 : intersection longueur_inter < 0. ;
  regle 17 : extension longueur_inter < 0. ;
  regle 18 : inclusion longueur_inter < 0. ;
);

# Check the RDS_PDIF shapes
#--------------------------
caracterise RDS_PDIF (
  regle 19 : largeur >= 2. ;
  regle 20 : longueur_inter min 2. ;
  regle 21 : notch >= 3. ;
);
relation RDS_PDIF , RDS_PDIF (
  regle 22 : distance axiale min 3. ;
);

# Check the RDS_NDIF shapes
#--------------------------
caracterise RDS_NDIF (
  regle 23 : largeur >= 2. ;
  regle 24 : longueur_inter min 2. ;
  regle 25 : notch >= 3. ;
);
relation RDS_NDIF , RDS_NDIF (
  regle 26 : distance axiale min 3. ;
);

# Check the RDS_PTIE shapes
#--------------------------
caracterise RDS_PTIE (
  regle 27 : largeur >= 2. ;
  regle 28 : longueur_inter min 2. ;
  regle 29 : notch >= 3. ;
);
relation RDS_PTIE , RDS_PTIE (
  regle 30 : distance axiale min 3. ;
);

# Check the RDS_NTIE shapes
#--------------------------
caracterise RDS_NTIE (
  regle 31 : largeur >= 2. ;
  regle 32 : longueur_inter min 2. ;
  regle 33 : notch >= 3. ;
);
relation RDS_NTIE , RDS_NTIE (
  regle 34 : distance axiale min 3. ;
);

define RDS_PDIF , RDS_PTIE union -> ANY_P_DIF;
define RDS_NDIF , RDS_NTIE union -> ANY_N_DIF;
```

Εικόνα 4.33

```
define RDS_NDIF , RDS_NTIE union -> ANY_N_DIF;

# Check the ANY_N_DIF ANY_P_DIFF exclusion
#-------------------------------------
relation ANY_N_DIF , ANY_P_DIF (
  regle 35 : distance axiale >= 3. ;
  regle 36 : enveloppe longueur_inter < 0. ;
  regle 37 : marge longueur_inter < 0. ;
  regle 38 : croix longueur_inter < 0. ;
  regle 39 : intersection longueur_inter < 0. ;
  regle 40 : extension longueur_inter < 0. ;
  regle 41 : inclusion longueur_inter < 0. ;
);

undefine ANY_P_DIF;
undefine ANY_N_DIF;

define RDS_NDIF , RDS_PDIF union -> NP_DIF;

# Check RDS_POLY related to NP_DIF
#---------------------------------
relation RDS_POLY , NP_DIF (
  regle 42 : distance axiale >= 1. ;
  regle 43 : intersection longueur_inter < 0. ;
);

define NP_DIF , RDS_POLY intersection -> CHANNEL;

# Check the RDS_POLY shapes
#--------------------------
caracterise RDS_POLY (
  regle 44 : largeur >= 1. ;
  regle 45 : longueur_inter min 1. ;
  regle 46 : notch >= 2. ;
);
relation RDS_POLY , RDS_POLY (
  regle 47 : distance axiale min 2.;
);

define NP_DIF , RDS_CONT intersection -> CONT_DIFF;
# Check the CHANNEL shapes
#-------------------------
caracterise CHANNEL (
  regle 48 : notch >= 3. ;
);
relation CHANNEL , CHANNEL (
  regle 49 : distance axiale min 3.;
);

undefine CHANNEL;

# Check RDS_POLY is distant from ACTIV ZONE of TRANSISTOR
#--------------------------------------------------------
relation RDS_POLY , RDS_ACTIV (
  regle 79 : distance axiale >= 1. ;
);

relation RDS_POLY , CONT_DIFF (
  regle 50 : distance axiale >= 2. ;
);

undefine CONT_DIFF;
undefine NP_DIF;
```

Εικόνα 4.34

```
# Check RDS_ALU1 shapes
#----------------------
caracterise RDS_ALU1 (
  regle 51 : largeur >= 1. ;
  regle 52 : longueur_inter min 1. ;
  regle 53 : notch >= 3. ;
);
relation RDS_ALU1 , RDS_ALU1 (
  regle 54 : distance axiale min 3. ;
);

# Check RDS_ALU2 shapes
#----------------------
caracterise RDS_ALU2 (
  regle 55 : largeur >= 2. ;
  regle 56 : longueur_inter min 2. ;
  regle 57 : notch >= 3. ;
);
relation RDS_ALU2 , RDS_ALU2 (
  regle 58 : distance axiale min 3. ;
);

# Check RDS_ALU3 shapes
#----------------------
caracterise RDS_ALU3 (
  regle 59 : largeur >= 2. ;
  regle 60 : longueur_inter min 2. ;
  regle 61 : notch >= 3. ;
);
relation RDS_ALU3 , RDS_ALU3 (
  regle 62 : distance axiale min 3. ;
);

# Check RDS_ALU4 shapes
#----------------------
caracterise RDS_ALU4 (
  regle 63 : largeur >= 2. ;
  regle 64 : longueur_inter min 2. ;
  regle 65 : notch >= 3. ;
);
relation RDS_ALU4 , RDS_ALU4 (
  regle 66 : distance axiale min 3. ;
);

# Check RDS_ALU5 shapes
#----------------------
caracterise RDS_ALU5 (
  regle 80 : largeur >= 2. ;
  regle 81 : longueur_inter min 2. ;
  regle 82 : notch >= 3. ;
);
relation RDS_ALU5 , RDS_ALU5 (
  regle 83 : distance axiale min 3. ;
);

# Check RDS_ALU6 shapes
#----------------------
caracterise RDS_ALU6 (
  regle 84 : largeur >= 2. ;
  regle 85 : longueur_inter min 2. ;
  regle 86 : notch >= 3. ;
);
relation RDS_ALU6 , RDS_ALU6 (
  regle 87 : distance axiale min 3. ;
);
```

Εικόνα 4.35

```
# Check ANY_VIA layers, stacking are free
#----------------------------------------
relation RDS_CONT , RDS_CONT (
  regle 67 : distance axiale >= 3. ;
);
relation RDS_VIA , RDS_VIA (
  regle 68 : distance axiale >= 4. ;
);
relation RDS_VIA2 , RDS_VIA2 (
  regle 69 : distance axiale >= 4. ;
);
relation RDS_VIA3 , RDS_VIA3 (
  regle 70 : distance axiale >= 4. ;
);
relation RDS_VIA4 , RDS_VIA4 (
  regle 88 : distance axiale >= 4. ;
);
relation RDS_VIA5 , RDS_VIA5 (
  regle 89 : distance axiale >= 4. ;
);
caracterise RDS_CONT (
  regle 71 : largeur >= 1. ;
  regle 72 : longueur <= 1. ;
);
caracterise RDS_VIA (
  regle 73 : largeur >= 1. ;
  regle 74 : longueur <= 1. ;
);
caracterise RDS_VIA2 (
  regle 75 : largeur >= 1. ;
  regle 76 : longueur <= 1. ;
);
caracterise RDS_VIA3 (
  regle 77 : largeur >= 1. ;
  regle 78 : longueur <= 1. ;
);
caracterise RDS_VIA4 (
  regle 90 : largeur >= 1. ;
  regle 91 : longueur <= 1. ;
);
caracterise RDS_VIA5 (
  regle 92 : largeur >= 1. ;
  regle 93 : longueur <= 1. ;
);

# Check the POLY2 shapes
#-----------------------
caracterise RDS_POLY2 (
  regle 94 : largeur >= 1. ;
  regle 95 : longueur_inter min 1. ;
  regle 96 : notch >= 5. ;
);
relation RDS_POLY2 , RDS_POLY2 (
  regle 97 : distance axiale min 5. ;
);

# Check RDS_POLY2 is really included inside RDS_POLY1
#----------------------------------------------------
relation RDS_POLY , RDS_POLY2 (
  regle 98 : distance axiale < 0.;
  regle 99 : enveloppe inferieure min 5. ;
  regle 100 : marge longueur_inter < 0. ;
  regle 101 : croix longueur_inter < 0. ;
  regle 102 : intersection longueur_inter < 0. ;
  regle 103 : extension longueur_inter < 0. ;
  regle 104 : inclusion longueur_inter < 0. ;
);
```

Εικόνα 4.36

```
fin regles
Unify : muxoor

Create Ring : muxoor_rng
Merge Errorfiles:

Merge Error Instances:
instructionCourante :  56
End DRC on: muxoor
Saving the Error file figure
Done
  0

File: muxoor.drc is empty: no errors detected.
```

Εικόνα 4.37

Όπως διακρίνουμε από την έξοδο πραγματοποιείται λεπτομερής έλεγχος σε κάθε στάδιο του σχεδιασμού στις συνδέσεις, στις διαχύσεις, σε όλα τα layers, τα τρανζίστορ. Το τελευταίο μήνυμα της εξόδου μας λέει ότι δεν εντοπίστηκαν λάθη στο τοποθετημένο αρχείο δηλαδή δεν υπάρχουν παραβάσεις των κανόνων σχεδιασμού. Οι κανόνες σχεδιασμού καθορίζονται από το αρχείο cmos.rds το οποίο χρησιμοποιεί το εργαλείο DRUC.

Με το αρχείο μας να έχει μην έχει σφάλματα προχωράμε στο τελευταίο βήμα για την ολοκλήρωση του σχεδιασμού το οποίο είναι η μετάφραση του συμβολικού layout muxoor.ap που έχουμε δημιουργήσει σε ένα πραγματικό layout. Αυτό θα επιτευχθεί με την χρησιμοποίηση του εργαλείου S2R. Όπως το DRUC έτσι και το S2R χρησιμοποιεί το αρχείο τεχνολογίας από την μεταβλητή περιβάλλοντος RDS_TECHNO_NAME. Επιπρόσθετα πρέπει να ελέγξουμε αν η μεταβλητή RDS_OUT που δίνει την μορφή εξόδου στο αρχείο είναι σωστά ορισμένη στην κατάληξη **cif.** Τρέχουμε την εντολή env | grep RDS και όπως βλέπουμε και στην Εικόνα 4.38 είναι σωστά ορισμένη.

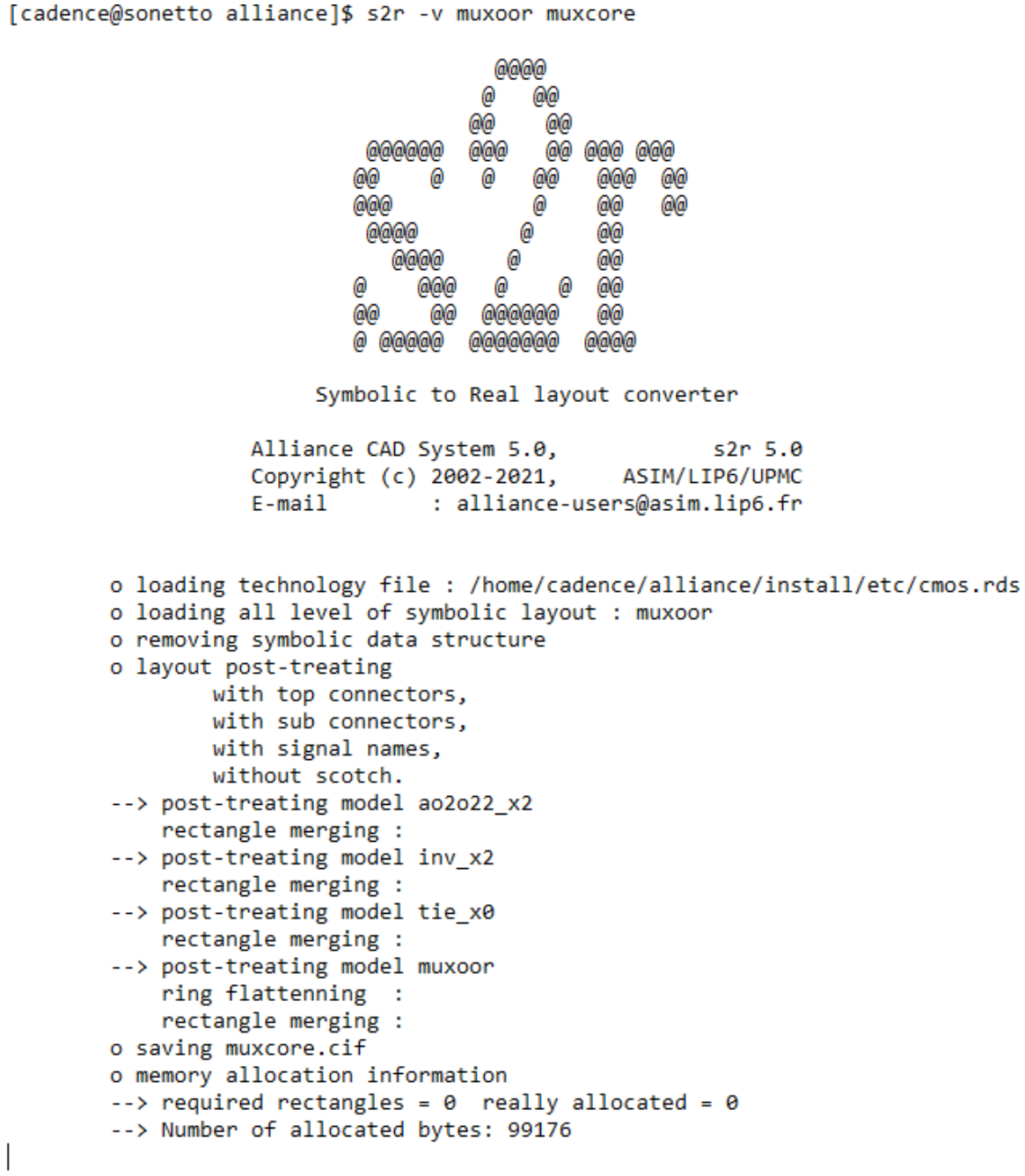
```
[cadence@sonetto alliance]$ env | grep RDS
RDS_OUT=cif
RDS_IN=cif
RDS_TECHNO_NAME=/home/cadence/alliance/install/etc/cmos.rds
```

Εικόνα 4.38

Με τις απαραίτητες μεταβλητές σωστά ορισμένες προχωράμε στην εκτέλεση του S2R γράφοντας την εντολή που ακολουθεί στην γραμμή εντολών και τα αποτελέσματα της υλοποίησης παρουσιάζονται στην Εικόνα 4.39

```
s2r -v muxoor muxcore
```

- Επιλογή **v**: Λειτουργία Verbose ενεργοποιημένη.
- Αρχείο muxcore.cif το τελικό αρχείο, το πραγματικό layout του σχεδιασμού μας.



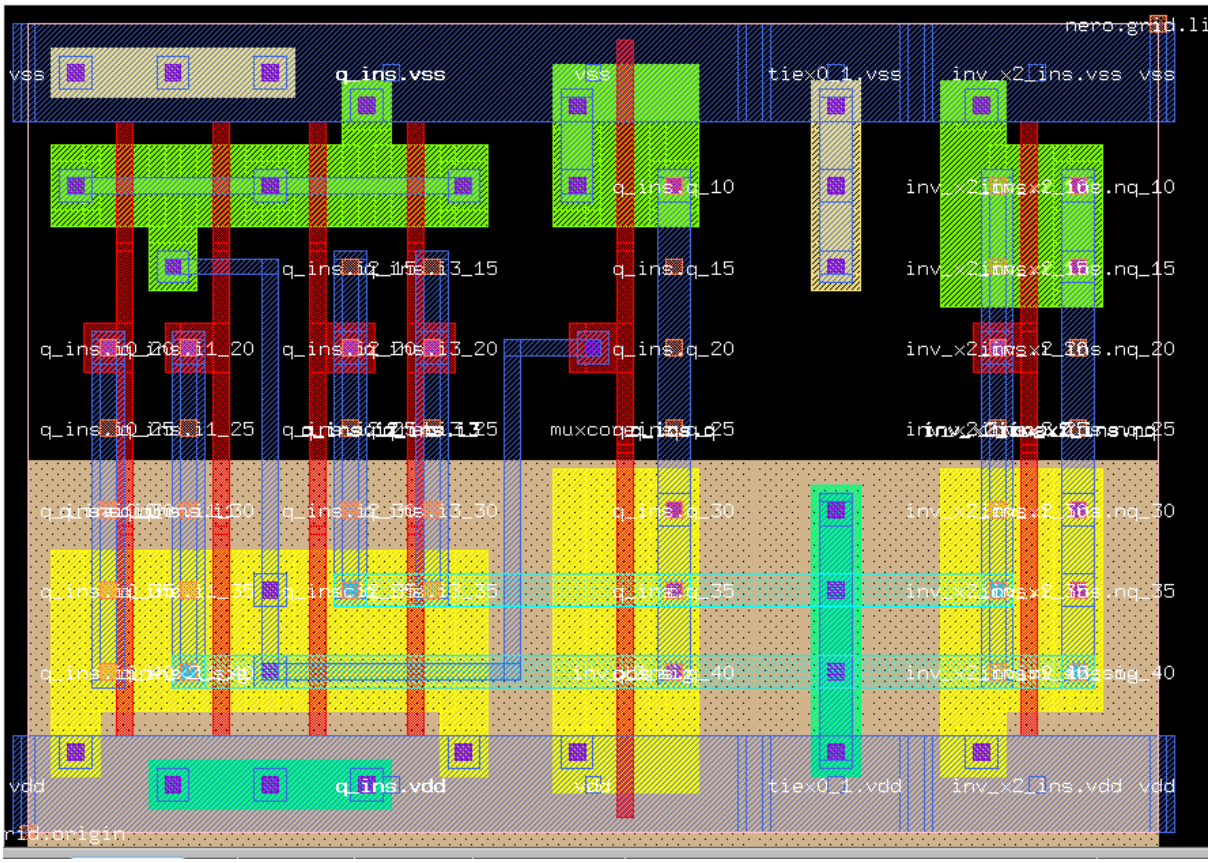
Εικόνα 4.39

Θα χρησιμοποιήσουμε το εργαλείο απεικόνισης DREAL για να δούμε το αρχείο πραγματικού layout muxcore.cif. Χρησιμοποιούμε την παρακάτω εντολή για να τρέξουμε το DREAL:

```
dreal -l muxcore
```

- Επιλογή **l**: Φορτώνει το όνομα του αρχείου που επιθυμούμε να απεικονίσει.

Στην Εικόνα 4.40 βλέπουμε το πραγματικό layout που σχεδιασμού μας του πολυπλέκτη. Από την μπάρα εργαλείων επιλέγουμε TOOLS και στην συνέχεια FLATTEN για καλύτερη απεικόνιση.



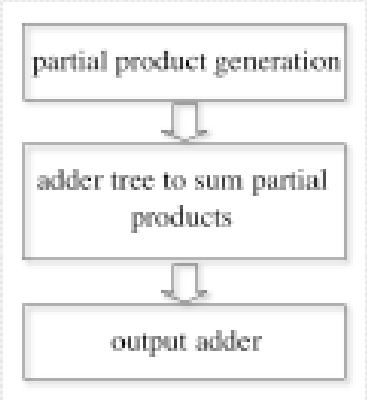
Εικόνα 4.40

5.ΥΛΟΠΟΙΗΣΗ ΠΟΛΛΑΠΛΑΣΙΑΣΤΗ

Έχοντας αναλύσει στα προηγούμενα κεφάλαια πως χρησιμοποιούμε τα εργαλεία του Alliance και έχοντας εξοικειωθεί με αυτά μέσω του παραδείγματος του πολυπλέκτη μπορούμε να προχωρήσουμε στο σχεδιασμό του πολλαπλασιαστή 4-bit που είναι και το θέμα της παρούσας πτυχιακής εργασίας. Ο σχεδιασμός του πολλαπλασιαστή θα πραγματοποιηθεί με την χρήση adder tree και πιο συγκεκριμένα με Wallace tree.

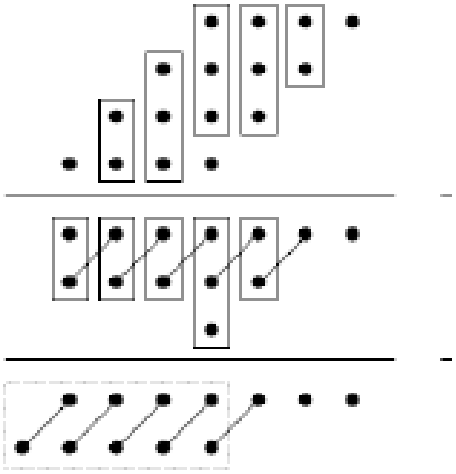
5.1 ΣΧΑΔΙΑΣΜΟΣ ΠΟΛΛΑΠΛΑΣΙΑΣΤΗ ΜΕ ΧΡΗΣΗ WALLACE TREE

Ο πολλαπλασιαστής που θα υλοποιήσουμε θα δημιουργηθεί χρησιμοποιώντας τρία μπλοκ όπως ακριβώς φαίνεται και στην Εικόνα 5.1. Τα τρία αυτά μπλοκ είναι: γεννήτρια μερικών γινομένων(partial product generation), δέντρο αθροιστών για άθροισμα μερικών γινομένων(adder tree to sum partial products) και έξοδο αθροιστών(output adder).



Εικόνα 5.1

Όπως ήδη αναφέραμε πιο πάνω για το δέντρο αθροιστών θα χρησιμοποιήσουμε τον αλγόριθμο Wallace. Ένα δέντρο Wallace είναι μια αποδοτική υλοποίηση υλικού ενός ψηφιακού κυκλώματος που πολλαπλασιάζει δύο ακέραιους αριθμούς. Στην Εικόνα 5.2 παρουσιάζεται ο τρόπος λειτουργίας του δέντρου Wallace.



Εικόνα 5.2

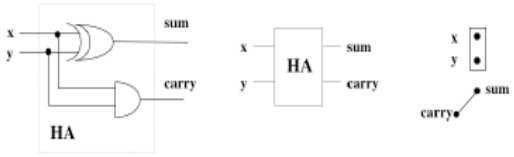
Τα ορθογώνια μπλοκ στο δέντρο αντιπροσωπεύει τα half adders και full adders.

Το Wallace δέντρο απαρτίζεται από 3 βήματα:

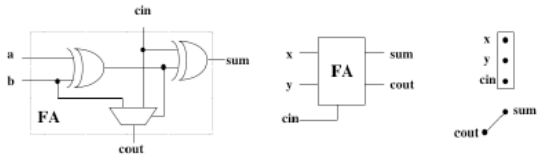
1. Πολλαπλασιάζει κάθε bit από τα ορίσματα, με κάθε ένα από τα bit του άλλου ορίσματος αντίστοιχα. Ουσιαστικά αυτό είναι και το έργο της γεννήτριας μερικών γινομένων.
2. Μειώνει τον αριθμό μερικών γινομένων με την χρήση full adders και half adders.
3. Ομαδοποιεί τα καλώδια σε ανά δύο, και τα τοποθετεί σε έναν αθροιστή.

Όπως αναφέρθηκε παραπάνω θα χρησιμοποιήσουμε half adders και full adders τα οποία αποτελούν βασικά structural cell κάθε αθροιστικού κυκλώματος και ειδικότερα ο full adder.

Στις Εικόνες 5.3,5.4 φαίνονται τα κυκλώματα ενός half adder και ενός full adder αντίστοιχα.



Εικόνα 5.3



Εικόνα 5.4

- **Half adder**: Έχει είσοδο δύο bit τα οποία προσθέτει και σαν έξοδο έχει ένα bit αθροίσματος και ένα bit κρατουμένου. Απαρτίζεται από δύο σχέσεις **S=A + B,** όπου S το άθροισμα που

έχει ίδιο βάρος με τις εισόδους A,B και **C=A*B,** όπου C το κρατούμενο εξόδου που έχει το αμέσως μεγαλύτερο βάρος.

- **Full adder**: Προσθέτει δύο δυαδικά ψηφία της ίδιας τάξης και παράγει ένα ψηφίο ίδιας αξίας S και ένα κρατούμενο εισόδου Cin της αμέσως μεγαλύτερης τάξης. Το κρατούμενο εισόδου προέρχεται από την αμέσως χαμηλότερης αξίας βαθμίδα. Ο full adder δέχεται δύο σήματα από τους δυαδικούς αριθμούς που προστίθεται και ένα κρατούμενο από τον προηγούμενο full adder, δηλαδή τρεις εισόδους. Απαρτίζεται από τις σχέσεις S= A+B+Cin και Cout= (A*B)+(A*Cin)+(B*Cin).

  Θα αναλύσουμε τώρα τα τρία βήματα για να καταλάβουμε πως λειτουργεί ο αλγόριθμος Wallace. Όπως έχουμε αναφέρει θα υλοποιήσουμε πολλαπλασιαστή 4-bit οπότε οι δύο αριθμοί μας θα αποτελούνται από 4 bit ο καθένας. Έστω ότι a_m είναι ο ένας αριθμός όπου m 0,1,2,3 τα τέσσερα bit του αριθμού και b_n ο δεύτερος αριθμός όπου n 0,1,2,3 τα τέσσερα bit του αριθμού. Το μερικό γινόμενο των a_m, b_n έχει βάρος $2^{(m+n)}$.

- Στο πρώτο βήμα πολλαπλασιάζουμε κάθε bit του ενός αριθμού με κάθε bit του άλλου αριθμού και το αποτέλεσμα θα είναι n^2bits, όπου n=4 επειδή έχουμε 4-bit πολλαπλασιαστή. Για τον αριθμό a έχουμε a_3,a_2,a_1,a_0 και για τον αριθμό b έχουμε b_3,b_2,b_1,b_0. Με φορά από τα δεξιά προς τα αριστερά πολλαπλασιάζουμε τα bit με τον τρόπου που αναφέραμε παραπάνω.

➢ Βάρος 1 :a_0b_0
➢ Βάρος 2 :a_0b_1, a_1b_0
➢ Βάρος 4: a_0b_2, a_1b_2, a_2b_0
➢ Βάρος 8: a_0b_3, a_1b_2, a_2b_1,a_3b_0
➢ Βάρος 16: a_1b_3, a_2b_2, a_3b_1
➢ Βάρος 32: a_2b_3, a_3b_2
➢ Βάρος 64: a_3b_3

- Στο δεύτερο βήμα υπάρχει η διαδικασία μείωσης. Όπου παρατηρούνται 3 γινόμενα κάθε φορά εισάγονται σε ένα full adder και τοποθετείται σε ένα νέο layer. Το κάθε bit αποτελεί ένα καλώδιο οπότε θα υπάρχουν τρία καλώδια που θα συνδεθούν στον full adder και θα έχουν το ίδιο βάρος. Το αποτέλεσμα της πρόσθεσης θα είναι ένα καλώδιο εξόδου με το ίδιο βάρος και ένα καλώδιο εξόδου με μεγαλύτερο βάρος για κάθε έναν full adder. Όπου υπάρχουν δύο bit δηλαδή δύο καλώδια του ίδιου βάρους τοποθετούνται σε ένα half adder και όταν υπάρχει μόνο ένα καλώδιο τότε περνάει στο επόμενο επίπεδο,

➢ Το βάρος 1 θα περάσει αμέσως σε επόμενο layer καθώς είναι μόνο ένα καλώδιο και θα έχει σαν έξοδο ένα καλώδιο.

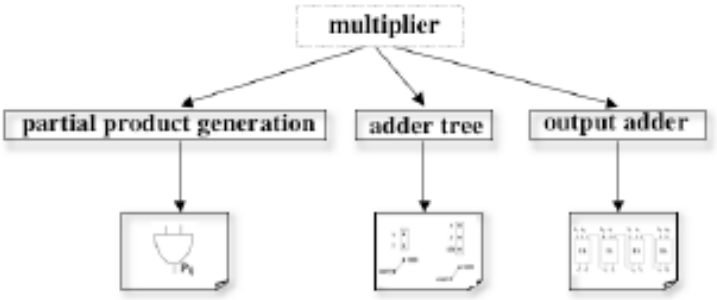
- Το βάρος 2 συνδέονται σε ένα half adder καθώς είναι δύο καλώδια και σαν έξοδο ένα καλώδιο βάρους 2 και ένα καλώδιο βάρους 4.
- Το βάρος 4 συνδέεται σε έναν full adder καθώς είναι τρία καλώδια και σαν έξοδο ένα καλώδιο βάρους 4 και ένα καλώδιο βάρους 8 που ουσιαστικά είναι το καλώδιο με το μεγαλύτερο βάρος όπως αναφέραμε και πιο πάνω.
- Το βάρος 8 συνδέεται σε ένα full adder τα τρία καλώδια και το ένα καλώδιο που απομένει περνάει στο επόμενο επίπεδο. Σαν έξοδο δύο καλώδια βάρους 8 ένα που πέρασε αμέσως και ένα που θα έχει το αποτέλεσμα της πρόσθεσης και ένα καλώδιο βάρους 16 σαν καλώδιο με μεγαλύτερο βάρος.
- Το βάρος 16 συνδέεται σε έναν full adder καθώς είναι τρία καλώδια και σαν έξοδο 1 καλώδιο βάρους 16 και ένα καλώδιο βάρους 32 σαν καλώδιο με μεγαλύτερο βάρος.
- Το βάρος 32 συνδέεται σε ένα half adder καθώς είναι δύο καλώδια και σαν έξοδο ένα καλώδιο βάρους 32 και ένα βάρους 64.
- Το βάρος 64 περνάει αμέσως στο επόμενο επίπεδο.

- Μετά το πρώτο επίπεδο μείωσης προχωράμε στο επόμενο επίπεδο μείωσης καθώς θέλουμε να έχουμε εξόδους με το πολύ δύο καλώδια.
- Το μόνο βάρος που χρειάζεται μείωση είναι το βάρος 8. Το συνδέουμε σε έναν full adder.
- Τα βάρη 4,16,32,64 τα συνδέουμε σε half adder καθώς έχουν δύο καλώδια. Για να συνδέσουμε το βάρος 64 θα χρειαστούμε και ένα ακόμα βάρος 128 για να αποτελέσει το βάρος μεγαλύτερης αξίας.

- Έχουμε φτάσει στο σημείο που δεν απαιτείται κάποια άλλη μείωση καθώς καμία έξοδος δεν έχει πάνω από δύο καλώδια.
- Βάρος 1: 1 καλώδιο.
- Βάρος 2: 1 καλώδιο.
- Βάρος 4: 1 καλώδιο.
- Βάρος 8: 2 καλώδια.
- Βάρος 16: 2 καλώδια.
- Βάρος 32: 2 καλώδια.
- Βάρος 64: 2 καλώδια.
- Βάρος 128: 1 καλώδιο.

- Έχοντας ολοκληρώσει τις μειώσεις σε όλα τα επίπεδα το μόνο που μένει είναι οι αριθμοί που προκύπτουν από τις προσθέσεις να τοποθετηθούν σε έναν αθροιστή και να μας δώσει το τελικό αποτέλεσμα. Όπως είδαμε πιο πάνω αυτός ο αθροιστής ο output adder είναι το τρίτο και τελευταίο μπλοκ που απαρτίζουν τον πολλαπλασιαστή μας.

Στην Εικόνα 5.5 που ακολουθεί παρουσιάζεται σε διάγραμμα η ιεραρχική ροή του πολλαπλασιαστή με τα τρία μπλοκ που το αποτελούν.



Εικόνα 5.5

Είμαστε έτοιμοι να προχωρήσουμε στην υλοποίηση του σχεδιασμού μας κάνοντας χρήση των εργαλείων του Alliance για την γεννήτρια μερικών γινομένων, το δέντρο αθροιστή και το αθροιστή εξόδου.[2][15][20]

5.2 ΣΧΕΔΙΑΣΜΟΣ ΓΕΝΝΗΤΡΙΑΣ ΜΕΡΙΚΩΝ ΓΙΝΟΜΕΝΩΝ

Για την υλοποίηση της γεννήτριας μερικών γινομένων θα χρησιμοποιήσουμε την περιγραφή συμπεριφοράς της Εικόνας 5.6. Αποθηκεύουμε των κώδικα με την κατάληξη **.vbe**.

```
-- 4-bit partial product generator
entity myppg4 is
port(a0,a1,a2,a3,b0,b1,b2,b3,vdd,vss : in bit;
     p00,p01,p02,p03,p10,p11,p12,p13 : out bit;
     p20,p21,p22,p23,p30,p31,p32,p33 : out bit);
end ppg4;

--behavior
architecture vbe of myppg4 is
begin
p00 <= b0 AND a0;
p01 <= b0 AND a1;
p02 <= b0 AND a2;
p03 <= b0 AND a3;
p10 <= b1 AND a0;
p11 <= b1 AND a1;
p12 <= b1 AND a2;
p13 <= b1 AND a3;
p20 <= b2 AND a0;
p21 <= b2 AND a1;
p22 <= b2 AND a2;
p23 <= b2 AND a3;
p30 <= b3 AND a0;
p31 <= b3 AND a1;
p32 <= b3 AND a2;
p33 <= b3 AND a3;
end vbe;
```

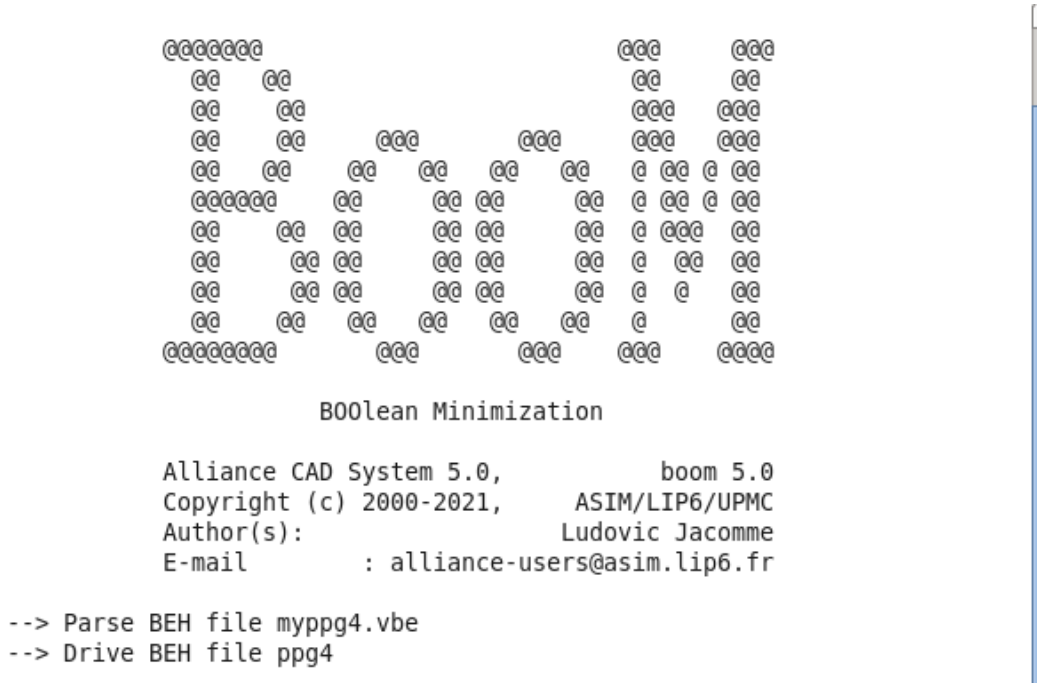
Εικόνα 5.6

Βλέπουμε ότι στα port έχουμε τα bit εισόδου, τα τέσσερα bit που ενός από τους αριθμούς και τα τέσσερα bit του άλλου αριθμού καθώς επίσης και τα bit εξόδου. Τα bit εξόδου περνάνε ένα προς ένα η πρόσθεση του κάθε bit του ενός αριθμού με το κάθε bit του άλλου αριθμού.

Έχοντας ήδη την περιγραφή συμπεριφοράς(behavioral) είμαστε σε θέση να χρησιμοποιήσουμε το εργαλείο BOOM και τα αποτελέσματα από την χρήση του φαίνονται στην Εικόνα 5.7. Με την εντολή που ακολουθεί χρησιμοποιούμε το BOOM:

```
boom -l 3 -d 0 myppg4 ppg4
```

- Επιλογή **l 3**: Καθορίζουμε το επίπεδο βελτιστοποίησης στο μέγιστο επίπεδο.
- Επιλογή **d 0**: Καθορίζουμε το την βελτιστοποίηση της καθυστέρησης.



Εικόνα 5.7

Βλέπουμε ότι έχει δημιουργηθεί το βελτιστοποιημένο αρχείο ppg4.vbe και προχωράμε στο εργαλείο που ακολουθεί πάντα στο BOOM στο στάδιο της RTL σύνθεσης το BOOG με στόχο να βελτιστοποιήσουμε την καθυστέρηση και πάλι.

Χρησιμοποιούμε το BOOG με την εντολή που ακολουθεί και τα αποτελέσματα από την χρήση του απεικονίζονται στην Εικόνα 5.8.

```
boog ppg4 myppg4 -x 0 -m 4
```

- Επιλογή **x 0:** Δημιουργεί ένα αρχείο .**xsc** το οποίο είναι ένας χρωματικός χάρτης για κάθε σχήμα που περιέχεται στο αρχείο εξόδου. Με το 0 χρωματίζουμε την κρίσιμη διαδρομή.
- Επιλογή **m 4**: Βελτιστοποίηση των καθυστερήσεων.

```
[cadence@sonetto alliance]$ boog ppg4 myppg4 -x 0 -m 4

                  @@@@@@@                          @@@@ @
                    @@   @@                       @@    @@
                    @@    @@                     @@      @
                    @@    @@     @@@       @@@    @@
                    @@   @@    @@   @@   @@   @@  @@
                    @@@@@@    @@     @@ @@     @@ @@     @@@@@
                    @@    @@  @@     @@ @@     @@ @@     @ @@
                    @@     @@ @@     @@ @@     @@ @@    @  @@
                    @@     @@ @@     @@ @@     @@  @@      @@
                    @@    @@   @@   @@   @@   @@    @@    @@
                  @@@@@@@@       @@@       @@@       @@@@

                         Binding and Optimizing On Gates

                  Alliance CAD System 5.0, boog 5.0 [2003/01/09]
                  Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
                  Author(s):                     Fran\ufffdois Donnet
                  E-mail        :    alliance-users@asim.lip6.fr

        MBK_VDD        : vdd
        MBK_VSS        : vss
        MBK_IN_LO      : vst
        MBK_OUT_LO     : vst
        MBK_WORK_LIB   : .
        MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'ppg4.vbe'...
Controlling file 'ppg4.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'ppg4.vbe'...
Capacitances on file 'ppg4.vbe'...
Unflattening file 'ppg4.vbe'...
Mapping file 'ppg4.vbe'...
Saving file 'myppg4.vst'...
Quick estimated critical path (no warranty)...324 ps from 'b3' to 'p33'
Quick estimated area (with over-cell routing)...20000 lambda\ufffd
Details...
        a2_x2: 16
        Total: 16
Saving critical path in xsch color file 'myppg4.xsc'...
End of boog...
```

Εικόνα 5.8

Διακρίνουμε ότι πραγματοποιείται βελτιστοποίηση των καθυστερήσεων, η κρίσιμη διαδρομή αποθηκεύεται σε ένα αρχείο χρωματικού χάρτη. Επίσης βλέπουμε ότι εκτελείται και χαρτογράφηση στο αρχείο που είναι και μια ακόμα ιδιότητα του BOOG και η έξοδος του εργαλείου είναι το βελτιστοποιημένο αρχείο myppg4.vst.

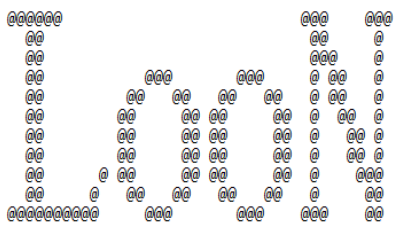
Στο αρχείο που μόλις αποκτήσαμε από το BOOG θα εκτελέσουμε το εργαλείο LOON. Το LOON θα εκτελέσει την ίδια διαδικασία με το BOOG και θα δημιουργήσει ένα νέο βελτιστοποιημένο αρχείο ppg4.vst.

Τρέχουμε το LOON στην γραμμή εντολών με την παρακάτω εντολή και στην Εικόνα 5.9 παρουσιάζονται τα αποτελέσματα από την χρήση του.

```
loon -x 0 -m 4 myppg4 ppg4
```

- Επιλογή **x 0**: Δημιουργεί ένα αρχείο .**xsc** το οποίο είναι ένας χρωματικός χάρτης για κάθε σχήμα που περιέχεται στο αρχείο εξόδου. Με το 0 χρωματίζουμε την κρίσιμη διαδρομή.
- Επιλογή **m 4**: Βελτιστοποίηση των καθυστερήσεων.

```
[cadence@sonetto alliance]$ loon -x 0 -m 4 myppg4 ppg4
```



```
                Local optimization on Nets

            Alliance CAD System 5.0, loon 5.0 [2003/12/07]
            Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
            Author(s):                     Fran\ufffdois Donnet
            E-mail        :    alliance-users@asim.lip6.fr

       MBK_IN_LO       : vst
       MBK_OUT_LO      : vst
       MBK_TARGET_LIB  : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'myppg4.vst'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Capacitances on file 'myppg4.vst'...
Delays on file 'myppg4.vst'...324 ps
Area on file 'myppg4.vst'...20000 lamda\ufffd (with over-cell routing)
Details...
        a2_x2: 16 (100%)
        Total: 16
Worst RC on file 'myppg4.vst'...0 ps
Inserting buffers on critical path for file 'ppg4.vst'...None inserted
Improving RC on critical path for file 'ppg4.vst'...324 ps
Improving all RC for file 'ppg4.vst'...
Worst RC on file 'ppg4.vst'...0 ps
Area on file 'ppg4.vst'...20000 lamda\ufffd (with over-cell routing)
Details...
        a2_x2: 16 (100%)
        Total: 16
Critical path (no warranty)...324 ps from 'b3' to 'p33'
Saving file 'ppg4.vst'...
Saving critical path in xsch color file 'ppg4.xsc'...
End of loon...
```

Εικόνα 5.9

Και εδώ όπως και με το BOOG διακρίνουμε βελτιστοποίηση καθυστέρησης και αποθήκευση της κρίσιμης διαδρομής σε αρχείο χρωματικού χάρτη. Ακόμα βλέπουμε ότι βελτιώνεται το RC στην κρίσιμη διαδρομή. Το μοντέλο καθυστέρησης RC είναι μια μέτρηση που χρησιμοποιείται στο σχεδιασμό VLSI για τον υπολογισμό της καθυστέρησης σήματος μεταξύ της τάσης εισόδου και της τάσης εξόδου του σήματος εισόδου. Σε αυτή την περίπτωση τον τρανζίστορ μπορεί να θεωρηθεί ως διακόπτης μεταξύ σε σειρά με αντίσταση.

5.3 ΔΕΝΤΡΟ ΑΘΡΟΙΣΤΩΝ

Έχουμε ολοκληρώσει το πρώτο στάδιο του σχεδιασμού μας που είναι το structural αρχείο ppg4.vst της γεννήτριας μερικών γινομένων και μπορούμε να προχωρήσουμε στην υλοποίηση του δεύτερου μπλοκ του δέντρου αθροιστών.

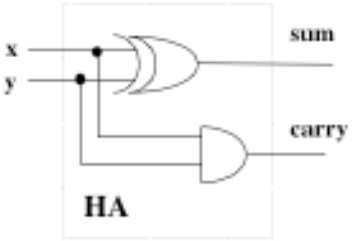
Με βάση όσα είπαμε και σε προηγούμενο κεφάλαιο για το δέντρο αθροιστών θα χρησιμοποιήσουμε half adders και full adders. Θα ξεκινήσουμε την σχεδίαση και την σύνθεση των αθροιστών αυτών και στην συνέχεια θα υλοποιήσουμε το δέντρο αυτό κάνοντας χρήση του αλγορίθμου Wallace.

Η περιγραφή συμπεριφοράς του half adder παρουσιάζεται στην Εικόνα 5.10 και θα το αποθηκεύσουμε σε μορφή **myha.vbe**.

```
-- half adder
entity ha is
port(a,b,vdd,vss : in bit;
     sum,carry   : out bit);
end ha;

--behavior
architecture vbe of ha is
signal a_bar, b_bar : bit;
begin
a_bar <= NOT a;
b_bar <= NOT b;
sum <= (a_bar AND b) OR (a AND b_bar);
carry <= a AND b;
end vbe;
```

Εικόνα 5.10

Κοιτώντας των κώδικα αλλά και την Εικόνα 5.11 που ακολουθεί παρατηρούμε ότι η περιγραφή συμπεριφοράς και το κύκλωμα της εικόνας δεν ταιριάζουν.



Εικόνα 5.11

Χρησιμοποιώντας τον εργαλείο BOOM θα βελτιστοποιήσουμε την καθυστέρηση αλλά ανοίγοντας το αρχείο ha.vbe όπως φαίνεται στην Εικόνα 5.13 παρατηρούμε ότι η πύλη OR έχει αντικατασταθεί με την πύλη XOR που ταιριάζει με το κύκλωμα της Εικόνας 5.11.

Χρησιμοποιούμε το BOOM με την εντολή:

```
boom -l 3 -d 0 myha ha
```

- Επιλογή **l 3**: Καθορίζουμε το επίπεδο βελτιστοποίησης στο μέγιστο επίπεδο.
- Επιλογή **d 0**: Καθορίζουμε το την βελτιστοποίηση της καθυστέρησης.

Στην Εικόνα 5.12 βλέπουμε ότι το εργαλείο έτρεξε σωστά και δημιούργησε το βελτιστοποιημένο αρχείο **ha.vbe.**

```
[cadence@sonetto alliance]$ boom -l 3 -d 0 myha ha
```

```
@@@@@@@@                                @@@      @@@
  @@    @@                                @@      @@
  @@     @@                               @@@    @@@
  @@     @@      @@@        @@@           @@@    @@@
  @@    @@     @@   @@    @@   @@    @ @@ @ @@
  @@@@@@@     @@      @@ @@      @@   @ @@ @ @@
  @@     @@   @@      @@ @@      @@   @ @@@   @@
  @@      @@ @@      @@ @@      @@   @   @@   @@
  @@      @@ @@      @@ @@      @@   @   @    @@
  @@     @@    @@    @@    @@    @@   @        @@
@@@@@@@@        @@@        @@@     @@@     @@@@

                BOOlean Minimization

Alliance CAD System 5.0,                boom 5.0
Copyright (c) 2000-2021,      ASIM/LIP6/UPMC
Author(s):                   Ludovic Jacomme
E-mail         : alliance-users@asim.lip6.fr

--> Parse BEH file myha.vbe
--> Drive BEH file ha
```

Εικόνα 5.12

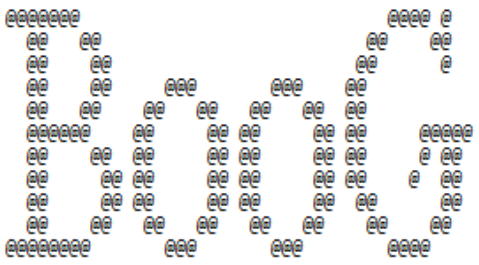
```vhdl
-- VHDL data flow description generated from `ha`
--              date : Wed Jan 27 18:02:46 2021


-- Entity Declaration

ENTITY ha IS
  PORT (
  a : in BIT;   -- a
  b : in BIT;   -- b
  vdd : in BIT; -- vdd
  vss : in BIT; -- vss
  sum : out BIT;        -- sum
  carry : out BIT       -- carry
  );
END ha;

-- Architecture Declaration

ARCHITECTURE behaviour_data_flow OF ha IS

BEGIN

carry <= (a AND b);

sum <= (a XOR b);
END;
```

Εικόνα 5.13

Αποκτώντας το αρχείο ha.vbe από το BOOM θα χρησιμοποιήσουμε το BOOG, LOON με στόχο να βελτιστοποιήσουμε την καθυστέρηση, να χρωματίσουμε την κρίσιμη περιοχή.

Χρησιμοποιούμε το BOOG με την εντολή που ακολουθεί και τα αποτελέσματα φαίνονται στην Εικόνα 5.14. Χρησιμοποιούμε ακριβώς τις ίδιες επιλογές για το εργαλείο όπως και στο προηγούμενο κεφάλαιο με την γεννήτρια μερικών γινομένων.

```
[cadence@sonetto alliance]$ boog ha myha -x 0 -m 4
```



```
       Binding and Optimizing On Gates

Alliance CAD System 5.0, boog 5.0 [2003/01/09]
Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
Author(s):                     Fran\ufffdois Donnet
E-mail        :    alliance-users@asim.lip6.fr

        MBK_VDD        : vdd
        MBK_VSS        : vss
        MBK_IN_LO      : vst
        MBK_OUT_LO     : vst
        MBK_WORK_LIB   : .
        MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'ha.vbe'...
Controlling file 'ha.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'ha.vbe'...
Capacitances on file 'ha.vbe'...
Unflattening file 'ha.vbe'...
Mapping file 'ha.vbe'...
Saving file 'myha.vst'...
Quick estimated critical path (no warranty)...357 ps from 'a' to 'sum'
Quick estimated area (with over-cell routing)...3500 lambda\ufffd
Details...
        xr2_x1: 1
        a2_x2: 1
        Total: 2
Saving critical path in xsch color file 'myha.xsc'...
End of boog...
```

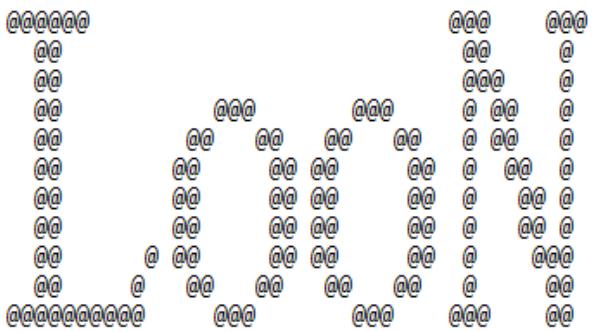
Εικόνα 5.14

Ακριβώς όπως κάναμε και στο παραπάνω κεφάλαιο χρησιμοποιούμε το LOON και εδώ με τις ίδιες ακριβώς επιλογές και τα αποτελέσματά του εργαλείου παρουσιάζονται στην Εικόνα 5.15.

Χρησιμοποιούμε το LOON στην γραμμή εντολών του τερματικού ως εξής:

```
loon -x 0 -m 4 myha ha
```

```
[cadence@sonetto alliance]$ loon -x 0 -m 4 myha ha
```



```
                Local optimization on Nets

             Alliance CAD System 5.0, loon 5.0 [2003/12/07]
             Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
             Author(s):                     Fran\ufffdois Donnet
             E-mail        :    alliance-users@asim.lip6.fr

      MBK_IN_LO       : vst
      MBK_OUT_LO      : vst
      MBK_TARGET_LIB  : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'myha.vst'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Capacitances on file 'myha.vst'...
Delays on file 'myha.vst'...357 ps
Area on file 'myha.vst'...3500 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 1 (64%)
        a2_x2: 1 (35%)
        Total: 2
Worst RC on file 'myha.vst'...0 ps
Inserting buffers on critical path for file 'ha.vst'...None inserted
Improving RC on critical path for file 'ha.vst'...357 ps
Improving all RC for file 'ha.vst'...
Worst RC on file 'ha.vst'...0 ps
Area on file 'ha.vst'...3500 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 1 (64%)
        a2_x2: 1 (35%)
        Total: 2
Critical path (no warranty)...357 ps from 'a' to 'sum'
Saving file 'ha.vst'...
Saving critical path in xsch color file 'ha.xsc'...
End of loon...
```

Εικόνα 5.15

Με την εκτέλεση και του εργαλείου LOON και αποκτώντας το βελτιστοποιημένο αρχείο **ha.vst** έχουμε τελειώσει με το ένα φύλλο του δέντρου αθροιστών και μας μένει το άλλο φύλλο full adder.

Και σε αυτή την περίπτωση θα αποθηκεύσουμε την περιγραφή συμπεριφοράς σε ένα αρχείο **myfa.vbe** και ο κώδικας του full adder φαίνεται στην Εικόνα 5.16.

```
-- full adder
entity fa is
port(a,b,cin,vdd,vss : in bit;
     sum,cout        : out bit);
end fa;

--behavior
architecture vbe of fa is
signal xor1 : bit;

begin
xor1 <= a XOR b;
cout <= b when xor1 = '0' else cin;
sum <= cin XOR xor1;
end vbe;
```

Εικόνα 5.16

Θα χρησιμοποιήσουμε ακριβώς την ίδια μεθοδολογία όπως κάναμε και για τον half adder με ακριβώς τα ίδια εργαλεία BOOM, BOOG, LOON και ακριβώς τις ίδιες επιλογές. Η μόνη διαφορά θα είναι το αρχείο που χρησιμοποιούμε το οποίο θα είναι το myfa.vbe για το BOOM και από εκεί και πέρα για τα άλλα δύο εργαλεία θα χρησιμοποιούμε το παραγόμενο βελτιστοποιημένο αρχείο. Για αυτό τον λόγο θα παρουσιαστούν οι εντολές για κάθε εργαλείο καθώς και τα αποτελέσματά τους μόνο.

Αρχικά χρησιμοποιούμε το BOOM με την παρακάτω εντολή και το αποτέλεσμα απεικονίζεται στην Εικόνα 5.17.

```
boom -l 3 -d 0 myfa fa
```

```
[cadence@sonetto alliance]$ boom -l 3 -d 0 myfa fa
                  @@@@@@@                        @@@     @@@
                    @@   @@                       @@     @@
                    @@    @@                      @@@   @@@
                    @@    @@     @@@       @@@    @@@   @@@
                    @@   @@    @@   @@   @@   @@   @ @@ @ @@
                    @@@@@@    @@     @@ @@     @@  @ @@ @ @@
                    @@    @@  @@     @@ @@     @@  @ @@@  @@
                    @@     @@ @@     @@ @@     @@  @  @@  @@
                    @@     @@ @@     @@ @@     @@  @  @   @@
                    @@    @@   @@   @@   @@   @@   @      @@
                  @@@@@@@@@       @@@       @@@    @@@    @@@@

                            BOOlean Minimization

                  Alliance CAD System 5.0,           boom 5.0
                  Copyright (c) 2000-2021,     ASIM/LIP6/UPMC
                  Author(s):                  Ludovic Jacomme
                  E-mail        : alliance-users@asim.lip6.fr

        --> Parse BEH file myfa.vbe
        --> Drive BEH file fa
```

Εικόνα 5.17

Στην συνέχεια χρησιμοποιούμε το BOOG που πάντα ακολουθεί του BOOM. Χρησιμοποιείται με την εντολή που ακολουθεί και το αποτέλεσμα παρουσιάζεται στην Εικόνα 5.18.

```
[cadence@sonetto alliance]$ boog fa myfa -x 0 -m 4

                 @@@@@@@                          @@@@ @
                   @@   @@                       @@    @@
                   @@    @@                     @@      @
                   @@    @@     @@@       @@@    @@
                   @@   @@    @@   @@   @@   @@  @@
                   @@@@@@    @@     @@ @@     @@ @@     @@@@@
                   @@    @@  @@     @@ @@     @@ @@     @ @@
                   @@     @@ @@     @@ @@     @@ @@    @  @@
                   @@     @@ @@     @@ @@     @@  @@      @@
                   @@    @@   @@   @@   @@   @@    @@    @@
                  @@@@@@@@       @@@       @@@        @@@@

                        Binding and Optimizing On Gates

                 Alliance CAD System 5.0, boog 5.0 [2003/01/09]
                 Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
                 Author(s):                     Fran\ufffdois Donnet
                 E-mail        :    alliance-users@asim.lip6.fr

        MBK_VDD        : vdd
        MBK_VSS        : vss
        MBK_IN_LO      : vst
        MBK_OUT_LO     : vst
        MBK_WORK_LIB   : .
        MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'fa.vbe'...
Controlling file 'fa.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'fa.vbe'...
Capacitances on file 'fa.vbe'...
Unflattening file 'fa.vbe'...
Mapping file 'fa.vbe'...
Saving file 'myfa.vst'...
Quick estimated critical path (no warranty)...756 ps from 'a' to 'sum'
Quick estimated area (with over-cell routing)...7000 lambda\ufffd
Details...
        xr2_x1: 2
        oa2ao222_x2: 1
        Total: 3
Saving critical path in xsch color file 'myfa.xsc'...
End of boog...
```

Εικόνα 5.18

Και τέλος ακολουθεί το LOON που το χρησιμοποιούμε με την εντολή που φαίνεται παρακάτω και στην Εικόνα 5.19 βλέπουμε τα αποτελέσματά του.

```
loon -x 0 -m 4 myfa fa
```

```
[cadence@sonetto alliance]$ loon -x 0 -m 4 myfa fa

                  @@@@@@                          @@@    @@@
                    @@                            @@     @
                    @@                            @@@    @
                    @@          @@@       @@@     @ @@   @
                    @@        @@   @@   @@   @@   @ @@   @
                    @@       @@     @@ @@     @@  @  @@  @
                    @@       @@     @@ @@     @@  @   @@ @
                    @@       @@     @@ @@     @@  @   @@ @
                    @@     @ @@     @@ @@     @@  @    @@@
                    @@    @   @@   @@   @@   @@   @     @@
                  @@@@@@@@@@     @@@       @@@    @@@    @@

                          Local optimization on Nets

                    Alliance CAD System 5.0, loon 5.0 [2003/12/07]
                    Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
                    Author(s):                     Fran\ufffdois Donnet
                    E-mail        :    alliance-users@asim.lip6.fr

            MBK_IN_LO       : vst
            MBK_OUT_LO      : vst
            MBK_TARGET_LIB  : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'myfa.vst'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Capacitances on file 'myfa.vst'...
Delays on file 'myfa.vst'...756 ps
Area on file 'myfa.vst'...7000 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 2 (64%)
        oa2ao222_x2: 1 (35%)
        Total: 3
Worst RC on file 'myfa.vst'...63 ps
Inserting buffers on critical path for file 'fa.vst'...None inserted
Improving RC on critical path for file 'fa.vst'...756 ps
Improving all RC for file 'fa.vst'...
Worst RC on file 'fa.vst'...63 ps
Area on file 'fa.vst'...7000 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 2 (64%)
        oa2ao222_x2: 1 (35%)
        Total: 3
Critical path (no warranty)...756 ps from 'a' to 'sum'
Saving file 'fa.vst'...
Saving critical path in xsch color file 'fa.xsc'...
End of loon...
```

Εικόνα 5.19

Έχοντας αποκτήσει και τα δύο απαραίτητα αρχεία fa.vst, ha.vst για την σύνθεση του δέντρου αθροιστών είμαστε σε θέση να προχωρήσουμε στην υλοποίηση του δέντρου.

Στο προηγούμενο κεφάλαιο εξηγήσαμε πως λειτουργεί ο αλγόριθμος Wallace σε θεωρητικό στάδιο, τώρα θα δούμε τον αλγόριθμο χρησιμοποιώντας την περιγραφή συμπεριφοράς του στην πράξη. Όπως θα δούμε στην Εικόνα 5.20 ο κώδικας του αλγορίθμου είναι μια structural περιγραφή, όμως για να μπορέσουμε να χρησιμοποιήσουμε τα εργαλεία σύνθεσης BOOM,BOOG και LOON θα πρέπει να τον μετατρέψουμε σε περιγραφή συμπεριφοράς δηλαδή σε μορφή **..vbe**. Αρχικά θα αποθηκεύσουμε τον κώδικα με όνομα mywallace.vst.

```
 entity wallace4 is
 port(p01,p02,p03 : in bit;
      p10,p11,p12,p13 : in bit;
      p20,p21,p22,p23 : in bit;
      p30,p31,p32,p33 : in bit;
      vdd,vss : in bit;
      p1,p2,p3a,p3b,p4a,p4b,p5a,p5b,p6a,p6b,p7b : out bit);
end wallace4;

architecture structural of wallace4 is
Component ha
 port(a,b : in bit;
      vdd,vss : in bit;
      sum,carry : out bit);
end component;

component fa
 port(a,b,cin : in bit;
      vdd,vss : in bit;
      sum,cout : out bit);
end component;

signal h1c,f1s,f1c,f2s,f2c : bit;
signal f3s,f3c,h2s,h2c : bit;
```

```
begin
-- Wallace tree
-- first level

ha1 : ha
 port map(a => p01, b  => p10,
          vdd => vdd, vss => vss,
          sum => p1, carry => h1c);

fa1 : fa
 port map(a => p02, b => p11, cin => p20,
          vdd => vdd, vss => vss,
          sum => f1s, cout => f1c);

fa2 : fa
 port map(a => p03, b => p12, cin => p21,
          vdd => vdd, vss => vss,
          sum => f2s, cout => f2c);

fa3 : fa
 port map(a => p13, b => p22, cin => p31,
          vdd => vdd, vss => vss,
          sum => f3s, cout => f3c);

ha2 : ha
 port map(a => p23, b  => p32,
          vdd => vdd, vss => vss,
          sum => h2s, carry => h2c);

-- second level
ha3 : ha
 port map(a => f1s, b  => h1c,
          vdd => vdd, vss => vss,
          sum => p2, carry => p3b);

fa4 : fa
 port map(a => f2s, b => f1c, cin => p30,
          vdd => vdd, vss => vss,
          sum => p3a, cout => p4b);

ha4 : ha
 port map(a => f3s, b  => f2c,
          vdd => vdd, vss => vss,
          sum => p4a, carry => p5b);

ha5 : ha
 port map(a => h2s, b  => f3c,
          vdd => vdd, vss => vss,
          sum => p5a, carry => p6b);

ha6 : ha
 port map(a => p33, b  => h2c,
          vdd => vdd, vss => vss,
          sum => p6a, carry => p7b);

end structural;
```

Εικόνα 5.20

Κοιτάζοντας τον κώδικα βλέπουμε ότι θέτονται τα bit εισόδου καθώς και τα bit εξόδου για τον αλγόριθμο Wallace. Στην συνέχεια χρησιμοποιούμε τα component τα οποία είναι τα φύλλα που είχαμε συνθέσει προηγουμένως full adder και half adder. Πιο κάτω ξεκινάει η ανάλυση του αλγορίθμου και πιο συγκεκριμένα το πρώτο layer μείωσης ακριβώς όπως είδαμε και στην θεωρητική εξήγηση του αλγορίθμου Wallace. Υπάρχει αντιστοιχία στα bit εισόδου και εξόδου του αλγορίθμου με τα αντίστοιχα bit των full adder και half adder και στα δύο layer μείωσης.

Για να μετατρέψουμε τώρα την structural περιγραφή σε behavioral περιγραφή θα χρησιμοποιήσουμε το εργαλείο του Alliance Flatbeh. Χρησιμοποιούμε το εργαλείο με την εντολή που ακολουθεί και η έξοδος του παρουσιάζεται στην Εικόνα 5.21.

```
flatbeh mywallace mywallace
```

- Το αρχείο mywallace είναι το αρχείο εισόδου με κατάληξη .vst που είναι το αρχικό αρχείο structural και το δεύτερο mywallace είναι η περιγραφή behavioral σε μορφή .vbe που είναι το αρχείο εξόδου.

```
[cadence@sonetto alliance]$ flatbeh mywallace mywallace

      @@@@@@@@@@  @@@@                    @@@@@@@                 @@@
        @@     @    @@                   @       @@   @@              @@
        @@      @   @@                 @@        @@    @@             @@
        @@          @@      @@@@       @@        @@    @@     @@@@@    @@ @@@
        @@    @     @@     @@   @    @@@@@@@@    @@    @@     @      @   @@@  @@
        @@@@@@      @@     @@   @@     @@        @@@@@@     @@      @@  @@    @@
        @@    @     @@       @@@@@     @@        @@     @@  @@@@@@@@@@  @@    @@
        @@          @@      @@    @@   @@        @@       @@ @@         @@    @@
        @@          @@     @@     @@   @@        @@      @@ @@        @  @@    @@
        @@          @@    @@      @@   @@        @@     @@   @@      @@  @@    @@
      @@@@@@      @@@@@@  @@@@  @@    @@@@  @@@@@@@@@       @@@@   @@@@  @@@@

                          a netlist abstractor

                Alliance CAD System 5.0, flatbeh 5.0 [2000/11/01]
                Copyright (c) 1993-2021,           ASIM/LIP6/UPMC
                Author(s):       Fran\ufffdois DONNET, Huu Nghia VUONG
                E-mail        :       alliance-users@asim.lip6.fr

      ========================  Environnement  ========================
      MBK_WORK_LIB        = .
      MBK_CATA_LIB        = .:/home/cadence/alliance/install/cells/sxlib:/home/cadence/alliance/install/cells/dp_sxlib:/home/cadence/allianc
      MBK_CATAL_NAME      = CATAL
      ============================== Files ==============================
      Netlist file        = mywallace.vst
      Output  file        = mywallace.vbe
      ===================================================================

Loading './mywallace.vst'
flattening figure mywallace
 loading oa2ao222_x2
 loading a2_x2
 loading xr2_x1
Restoring array's orders
BEH : Saving 'mywallace' in a vhdl file (vbe)
```

Εικόνα 5.21

Βλέπουμε ότι το Flatbeh δημιούργησε το αρχείο mywallace.vbe και είμαστε σε θέση να προχωρήσουμε στην διαδικασία σύνθεσης με τα εργαλεία που χρησιμοποιήσαμε και σε προηγούμενα βήματα. Και εδώ όπως και πριν για τα εργαλεία BOOM,BOOG,LOON θα

χρησιμοποιήσουμε ακριβώς τις ίδιες επιλογές καθώς θέλουμε να πετύχουμε τα ίδια αποτελέσματα, με την μόνη διαφορά να είναι το αρχείο που χρησιμοποιούμε σε αυτό το βήμα που αρχικά είναι το mywallace.vbe. Για τον λόγο αυτό θα παρουσιάσουμε μόνον τις εντολές που θα χρησιμοποιήσουμε καθώς και τις εξόδους αυτών.

Αρχικά θα χρησιμοποιήσουμε το εργαλείο BOOM του οποίου η εντολή φαίνεται στην γραμμή που ακολουθεί και το αποτέλεσμά του παρουσιάζεται στην Εικόνα 5.22.

```
boom -l 3 -d 0 mywallace wallace
```

```
[cadence@sonetto alliance]$ boom -l 3 -d 0 mywallace wallace
                  @@@@@@@                          @@@     @@@
                    @@   @@                         @@      @@
                    @@    @@                        @@@   @@@
                    @@    @@      @@@       @@@     @@@   @@@
                    @@   @@    @@   @@   @@   @@   @ @@ @ @@
                    @@@@@@    @@     @@ @@     @@  @ @@ @ @@
                    @@    @@  @@     @@ @@     @@  @ @@@  @@
                    @@     @@ @@     @@ @@     @@  @  @@  @@
                    @@     @@ @@     @@ @@     @@  @  @   @@
                    @@    @@   @@   @@   @@   @@   @       @@
                  @@@@@@@@       @@@       @@@    @@@    @@@@

                            BOOlean Minimization

                  Alliance CAD System 5.0,           boom 5.0
                  Copyright (c) 2000-2021,     ASIM/LIP6/UPMC
                  Author(s):                  Ludovic Jacomme
                  E-mail        : alliance-users@asim.lip6.fr

        --> Parse BEH file mywallace.vbe
        --> Drive BEH file wallace
```

Εικόνα 5.22

Έχοντας αποκτήσει το βελτιστοποιημένο αρχείο Wallace.vbe προχωράμε με εργαλείο που πάντα ακολουθεί το BOOM, το εργαλείο BOOG. Το τρέχουμε στην γραμμή εντολών με την εξής εντολή και η έξοδος του απεικονίζεται στην Εικόνα 5.23:

```
[cadence@sonetto alliance]$ boog wallace swallace -x 0 -m 4

                  @@@@@@@                          @@@@ @
                    @@   @@                       @@    @@
                    @@    @@                      @@     @
                    @@    @@      @@@      @@@    @@
                    @@   @@    @@   @@   @@   @@  @@
                    @@@@@@    @@     @@ @@     @@ @@     @@@@@
                    @@    @@  @@     @@ @@     @@ @@     @ @@
                    @@     @@ @@     @@ @@     @@ @@    @  @@
                    @@     @@ @@     @@ @@     @@  @@      @@
                    @@    @@   @@   @@   @@   @@    @@    @@
                  @@@@@@@@      @@@      @@@       @@@@

                         Binding and Optimizing On Gates

                  Alliance CAD System 5.0, boog 5.0 [2003/01/09]
                  Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
                  Author(s):                     Fran\ufffdois Donnet
                  E-mail        :    alliance-users@asim.lip6.fr

        MBK_VDD        : vdd
        MBK_VSS        : vss
        MBK_IN_LO      : vst
        MBK_OUT_LO     : vst
        MBK_WORK_LIB   : .
        MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'wallace.vbe'...
Controlling file 'wallace.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'wallace.vbe'...
Capacitances on file 'wallace.vbe'...
Unflattening file 'wallace.vbe'...
Mapping file 'wallace.vbe'...
Saving file 'swallace.vst'...
Quick estimated critical path (no warranty)...1737 ps from 'p21' to 'p3a'
Quick estimated area (with over-cell routing)...93500 lambda\ufffd
Details...
        xr2_x1: 21
        inv_x2: 6
        nxr2_x1: 4
        a2_x2: 4
        na2_x1: 3
        noa22_x1: 3
        mx2_x2: 3
        no2_x1: 2
        o2_x2: 2
        an12_x1: 1
        no3_x1: 1
        nao22_x1: 1
        noa2a2a23_x1: 1
        oa2ao222_x2: 1
        Total: 53
Saving critical path in xsch color file 'swallace.xsc'...
End of boog...
```

Εικόνα 5.23

Και τέλος ακολουθεί το LOON που το χρησιμοποιούμε με την εντολή που φαίνεται παρακάτω και στην Εικόνα 5.24 βλέπουμε τα αποτελέσματά του.

```
[cadence@sonetto alliance]$ loon -x 0 -m 4 swallace wallace
```

```
                Local optimization on Nets

              Alliance CAD System 5.0, loon 5.0 [2003/12/07]
              Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
              Author(s):                    Fran\ufffdois Donnet
              E-mail        :    alliance-users@asim.lip6.fr

       MBK_IN_LO       : vst
       MBK_OUT_LO      : vst
       MBK_TARGET_LIB  : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'swallace.vst'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Capacitances on file 'swallace.vst'...
Delays on file 'swallace.vst'...1808 ps
Area on file 'swallace.vst'...93500 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 21 (50%)
        inv_x2: 6 (4%)
        nxr2_x1: 4 (9%)
        a2_x2: 4 (5%)
        na2_x1: 3 (3%)
        noa22_x1: 3 (4%)
        mx2_x2: 3 (7%)
        no2_x1: 2 (2%)
        o2_x2: 2 (2%)
        an12_x1: 1 (1%)
        no3_x1: 1 (1%)
        nao22_x1: 1 (1%)
        noa2a2a23_x1: 1 (2%)
        oa2ao222_x2: 1 (2%)
        Total: 53
Worst RC on file 'swallace.vst'...127 ps
Inserting buffers on critical path for file 'wallace.vst'...None inserted
Improving RC on critical path for file 'wallace.vst'...1759 ps
Improving all RC for file 'wallace.vst'...
Worst RC on file 'wallace.vst'...127 ps
Area on file 'wallace.vst'...93500 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 21 (50%)
        inv_x2: 5 (4%)
        nxr2_x1: 4 (9%)
        a2_x2: 4 (5%)
        na2_x1: 3 (3%)
        noa22_x1: 3 (4%)
        mx2_x2: 3 (7%)
        no2_x1: 2 (2%)
        o2_x2: 2 (2%)
        an12_x1: 1 (1%)
        no3_x1: 1 (1%)
        nao22_x1: 1 (1%)
        noa2a2a23_x1: 1 (2%)
        inv_x1: 1 (0%)
        oa2ao222_x2: 1 (2%)
        Total: 53
Critical path (no warranty)...1759 ps from 'p03' to 'p4b'
Saving file 'wallace.vst'...
Saving critical path in xsch color file 'wallace.xsc'...
End of loon...
```

Εικόνα 5.24

Με την ολοκλήρωση και του LOON και την απόκτηση του αρχείου wallace.vst έχει ολοκληρωθεί και η διαδικασία σύνθεσης για το δεύτερο μπλοκ το δέντρο αθροιστών που αποτελεί τον σχεδιασμό μας.

5.4 ΑΘΡΟΙΣΤΗΣ ΕΞΟΔΟΥ

Για τον σχεδιασμό του αθροιστή εξόδου θα χρησιμοποιήσουμε πάλι την ίδια μεθοδολογία. Έχουμε μια structural περιγραφή την οποία αποθηκεύουμε σε ένα αρχείο add5.vst. Στην Εικόνα 5.25 φαίνεται ο κώδικας που θα χρησιμοποιήσουμε για τον αθροιστή εξόδου. Και σε αυτή την περίπτωση αρχικά δηλώνονται τα bit εισόδου και εξόδου, καθώς και τα component full adder, half adder γιατί και σε αυτό το μπλοκ τα χρειαζόμαστε για να υλοποιήσουμε τις τελικές προσθέσεις των bit ώστε να παραχθεί το τελικό μας αποτέλεσμα ο πολλαπλασιασμός.

```
entity add5 is
 port(a3,a4,a5,a6 : in bit;
      b3,b4,b5,b6,b7 : in bit;
      vdd,vss : in bit;
      s3,s4,s5,s6,s7,c7 : out bit);
end add5;

architecture structural of add5 is
Component ha
 port(a,b : in bit;
      vdd,vss : in bit;
      sum,carry : out bit);
end component;

component fa
 port(a,b,cin : in bit;
      vdd,vss : in bit;
      sum,cout : out bit);
end component;

signal c3,c4,c5,c6 : bit;

begin
-- carry ripple adder for the 4-bit multiplier
ha1 : ha
 port map(a => a3, b  => b3,
          vdd => vdd, vss => vss,
          sum => s3, carry => c3);

fa1 : fa
 port map(a => a4, b => b4, cin => c3,
          vdd => vdd, vss => vss,
          sum => s4, cout => c4);

fa2 : fa
 port map(a => a5, b => b5, cin => c4,
          vdd => vdd, vss => vss,
          sum => s5, cout => c5);

fa3 : fa
 port map(a => a6, b => b6, cin => c5,
          vdd => vdd, vss => vss,
          sum => s6, cout => c6);

ha2 : ha
 port map(a => c6, b  => b7,
          vdd => vdd, vss => vss,
          sum => s7, carry => c7);

end structural;
```

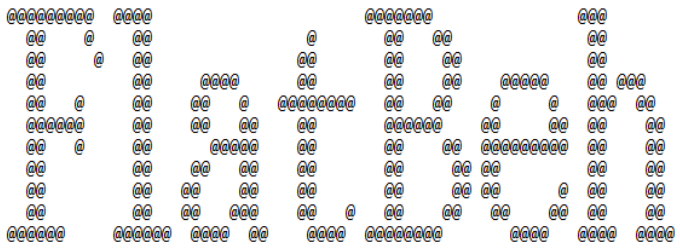
Εικόνα 5.25

Θα κάνουμε χρήση του εργαλείου Flatbeh για να μετατρέψουμε την structural περιγραφή σε behavioral και στην συνέχεια για τα εργαλεία BOOM,BOOG,LOON θα χρησιμοποιήσουμε ακριβώς τις ίδιες επιλογές, όπως και στο προηγούμενο κεφάλαιο καθώς θέλουμε να πετύχουμε τα ίδια αποτελέσματα, με την μόνη διαφορά να είναι το αρχείο που χρησιμοποιούμε σε αυτό το βήμα που αρχικά είναι το myadd5.vbe που θα αποκτήσουμε από την χρήση του Flatbeh.

Αρχικά θα χρησιμοποιήσουμε το εργαλείο Flatbeh του οποίου η εντολή φαίνεται στην γραμμή που ακολουθεί και το αποτέλεσμά του παρουσιάζεται στην Εικόνα 5.26.

```
flatbeh myadd5 myadd5
```

```
[cadence@sonetto alliance]$ flatbeh myadd5 myadd5
```



```
                        a netlist abstractor

          Alliance CAD System 5.0, flatbeh 5.0 [2000/11/01]
          Copyright (c) 1993-2021,           ASIM/LIP6/UPMC
          Author(s):       Fran\ufffdois DONNET, Huu Nghia VUONG
          E-mail        :       alliance-users@asim.lip6.fr

    ========================  Environnement  ========================
    MBK_WORK_LIB        = .
    MBK_CATA_LIB        = .:/home/cadence/alliance/install/cells/sxlib:/home/cadence/alliance/install/cells/dp_sxlib:/home/cadence
    MBK_CATAL_NAME      = CATAL
    ============================== Files ==============================
    Netlist file        = myadd5.vst
    Output  file        = myadd5.vbe
    ===================================================================

Loading './myadd5.vst'
flattening figure myadd5
 loading oa2ao222_x2
 loading a2_x2
 loading xr2_x1
Restoring array's orders
BEH : Saving 'myadd5' in a vhdl file (vbe)
```

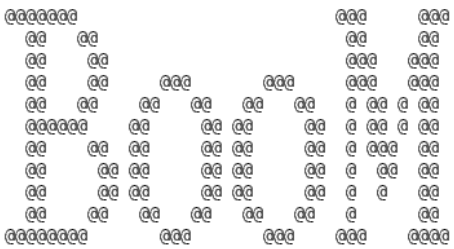
Εικόνα 5.26

Έχοντας αποκτήσει το αρχείο myadd5.vbe που απαιτείται για την χρήση του BOOM προχωράμε στην χρήση του εργαλείου για την βελτιστοποίηση του αρχείου. Στην Εικόνα 5.27 παρουσιάζεται η έξοδος του.

Κάνουμε χρήση του εργαλείου BOOM με την ακόλουθη εντολή:

```
boom -l 3 -d 0 myadd5 add5
```

```
[cadence@sonetto alliance]$ boom -l 3 -d 0 myadd5 add5
```



```
                BOOlean Minimization

Alliance CAD System 5.0,           boom 5.0
Copyright (c) 2000-2021,     ASIM/LIP6/UPMC
Author(s):                  Ludovic Jacomme
E-mail        : alliance-users@asim.lip6.fr

--> Parse BEH file myadd5.vbe
--> Drive BEH file add5
```

Εικόνα 5.27

Παίρνουμε το ήδη βελτιστοποιημένο αρχείο add5.vbe και χρησιμοποιούμε το BOOG για περεταίρω βελτιστοποίηση ακριβώς όπως κάναμε και στις άλλες περιπτώσεις.

Χρησιμοποιούμε το BOOG με την εντολή που ακολουθεί και τα αποτελέσματά του φαίνονται στην Εικόνα 5.28.

```
boog add5 sadd5 -x 0 -m 4
```

```
[cadence@sonetto alliance]$ boog add5 sadd5 -x 0 -m 4

                 @@@@@@@                          @@@@ @
                   @@   @@                          @@    @@
                   @@    @@                        @@      @
                   @@    @@      @@@       @@@     @@
                   @@   @@     @@   @@   @@   @@   @@
                   @@@@@@     @@     @@ @@     @@ @@      @@@@@
                   @@    @@   @@     @@ @@     @@ @@     @ @@
                   @@     @@ @@     @@ @@     @@ @@    @   @@
                   @@     @@ @@     @@ @@     @@  @@      @@
                   @@    @@   @@   @@   @@   @@    @@    @@
                 @@@@@@@@       @@@       @@@       @@@@

                        Binding and Optimizing On Gates

                    Alliance CAD System 5.0, boog 5.0 [2003/01/09]
                    Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
                    Author(s):                    Fran\ufffdois Donnet
                    E-mail        :    alliance-users@asim.lip6.fr

        MBK_VDD        : vdd
        MBK_VSS        : vss
        MBK_IN_LO      : vst
        MBK_OUT_LO     : vst
        MBK_WORK_LIB   : .
        MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'add5.vbe'...
Controlling file 'add5.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'add5.vbe'...
Capacitances on file 'add5.vbe'...
Unflattening file 'add5.vbe'...
Mapping file 'add5.vbe'...
Saving file 'sadd5.vst'...
Quick estimated critical path (no warranty)...2393 ps from 'a4' to 's7'
Quick estimated area (with over-cell routing)...71000 lambda\ufffd
Details...
        xr2_x1: 11
        oa2a22_x2: 5
        na2_x1: 5
        inv_x2: 5
        a2_x2: 4
        no2_x1: 3
        o2_x2: 3
        noa22_x1: 2
        oa22_x2: 2
        a3_x2: 2
        no4_x1: 1
        nao2o22_x1: 1
        ao2o22_x2: 1
        Total: 45
Saving critical path in xsch color file 'sadd5.xsc'...
End of boog...
```

Εικόνα 5.28

Τέλος ακολουθεί το LOON που το χρησιμοποιούμε με την εντολή που φαίνεται παρακάτω και στην Εικόνα 5.29 βλέπουμε τα αποτελέσματά του.

```
loon -x 0 -m 4 sadd5 add5
```

```
[cadence@sonetto alliance]$ loon -x 0 -m 4 sadd5 add5

                  @@@@@@                          @@@    @@@
                    @@                             @@     @
                    @@                             @@@    @
                    @@           @@@       @@@     @ @@   @
                    @@         @@   @@   @@   @@    @ @@   @
                    @@        @@     @@ @@     @@   @  @@   @
                    @@        @@     @@ @@     @@   @   @@  @
                    @@        @@     @@ @@     @@   @    @@ @
                    @@      @ @@     @@ @@     @@   @     @@@
                    @@     @   @@   @@   @@   @@   @      @@
                  @@@@@@@@@@     @@@       @@@    @@@     @@

                          Local optimization on Nets

                  Alliance CAD System 5.0, loon 5.0 [2003/12/07]
                  Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
                  Author(s):                     Fran\ufffdois Donnet
                  E-mail        :    alliance-users@asim.lip6.fr

          MBK_IN_LO       : vst
          MBK_OUT_LO      : vst
          MBK_TARGET_LIB  : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'sadd5.vst'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Capacitances on file 'sadd5.vst'...
Delays on file 'sadd5.vst'...2516 ps
Area on file 'sadd5.vst'...71000 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 11 (34%)
        oa2a22_x2: 5 (15%)
        na2_x1: 5 (7%)
        inv_x2: 5 (5%)
        a2_x2: 4 (7%)
        no2_x1: 3 (4%)
        o2_x2: 3 (5%)
        noa22_x1: 2 (4%)
        oa22_x2: 2 (4%)
        a3_x2: 2 (4%)
        no4_x1: 1 (2%)
        nao2o22_x1: 1 (2%)
        ao2o22_x2: 1 (3%)
        Total: 45
Worst RC on file 'sadd5.vst'...147 ps
Inserting buffers on critical path for file 'add5.vst'...None inserted
Improving RC on critical path for file 'add5.vst'...2462 ps
Improving all RC for file 'add5.vst'...
Worst RC on file 'add5.vst'...147 ps
Area on file 'add5.vst'...71000 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 11 (34%)
        oa2a22_x2: 5 (15%)
        na2_x1: 5 (7%)
        inv_x2: 5 (5%)
        a2_x2: 4 (7%)
        no2_x1: 3 (4%)
        o2_x2: 3 (5%)
        noa22_x1: 2 (4%)
        oa22_x2: 2 (4%)
        a3_x2: 2 (4%)
        no4_x1: 1 (2%)
        nao2o22_x1: 1 (2%)
        ao2o22_x2: 1 (3%)
        Total: 45
Critical path (no warranty)...2462 ps from 'a4' to 's7'
Saving file 'add5.vst'...
Saving critical path in xsch color file 'add5.xsc'...
End of loon...
```

Εικόνα 5.29

Με την απόκτηση του αρχείου add5.vst το οποίο είναι βελτιστοποιημένο έχουμε ολοκληρώσει και την σύνθεση του τρίτου και τελευταίου μπλοκ του σχεδιασμού μας και είμαστε σε θέση να προχωρήσουμε στο σχεδιασμό του πολλαπλασιαστή.

5.5 ΠΟΛΛΑΠΛΑΣΙΑΣΤΗΣ 4-bit

Έχουμε φτάσει στο τελευταίο στάδιο του σχεδιασμού μας το οποίο είναι η σύνθεση και η υλοποίησή του. Συνθέτοντας και τα τρία μπλοκ τα οποία αποτελούν τον σχεδιασμό μας μπορούμε να αρχίσουμε με την structural περιγραφή του η οποία φαίνεται στην Εικόνα 5.30.

```
entity multiplier4 is
 port(a0,a1,a2,a3 : in bit;
      b0,b1,b2,b3 : in bit;
      vdd,vss : in bit;
      p0,p1,p2,p3,p4,p5,p6,p7,p8 : out bit);
end multiplier4;

architecture structural of multiplier4 is
Component ppg4
 port(a0,a1,a2,a3 : in bit;
      b0,b1,b2,b3 : in bit;
      vdd,vss : in bit;
      p00,p01,p02,p03 : out bit;
      p10,p11,p12,p13 : out bit;
      p20,p21,p22,p23 : out bit;
      p30,p31,p32,p33 : out bit);
end component;

Component wallace
 port(p01,p02,p03 : in bit;
      p10,p11,p12,p13 : in bit;
      p20,p21,p22,p23 : in bit;
      p30,p31,p32,p33 : in bit;
      vdd,vss : in bit;
      p1,p2,p3a,p3b,p4a,p4b,p5a,p5b,p6a,p6b,p7b : out bit);
end component;

Component add5
 port(a3,a4,a5,a6 : in bit;
      b3,b4,b5,b6,b7 : in bit;
      vdd,vss : in bit;
      s3,s4,s5,s6,s7,c7 : out bit);
end component;

signal p01,p02,p03 : bit;
signal p10,p11,p12,p13 : bit;
signal p20,p21,p22,p23 : bit;
signal p30,p31,p32,p33 : bit;
signal p3a,p3b,p4a,p4b,p5a,p5b,p6a,p6b,p7b : bit;

begin
-- partial product generation
ppg : ppg4
 port map(a0 => a0, a1 => a1, a2 => a2, a3 => a3,
          b0 => b0, b1 => b1, b2 => b2, b3 => b3,
          vdd => vdd, vss => vss,
          p00 => p0, p01 => p01, p02 => p02, p03 => p03,
          p10 => p10, p11 => p11, p12 => p12, p13 => p13,
          p20 => p20, p21 => p21, p22 => p22, p23 => p23,
          p30 => p30, p31 => p31, p32 => p32, p33 => p33);

-- Wallace tree
w4 : wallace
 port map(p01 => p01, p02 => p02, p03 => p03,
      p10 => p10, p11 => p11, p12 => p12, p13 =>p13,
      p20 => p20, p21  =>p21, p22 => p22, p23 =>p23,
      p30 => p30, p31 => p31, p32 => p32, p33 =>p33,
      vdd => vdd, vss => vss,
      p1 => p1 , p2 => p2, p3a => p3a, p3b => p3b, p4a => p4a,
      p4b => p4b, p5a => p5a, p5b => p5b, p6a => p6a, p6b => p6b,
      p7b => p7b);

-- output adder

add : add5
 port map(a3 => p3a, a4 => p4a, a5 => p5a, a6 => p6a,
          b3 => p3b, b4 => p4b, b5 => p5b, b6 => p6b, b7 => p7b,
          vdd => vdd, vss => vss,
          s3 => p3, s4 => p4, s5 => p5, s6 => p6, s7 => p7, c7 => p8);

end structural;
```

Εικόνα 5.30

Θέτουμε τα bit εισόδου και εξόδου για τον πολλαπλασιαστή και στην συνέχεια ορίζουμε τα component και αυτά με την σειρά του με τα bit εισόδου και εξόδου. Στην συνέχεια πραγματοποιούμε μία χαρτογράφηση port που στην πράξη είναι μια συσχέτιση των bit εισόδου και εξόδου των component αντίστοιχα.

Αποθηκεύουμε την structural περιγραφή σε ένα αρχείο με όνομα multiplier4.vst. Για να μπορέσουμε να προχωρήσουμε στην σύνθεση πρέπει να μετατρέψουμε την structural περιγραφή σε behavioral όπως κάναμε και προηγουμένως.

Τα αποτελέσματα από την χρήση του εργαλείου Flatbeh παρουσιάζονται στην Εικόνα 5.31 και η χρησιμοποίησή του πραγματοποιείται με την εντολή:

flatbeh mymultiplier4 mymultiplier4

```
[cadence@sonetto alliance]$ flatbeh mymultiplier4 mymultiplier4

      @@@@@@@@@  @@@@                   @@@@@@@             @@@
        @@    @    @@              @       @@   @@            @@
        @@     @   @@                @@     @@    @@           @@
        @@         @@     @@@@       @@     @@    @@@@@    @@ @@@
        @@   @     @@    @@   @   @@@@@@@@   @@   @@     @     @@@  @@
        @@@@@@     @@    @@   @@     @@      @@@@@@    @@     @@  @@    @@
        @@   @     @@      @@@@@    @@      @@    @@  @@@@@@@@@  @@    @@
        @@         @@    @@   @@    @@      @@     @@ @@         @@    @@
        @@         @@   @@    @@    @@      @@     @@ @@      @  @@    @@
        @@         @@   @@@    @@  @   @@    @@    @@    @@  @@    @@
      @@@@@@    @@@@@@  @@@@  @@    @@@@  @@@@@@@@@      @@@@   @@@@  @@@@

                          a netlist abstractor

              Alliance CAD System 5.0, flatbeh 5.0 [2000/11/01]
              Copyright (c) 1993-2021,          ASIM/LIP6/UPMC
              Author(s):       Fran\ufffdois DONNET, Huu Nghia VUONG
              E-mail       :       alliance-users@asim.lip6.fr

      ========================  Environnement  ========================
      MBK_WORK_LIB        = .
      MBK_CATA_LIB        = .:/home/cadence/alliance/install/cells/sxlib:/home/cadence/alliance/install/cells/dp_sxlib:/home/cadence/alliance/install/cells/rflib:/home/cadence/alliance/install/cells/rf2lib:/h
      MBK_CATAL_NAME      = CATAL
      ============================== Files ==============================
      Netlist file        = mymultiplier4.vst
      Output  file        = mymultiplier4.vbe
      ===================================================================

Loading './mymultiplier4.vst'
flattening figure mymultiplier4
 loading ao2o22_x2
 loading nao2o22_x1
 loading a3_x2
 loading oa22_x2
 loading no4_x1
 loading oa2a22_x2
 loading oa2ao222_x2
 loading inv_x1
 loading inv_x2
 loading noa2a2a23_x1
 loading o2_x2
 loading mx2_x2
 loading nao22_x1
 loading xr2_x1
 loading noa22_x1
 loading nxr2_x1
 loading no2_x1
 loading no3_x1
 loading na2_x1
 loading an12_x1
 loading a2_x2
Restoring array's orders
BEH : Saving 'mymultiplier4' in a vhdl file (vbe)
```

Εικόνα 5.31

Βλέπουμε ότι το εργαλείο φορτώνει όλες τις απαραίτητες πύλες από το κάθε που είναι απαραίτητες και παράγει το αρχείο που χρειάζεται σε behavioral συμπεριφορά για να προχωρήσουμε στην σύνθεση του σχεδιασμού μας.

Με το αρχείο multiplier4.vbe έτοιμο μπορούμε να προχωρήσουμε στο επόμενο βήμα της διαδικασίας που είναι η βελτιστοποίηση του συγκεκριμένου αρχείου. Θα κάνουμε χρήση του εργαλείου BOOM. Χρησιμοποιούμε το BOOM με την εντολή που φαίνεται παρακάτω και η έξοδος του φαίνεται στην Εικόνα 5.32.

```
boom -l 3 -d 0 mymultiplier4 multiplier4
```

- Επιλογή **l 3**: Καθορίζουμε το επίπεδο βελτιστοποίησης στο μέγιστο επίπεδο.
- Επιλογή **d 0**: Καθορίζουμε το την βελτιστοποίηση της καθυστέρησης.

```
[cadence@sonetto alliance]$ boom -l 3 -d 0 mymultiplier4 multiplier4

                  @@@@@@@                          @@@     @@@
                    @@   @@                         @@     @@
                    @@    @@                        @@@   @@@
                    @@    @@     @@@       @@@      @@@   @@@
                    @@   @@    @@   @@   @@   @@   @ @@ @ @@
                    @@@@@@    @@     @@ @@     @@  @ @@ @ @@
                    @@    @@  @@     @@ @@     @@  @ @@@  @@
                    @@     @@ @@     @@ @@     @@  @  @@  @@
                    @@     @@ @@     @@ @@     @@  @  @   @@
                    @@    @@   @@   @@   @@   @@   @      @@
                  @@@@@@@@       @@@       @@@    @@@    @@@@

                           BOOlean Minimization

                  Alliance CAD System 5.0,           boom 5.0
                  Copyright (c) 2000-2021,     ASIM/LIP6/UPMC
                  Author(s):                  Ludovic Jacomme
                  E-mail        : alliance-users@asim.lip6.fr

        --> Parse BEH file mymultiplier4.vbe
        --> Drive BEH file multiplier4
```

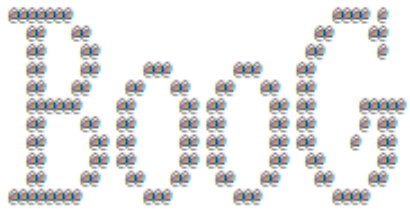
Εικόνα 5.32

Με την απόκτηση του βελτιστοποιημένου αρχείου προχωράμε στην χρήση του BOOG με σκοπό περισσότερη βελτιστοποίηση στις καθυστερήσεις, τον χρωματισμό της κρίσιμης περιοχής καθώς και χαρτογράφηση. Τα αποτελέσματα του BOOG απεικονίζονται στην Εικόνα 5.33 ,καθώς και ην εντολή με την οποία το χρησιμοποιούμε παρουσιάζεται στην επόμενη γραμμή.

```
boog multiplier4 smultiplier4 -x 0 -m 4
```

- Επιλογή **x 0:** Δημιουργεί ένα αρχείο .**xsc** το οποίο είναι ένας χρωματικός χάρτης για κάθε σχήμα που περιέχεται στο αρχείο εξόδου. Με το 0 χρωματίζουμε την κρίσιμη διαδρομή.
- Επιλογή **m 4**: Βελτιστοποίηση των καθυστερήσεων.

```
[cadence@sonetto alliance]$ boog multiplier4 smultiplier4 -x 0 -m 4
```



```
                Binding and Optimizing On Gates

               Alliance CAO System 5.0, boog 5.0 [2003/01/09]
               Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
               Author(s):                    Fran\ufffdois Donnet
               E-mail        :    alliance-users@asim.lip6.fr

        MBK_VDD         : vdd
        MBK_VSS         : vss
        MBK_IN_LO       : vst
        MBK_OUT_LO      : vst
        MBK_WORK_LIB    : .
        MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'multiplier4.vbe'...
Controlling file 'multiplier4.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'multiplier4.vbe'...
Capacitances on file 'multiplier4.vbe'...
Unflattening file 'multiplier4.vbe'...
Mapping file 'multiplier4.vbe'...
Saving file 'smultiplier4.vst'...
Quick estimated critical path (no warranty)...3153 ps from 'a0' to 'p3'
Quick estimated area (with over-cell routing)...275000 lambda\ufffd
Details...
        na2_x1: 38
        no2_x1: 33
        a2_x2: 17
        o2_x2: 14
        inv_x2: 12
        noa22_x1: 10
        oa22_x2: 9
        na3_x1: 8
        ao22_x2: 7
        no3_x1: 7
        xr2_x1: 7
        nao22_x1: 6
        o3_x2: 5
        oa2ao222_x2: 5
        nao2o22_x1: 5
        a3_x2: 4
        nxr2_x1: 4
        na4_x1: 2
        oa2a22_x2: 2
        mx3_x2: 2
        zero_x0: 1
        on12_x1: 1
        a4_x2: 1
        an12_x1: 1
        noa2a2a23_x1: 1
        oa2a2a2a24_x2: 1
        o4_x2: 1
        oa2a2a23_x2: 1
        Total: 205
Saving critical path in xsch color file 'smultiplier4.xsc'...
End of boog...
```

Εικόνα 5.33

Κοιτώντας την μεταβλητή MNK_TARGET_LIB βλέπουμε ότι το BOOG χρησιμοποιεί την τυπική cell βιβλιοθήκη για να πραγματοποιήσει χαρτογράφηση σε όλους τους δυαδικούς κόμβους που παρουσιάζεται στην συνέχεια, όπως είχαμε αναφέρει και στο αντίστοιχο υπό κεφάλαιο που αναλύσαμε τις λειτουργίες του BOOG. Διακρίνουμε επίσης ότι έχει χρωματίσει το κρίσιμο μονόπατη και έχει βελτιστοποιήσει τις καθυστερήσεις.

Στο βελτιστοποιημένο αρχείο multiplier4.vst που αποκτήσαμε από το BOOG θα χρησιμοποιήσουμε το LOON με σκοπό να μειώσουμε ακόμα πιο πού τις καθυστερήσεις. Το εργαλείο θα το πετύχει αυτό εισάγοντας buffer όπου είναι απαραίτητο.

Χρησιμοποιούμε το LOON στο τερματικό με την ακόλουθη εντολή:

```
loon -x 0 -m 4 smultiplier4 multiplier4
```

- Επιλογή **x 0**: Δημιουργεί ένα αρχείο .**xsc** το οποίο είναι ένας χρωματικός χάρτης για κάθε σχήμα που περιέχεται στο αρχείο εξόδου. Με το 0 χρωματίζουμε την κρίσιμη διαδρομή.
- Επιλογή **m 4**: Βελτιστοποίηση των καθυστερήσεων.

Στην Εικόνα 5.34 που ακολουθεί φαίνεται η έξοδος του εργαλείου LOON.

```
[cadence@sonetto alliance]$ loon -x 0 -m 4 smultiplier4 multiplier4
```

```
                Local optimization on Nets

            Alliance CAD System 5.0, loon 5.0 [2003/12/07]
            Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
            Author(s):                      Fran\ufffdois Donnet
            E-mail          :    alliance-users@asim.lip6.fr

        MBK_IN_LO       : vst
        MBK_OUT_LO      : vst
        MBK_TARGET_LIB  : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
100% delay optimization
Reading file 'smultiplier4.vst'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Capacitances on file 'smultiplier4.vst'...
Delays on file 'smultiplier4.vst'...3104 ps
Area on file 'smultiplier4.vst'...275000 lamda\ufffd (with over-cell routing)
Details...
        na2_x1: 38 (13%)
        no2_x1: 33 (12%)
        a2_x2: 17 (7%)
        o2_x2: 14 (6%)
        inv_x2: 12 (3%)
        noa22_x1: 10 (5%)
        oa22_x2: 9 (4%)
        na3_x1: 8 (3%)
        ao22_x2: 7 (3%)
        no3_x1: 7 (3%)
        xr2_x1: 7 (5%)
        nao22_x1: 6 (3%)
        o3_x2: 5 (2%)
        oa2ao222_x2: 5 (4%)
        nao2o22_x1: 5 (3%)
        a3_x2: 4 (2%)
        nxr2_x1: 4 (3%)
        na4_x1: 2 (1%)
        oa2a22_x2: 2 (1%)
        mx3_x2: 2 (2%)
        zero_x0: 1 (0%)
        on12_x1: 1 (0%)
        a4_x2: 1 (0%)
        an12_x1: 1 (0%)
        noa2a2a23_x1: 1 (0%)
        oa2a2a2a24_x2: 1 (1%)
        o4_x2: 1 (0%)
        oa2a2a23_x2: 1 (1%)
        Total: 205
Worst RC on file 'smultiplier4.vst'...400 ps
Inserting buffers on critical path for file 'multiplier4.vst'...None inserted
Improving RC on critical path for file 'multiplier4.vst'...3083 ps
Improving all RC for file 'multiplier4.vst'...
Worst RC on file 'multiplier4.vst'...400 ps
Area on file 'multiplier4.vst'...276000 lamda\ufffd (with over-cell routing)
Details...
        na2_x1: 38 (13%)
        no2_x1: 33 (11%)
        a2_x2: 17 (7%)
        o2_x2: 14 (6%)
        inv_x2: 11 (2%)
        noa22_x1: 10 (5%)
        oa22_x2: 9 (4%)
        na3_x1: 8 (3%)
        ao22_x2: 7 (3%)
        no3_x1: 7 (3%)
        xr2_x1: 7 (5%)
        nao22_x1: 6 (3%)
        o3_x2: 5 (2%)
        oa2ao222_x2: 5 (4%)
        nao2o22_x1: 5 (3%)
        a3_x2: 4 (2%)
        nxr2_x1: 4 (3%)
        na4_x1: 2 (1%)
        oa2a22_x2: 2 (1%)
        mx3_x2: 2 (2%)
        zero_x0: 1 (0%)
        on12_x1: 1 (0%)
        a4_x2: 1 (0%)
        an12_x1: 1 (0%)
        noa2a2a23_x1: 1 (0%)
        oa2a2a2a24_x2: 1 (1%)
        o4_x2: 1 (0%)
        oa2a2a23_x2: 1 (1%)
        inv_x8: 1 (0%)
        Total: 205
Critical path (no warranty)...3083 ps from 'a0' to 'p3'
Saving file 'multiplier4.vst'...
Saving critical path in xsch color file 'multiplier4.xsc'...
End of loon...
```

Εικόνα 5.34

Όπως μπορούμε να διακρίνουμε και το LOON χρησιμοποιεί την τυπική cell βιβλιοθήκη για να πραγματοποιήσει χαρτογράφηση σε όλους τους δυαδικούς κόμβους. Και παρουσιάζει για κάθε δυαδικό κόμβο το ποσοστό των καθυστερήσεων που έχει βελτιστοποιηθεί για τον καθένα αντίστοιχα. Στην συνέχεια βλέπουμε ότι γίνεται η εισαγωγή buffer για περεταίρω μείωση των καθυστερήσεων στο κρίσιμο μονοπάτι. Επίσης βελτιώνονται και οι RC καθυστερήσεις. Θυμίζουμε ότι το μοντέλο καθυστέρησης RC είναι μια μέτρηση που χρησιμοποιείται στο σχεδιασμό VLSI για τον υπολογισμό της καθυστέρησης σήματος μεταξύ της τάσης εισόδου και της τάσης εξόδου του σήματος εισόδου. Διακρίνουμε ότι υπάρχει βελτιστοποίηση καθυστέρησης χαρακτηριστικά στον δυαδικό κόμβο **no2_x1: 33 (11%)** που ήταν αρχικά στο 12%.

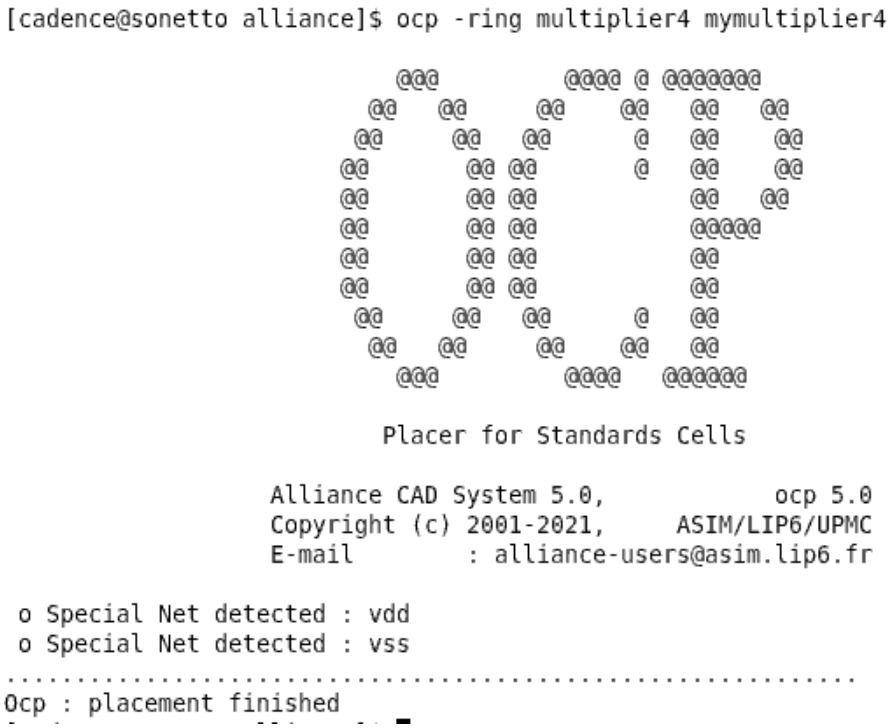
Με την υλοποίηση των διαδικασιών σύνθεσης και του LOON και λαμβάνοντας το βελτιστοποιημένο αρχείο multiplier4.vst προχωράμε στο στάδιο της τοποθέτησης και της δρομολόγησης.

Σε πρώτη φάση θα τοποθετήσουμε τον σχεδιασμό μας με το εργαλείο OCP. Σαν είσοδο θα πάρει το αρχείο που αποκτήθηκε από το LOON multiplier4.vst και σαν έξοδο θα παράγει ένα αρχείο multiplier4.ap. Στην Εικόνα 5.35 παρουσιάζονται τα αποτελέσματα αυτής της τοποθέτησης.

Χρησιμοποιούμε το εργαλείο του Alliance OCP με την εξής εντολή:

```
ocp -ring multiplier4 mymultiplier4
```

- Επιλογή **ring**: Το OCP τοποθετεί connectors για το εργαλείο τοποθέτησης ring. Η συγκεκριμένη τοποθέτηση θα πραγματοποιηθεί στην πλευρές που είναι η πηγή και η γείωση.



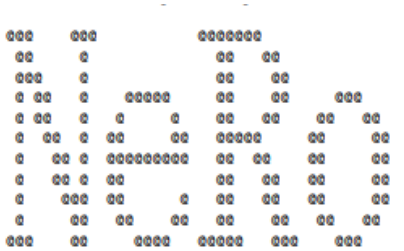
Εικόνα 5.35

Από την έξοδο εύκολα διακρίνουμε ότι η τοποθέτηση των connector έχει πραγματοποιηθεί ακριβώς στην γείωση και την πηγή του σχεδιασμού μας.

Τώρα μένει να δρομολογήσουμε τον σχεδιασμό μας χρησιμοποιώντας το εργαλείο NERO τα αποτελέσματα του οποίου φαίνονται στην Εικόνα 5.36.Χρησιμοποιούμε το NERO με την εντολή:

nero -V -2 -p mymultiplier4 multiplier4 multiplier4

- Επιλογή **V**: Verbose mode αλλά σε μεγαλύτερη κλίμακα.
- Επιλογή **2**: Ορίζει τον αριθμό των layer που χρησιμοποιούνται για την δρομολόγηση. Προεπιλογή για μικρά σχέδια όπως το δικό μας είναι 2, για μεγάλα σχέδια είναι 4.
- Επιλογή **p**: Καθορίζει όνομα αρχείου για το αρχείο τοποθέτησης διαφορετικό από αυτό που έχει η netlist.



```
                Negotiating Router

Alliance CAD System 5.0 20040928,  nero 5.0
Copyright (c) 2002-2005,     ASIM/LIP6/UPMC
E-mail        : alliance-users@asim.lip6.fr
                S/N 20021117.1

  o  MBK environment :
     MBK_IN_LO       := vst
...
     MBK_CATA_LIB    := .
                        /usr/local/alliance/cells/sxlib
...
                        /usr/local/alliance/cells/padlib
     MBK_CATAL_NAME  := CATAL
     MBK_VDD         := vdd
     MBK_VSS         := vss
     MBK_SEPAR       := .
  o  Loading netlist "multi4"...
  o  Loading layout "multi4_p"...
  o  Flattening layout...
  o  Flattening netlist...
  o  Building netlist dual representation (lofigchain)...
  o  Binding logical & physical views...
  o  Loading design into grid...
  o  Local routing stage.
     - [ 158] (hp :=     9) "r 0"
...
     - [   0] (hp :=   192) "x 1"

  o  Routing stats :
     - routing iterations    := 314327
     - re-routing iterations := 22122
     - ratio                 := 6.57514%.

  o  Dumping routing grid.
  o  Saving MBK figure "multi4".
  o  Saving layout as "multi4"...
```

Εικόνα 5.36

Με την απόκτηση του τοποθετημένου αρχείου multiplier4.ap έχουμε ολοκληρώσει και το στάδιο της τοποθέτησης και θα προχωρήσουμε στο στάδιο της του ελέγχου της ορθότητας του σχεδιασμού.

Θα χρησιμοποιήσουμε το εργαλείο COUGAR για να δημιουργήσουμε μια structural περιγραφή από το τοποθετημένο αρχείο μας. Το COUGAR εκτός από επαλήθευση του σχεδιασμού μας μας δίνει και πληροφορίες σχετικά με την παρασιτική χωρητικότητα όπως θα δούμε και στην Εικόνα 5.37 με την έξοδο του εργαλείου.

Με την παρακάτω εντολή κάνουμε χρήση του COUGAR.

```
cougar -f -v multiplier4 multiplier4_lay
```

- Επιλογή **v**: Η λειτουργία Verbose είναι ενεργοποιημένη. Κάθε βήμα της ανάλυσης εμφανίζεται στην τυπική έξοδο, με μερικά στατιστικά στοιχεία.

- Επιλογή **f**: Το cell συμβολικού layout ισοπεδώνεται στο επίπεδο καταλόγου πριν από την εξαγωγή. Χρησιμοποιείται το man catal για λεπτομέρειες στο συγκεκριμένο επίπεδο καταλόγου. Αν δεν υπάρχει ο κατάλογος ή είναι κενός, η netlist είναι μια διασύνδεση των τρανζίστορ, αλλιώς είναι μια διασύνδεση πυλών των οποίων τα ονόματα ορίζονται στον κατάλογο.

```
[cadence@sonetto alliance]$ cougar -f -v multiplier4 multiplier4_lay

                  Netlist extractor ... formerly Lynx

                Alliance CAD System 5.0,         cougar 1.21
                Copyright (c) 1998-2021,      ASIM/LIP6/UPMC
                Author(s): Ludovic Jacomme and Gregoire Avot
                Contributor(s):             Picault Stephane
                E-mail        :  alliance-users@asim.lip6.fr

        ---> Parse technological file /home/cadence/alliance/install/etc/cmos.rds

                RDS_LAMBDA        = 4
                RDS_UNIT          = 4
                RDS_PHYSICAL_GRID = 2
                MBK_SCALE_X       = 100

        ---> Extract symbolic figure multiplier4

                ---> Flatten figure

                ---> Translate Mbk -> Rds

                ---> Build windows
                <--- 360

                ---> Rectangles    : 9302
                ---> Figure size   : (   -100,   -100 )
                                     (  55100,  60100 )

                ---> Cut transistors
                <--- 0
                ---> Build equis
                <--- 226
                ---> Delete windows
                ---> Build signals
                <--- 226
                ---> Build instances
                <--- 316
                ---> Build transistors
                <--- 0
                ---> Save netlist

        <--- done !

        ---> Total extracted capacitance
        <--- 0.0pF
|
```

Εικόνα 5.37

Αναφέραμε ότι το COUGAR μας δίνει και παρασιτικές πληροφορίες. Όπως βλέπουμε χρησιμοποιεί το αρχείο τεχνολογίας cmos.rds από το οποίο λαμβάνονται οι φυσικές παράμετροι για να δοθούν οι πληροφορίες αυτές.

Με την υλοποίηση του COUGAR λάβαμε μια structural περιγραφή του σχεδιασμού μας το αρχείο multiplier4_lay.al το οποίο θα το συγκρίνουμε ,χρησιμοποιώντας το εργαλείο LVX, με το αρχείο multiplier4.vst που είχαμε αποκτήσει από το LOON. Με την σύγκριση αυτή θα διαπιστώσουμε αν η τοποθέτηση και η δρομολόγηση έχουν πραγματοποιηθεί σωστά.

Κάνουμε χρήση του LVXμε την εντολή που ακολουθεί και εμφανίζουμε τα αποτελέσματά του στην Εικόνα 5.38.

```
lvx vst vst multiplier4 multiplier4_lay
```

```
[cadence@sonetto alliance]$ lvx vst vst multiplier4 multiplier4_lay

                    @@@@@@      @@@@    @@@ @@@@   @@@@
                      @@          @@      @   @@     @
                      @@          @@      @    @@   @
                      @@           @@    @      @@ @
                      @@           @@    @       @@
                      @@            @@  @        @@
                      @@            @@  @        @@@
                      @@             @@@         @  @@
                      @@      @      @@@        @    @@
                      @@     @       @         @     @@
                    @@@@@@@@@@       @       @@@    @@@@

                          Gate Netlist Comparator

                    Alliance CAD System 5.0,           lvx 1.5
                    Copyright (c) 1992-2021,     ASIM/LIP6/UPMC
                    E-mail        : alliance-users@asim.lip6.fr


***** Loading multiplier4 (vst)...

Warning 2 : consistency checks will be disabled
***** Loading multiplier4_lay (vst)...


***** Compare Terminals ..............
***** O.K.      (0 sec)

***** Compare Instances ..............
***** O.K.      (0 sec)

***** Compare Connections ............................
***** O.K.      (0 sec)

===== Terminals .......... 19
===== Instances .......... 205
===== Connectors ......... 1143


***** Netlists are Identical. *****     (0 sec)
```

Εικόνα 5.38

Η έξοδος του LVX μας λέει ότι τα δύο αρχεία είναι πανομοιότυπα άρα η τοποθέτηση και η δρομολόγηση έχουν πραγματοποιηθεί σωστά.

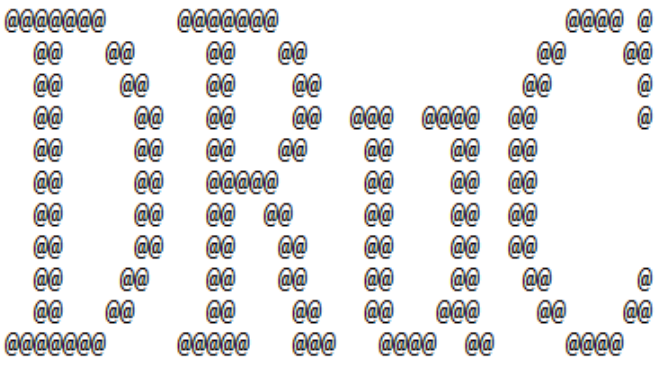
Στην συνέχεια θα πραγματοποιήσουμε έλεγχο κανόνων VLSI για να διαπιστώσουμε αν όλα τρέχουν σωστά πριν προχωρήσουμε στο τελευταίο βήμα του σχεδιασμό μας. Οι κανόνες που χρησιμοποιεί το εργαλείο DRUC ορίζονται από την μεταβλητή περιβάλλοντος

RDS_TECHNO_NAME και από το αρχείο τεχνολογίας cmos.rds. Θα χρησιμοποιήσουμε στο DRUC το αρχείο multiplier4.ap που αποκτήθηκε από τον δρομολογητή NERO.

Χρησιμοποιούμε το DRUC με την παρακάτω εντολή και τα αποτελέσματά του παρουσιάζονται στην Εικόνα 5.39.

druc multiplier4

```
[cadence@sonetto alliance]$ druc multiplier4
```



```
                Design Rule Checker

 Alliance CAD System 5.0,           druc 5.0
 Copyright (c) 1993-2021,     ASIM/LIP6/UPMC
 E-mail        : alliance-users@asim.lip6.fr

Flatten DRC on: multiplier4
Delete MBK figure : multiplier4
Load Flatten Rules : /home/cadence/alliance/install/etc/cmos.rds

Unify : multiplier4

Create Ring : multiplier4_rng
Merge Errorfiles:

Merge Error Instances:
instructionCourante :  56
End DRC on: multiplier4
Saving the Error file figure
Done
  0

File: multiplier4.drc is empty: no errors detected.
```

Εικόνα 5.39

Εκτελώντας τον έλεγχο κανόνων VLSI DRC βλέπουμε ότι δεν υπάρχει κάποια παραβίαση σε κάποιον από τους κανόνες και είμαστε σε θέση να εκτελέσουμε το τελευταίο βήμα του σχεδιασμού μας το οποίο είναι η υλοποίηση του πραγματικού layout.

Θα μετατρέψουμε το συμβολικό layout σε ένα πραγματικό χρησιμοποιώντας το αρχείο τεχνολογίας cmos.rds που παρέχεται από το Alliance. Η έξοδος του εργαλείου παρουσιάζεται στην Εικόνα 5.40.

Χρησιμοποιούμε το εργαλείο S2R του Alliance με την εξής εντολή στο τερματικό:

```
s2r multiplier4
```

```
[cadence@sonetto alliance]$ s2r multiplier4
                    @@@@
                   @   @@
                  @@    @@
                 @@@@@@  @@@   @@ @@@ @@@
                @@    @   @   @@   @@@  @@
                @@@          @    @@   @@
                 @@@@        @     @@
                   @@@@     @      @@
                @     @@@   @    @  @@
                @@    @@  @@@@@@   @@
                @ @@@@@  @@@@@@@  @@@@

                    Symbolic to Real layout converter

               Alliance CAD System 5.0,           s2r 5.0
               Copyright (c) 2002-2021,     ASIM/LIP6/UPMC
               E-mail       : alliance-users@asim.lip6.fr


        o loading technology file : /home/cadence/alliance/install/etc/cmos.rds
        o loading all level of symbolic layout : multiplier4
        o removing symbolic data structure
        o layout post-treating
                with top connectors,
                with sub connectors,
                with signal names,
                without scotch.
        o saving multiplier4.cif
|
```

Εικόνα 5.40

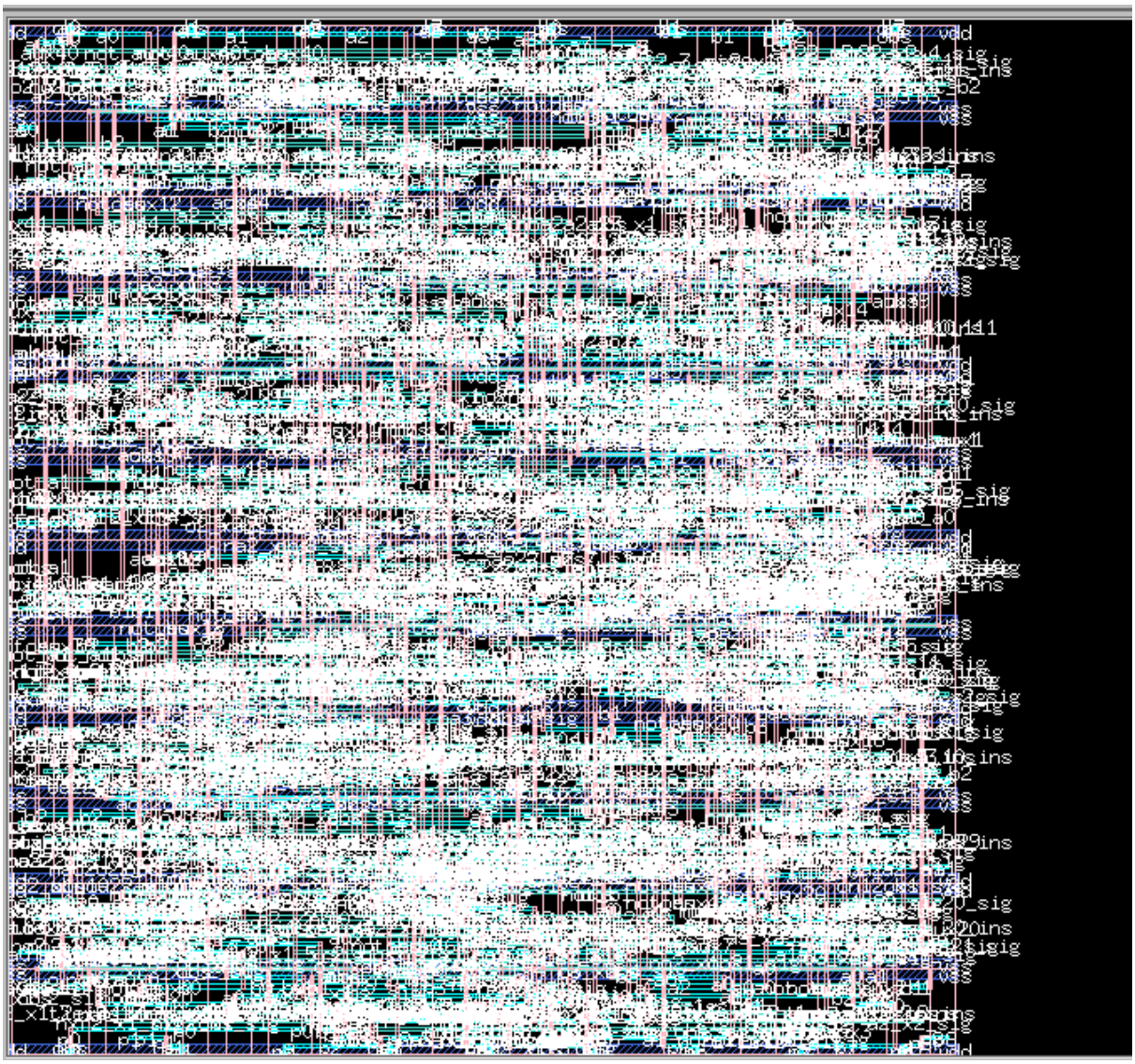
Όπως βλέπουμε και στην εικόνα το εργαλείο κάνει χρήση του αρχείου τεχνολογίας, φορτώνει το συμβολικό layout και αφαιρεί όλα τα συμβολικά δεδομένα και τέλος δημιουργεί το τελικό αρχείο multiplier4.cif που είναι ουσιαστικά ο πολλαπλασιαστής μας ολοκληρωμένος.

Για να μπορέσουμε να απεικονίσουμε τον τελικό αρχείου του σχεδιασμού θα χρησιμοποιήσουμε το εργαλείο απεικόνισης του Alliance το DREAL.

Χρησιμοποιούμε το συγκεκριμένο εργαλείο όπως φαίνεται παρακάτω:

Dreal

Μόλις τρέξουμε την εντολή μας εμφανίζει ένα παράθυρο πηγαίνουμε στο File, πατάμε Open και επιλέγουμε το αρχείο multiplier4.cif που αποκτήσαμε από το S2R. Το πραγματικό layout του σχεδιασμού μας απεικονίζεται στην Εικόνα 5.41.



Εικόνα 5.41

6.ΥΛΟΠΟΙΗΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΓΙΑ ΑΥΤΟΜΑΤΟΠΟΙΗΣΗ ΤΟΥ ΣΧΕΔΙΑΣΜΟΥ

Στο κεφάλαιο αυτό θα υλοποιήσουμε πάλι τον βασικό μας σχεδιασμό αυτόν του πολλαπλασιαστή 4-bit με την μόνη διαφορά ότι όλα τα απαραίτητα εργαλεία καθώς και όλες οι απαραίτητες μεταβλητές περιβάλλοντος θα βρίσκονται συγκεντρωμένα σε ένα πρόγραμμα. Στόχος αυτής της διαδικασίας είναι η αυτοματοποίηση της όλης διαδικασίας και η διευκόλυνση τυχόν μελλοντικές τροποποιήσεις-βελτιστοποιήσεις στον σχεδιασμό μας. Το πρόγραμμα που θα δημιουργήσουμε θα είναι ένα εκτελέσιμο πρόγραμμα γραμμής εντολών που θα το τρέχουμε εύκολα και γρήγορα από το τερματικό. Στην ροή του κεφαλαίου όμως για λόγους επεξήγησης θα παρουσιάζουμε το κάθε εργαλείο βήμα-βήμα με τα αποτελέσματά του. Στο τέλος του κεφαλαίου θα παρουσιαστεί και ο κώδικας του προγράμματος καθώς και ο τρόπος που θα τον τρέχουμε στο τερματικό με μία μόνο εντολή.

6.1 ΣΥΝΘΕΣΗ

Για να ξεκινήσουμε την σύνθεση του σχεδιασμού θα πάρουμε μια περιγραφή συμπεριφοράς(behavioral) που μας παρέχει το Alliance. Η περιγραφή αυτή είναι το αρχείο multi4.vhdl. Όπως διακρίνουμε το αρχείο έχει κατάληξη **.vhdl** είναι δηλαδή γραμμένο σε γλώσσα VHDL, αλλά εμείς χρειαζόμαστε ένα αρχείο συμπεριφοράς **.vbe**. Για το λόγο αυτό θα χρησιμοποιήσουμε το εργαλείο VASY για να μετατρέψουμε την περιγραφή VHDL σε περιγραφή συμπεριφοράς. Στην Εικόνα 6.1 που ακολουθεί παρουσιάζεται το αρχικό αρχείο multi4.vhdl.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Multi4 is
5
6
7      port ( X  : in Std_Logic_Vector(3 downto 0) ;
8             Y  : in Std_Logic_Vector(3 downto 0) ;
9             R  : out Std_Logic_Vector(7 downto 0) );
10
11 end Multi4;
12
13 ----------------------------------------------------------------------
14
15 architecture beh OF Multi4 is
16
17   signal PP1 : Std_Logic_Vector(4 downto 0);
18   signal PP2 : Std_Logic_Vector(4 downto 0);
19   signal PP3 : Std_Logic_Vector(4 downto 0);
20   signal PP4 : Std_Logic_Vector(4 downto 0);
21
22   signal PP12 : Std_Logic_Vector(5 downto 0);
23   signal PP34 : Std_Logic_Vector(5 downto 0);
24
25 begin
26
27    PP1(0) <= Y(0) and X(0);
28    PP1(1) <= Y(0) and X(1);
29    PP1(2) <= Y(0) and X(2);
30    PP1(3) <= Y(0) and X(3);
31    PP1(4) <= '0';
32
33    PP2(0) <= '0';
34    PP2(1) <= Y(1) and X(0);
35    PP2(2) <= Y(1) and X(1);
36    PP2(3) <= Y(1) and X(2);
37    PP2(4) <= Y(1) and X(3);
38
39    PP3(0) <= Y(2) and X(0);
40    PP3(1) <= Y(2) and X(1);
41    PP3(2) <= Y(2) and X(2);
42    PP3(3) <= Y(2) and X(3);
43    PP3(4) <= '0';
```

Εικόνα 6.1

Ένα αρχείο VHDL αποτελείται από 3 τμήματα τα οποία είναι:

1. Το τμήμα δήλωσης βιβλιοθηκών. Στο αρχείο μας χρησιμοποιούμε την βιβλιοθήκη IEEE. Βλέπουμε επίσης ότι χρησιμοποιούμε IEEE.STD_LOGIL_1164.ALL, που εισάγει τύπο δεδομένων std_logic και std_vector και επιτρέπει την χρήση περισσότερων τιμών σε ένα σήμα εκτός από τις απλές τιμές ‘0’ και ‘1’.
2. Το τμήμα δήλωσης της εξωτερικής μορφής του κυκλώματος -οντότητας(entity), όπου ορίζονται το όνομα το όνομα του και οι διεπαφές, σήματα εισόδου και εξόδου. Στον δικό μας κώδικα παρατηρούμε αυτό το τμήμα στις γραμμές 4 έως 11.
3. Περιγραφή της αρχιτεκτονικής της οντότητας (architecture) όπου περιγράφεται η λειτουργία και η συμπεριφορά του. Στις γραμμές 17 έως 23 είναι η περιοχή δήλωσης σημάτων και στις γραμμές 25 έως 43 είναι το κύριο σώμα της αρχιτεκτονικής.[3]

Πριν προχωρήσουμε στην χρησιμοποίηση του εργαλείου VASY θα πρέπει να ορίσουμε τις απαραίτητες μεταβλητές περιβάλλοντος που είναι απαραίτητες. Αυτές οι μεταβλητές είναι MBK_WORK_LIB= . , είναι προεπιλεγμένη ρύθμιση με την εγκατάσταση του Alliance και

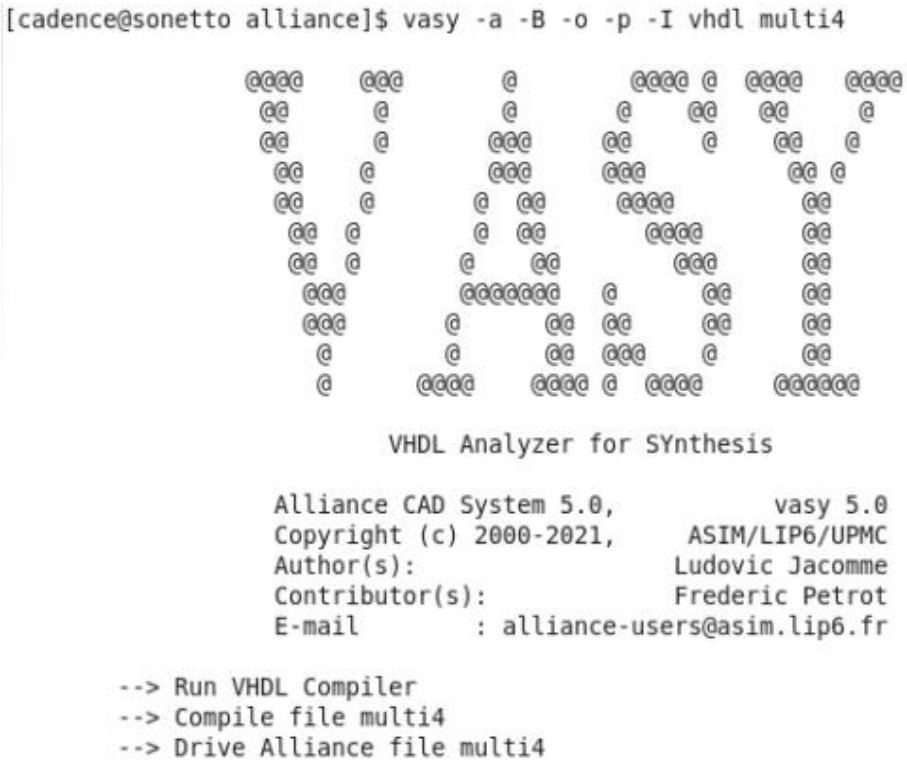
MBK_CATAL_NAME= NO_CATAL ουσιαστικά το CATAL είναι ένα αρχείο που περιέχει όλες τις πύλες. Ορίζουμε τις μεταβλητές αυτές με τις παρακάτω εντολές:

```
export MBK_CATAL_NAME= NO_CATAL
```

Έχοντας ορίσει τις μεταβλητές περιβάλλοντος τρέχουμε το εργαλείο VASY για την μετατροπή του αρχείου. Τρέχουμε το εργαλείο με την παρακάτω εντολή:

```
vasy -a -B -o -p -I vhdl multi4
```

Στην Εικόνα 6.2 βλέπουμε ότι το VASY τρέχει τον compiler VHDL στο αρχείο multi4.vhdl και αφού ολοκληρώσει τον έλεγχο παράγει το αρχείο multi4.vbe



Εικόνα 6.2

Όπως είχαμε δείξει και στο παράδειγμα με τον πολυπλέκτη θα χρησιμοποιήσουμε ένα αρχείο pattern το οποίο θα περιέχει τις τιμές εισόδου που θέλουμε να περάσουμε στο πολλαπλασιαστή και θα εξετάζονται σε αυτό το αρχείο όλες οι πιθανές τιμές των εισόδων. Στην Εικόνα 6.3 παρουσιάζεται ένα μικρό κομμάτι του αρχείου pattern λόγο της μεγάλης του έκτασης. Βλέπουμε ότι σαν bit εισόδου έχουμε δύο αριθμούς 4-bit και σαν έξοδο έχουμε έναν αριθμό 8-bit. Δίπλα με τον όρο X δηλώνουμε ότι πρόκειται για δεκαδικές τιμές. Στην συνέχεια δηλώνουμε εισόδους την πηγή και την γείωση. Με τον όρο B δηλώνουμε ότι πρόκειται για δυαδικές τιμές. Προχωρώντας ξεκινάμε και παίρνουμε όλους τους πιθανούς συνδυασμούς τιμών για τις εισόδους A και B και τις τροποποιούμε ανά 10ns για να

ελέγξουμε ότι το κύκλωμα λειτουργεί σωστά για όλες τις πιθανές τιμές εισόδου. Δίπλα από τις εισόδους A και B βλέπουμε το αποτέλεσμα Res όπου εκεί θα εμφανίζεται το αποτέλεσμα από την πράξη των αριθμών. Με τα σύμβολα ?** λέμε στον simulator να υπολογίζει κάθε τιμή εξόδου. Και τέλος βάζουμε την δυαδική τιμή '0' στην γείωση και την τιμή '1' στην πηγή. Το αρχείο έχει όνομα multi4.pat.

```
in       x (3 downto 0) X;;;
in       y (3 downto 0) X;;;
out      r (7 downto 0) X;;;
in       vss B;;
in       vdd B;;

begin

-- Pattern description :
--          A  B   Res     V V

< +10ns>: 0  0  ?**  0 1;
< +10ns>: 0  0  ?**  0 1;
< +10ns>: 0  1  ?**  0 1;
< +10ns>: 0  1  ?**  0 1;
< +10ns>: 1  0  ?**  0 1;
< +10ns>: 1  0  ?**  0 1;

< +10ns>: 1  1  ?**  0 1;
< +10ns>: 1  1  ?**  0 1;
< +10ns>: 1  2  ?**  0 1;
< +10ns>: 1  2  ?**  0 1;
< +10ns>: 1  3  ?**  0 1;
< +10ns>: 1  3  ?**  0 1;
< +10ns>: 1  4  ?**  0 1;
< +10ns>: 1  4  ?**  0 1;
< +10ns>: 1  5  ?**  0 1;
< +10ns>: 1  5  ?**  0 1;
< +10ns>: 1  6  ?**  0 1;
< +10ns>: 1  6  ?**  0 1;
< +10ns>: 1  7  ?**  0 1;
< +10ns>: 1  7  ?**  0 1;
```

Εικόνα 6.3

Θα προσομοιώσουμε τώρα την περιγραφή συμπεριφοράς χρησιμοποιώντας το αρχείο pattern multi4.pat με το εργαλείο ASIMUT. Και εδώ θα πρέπει να ορίσουμε κάποιες μεταβλητές περιβάλλοντος MBK_IN_LO, είναι η πιο σημαντική εντολή για το ASIMUT καθώς καθορίζει τον τύπο του αρχείου εισόδου. Επίσης ορίζουμε την MBK_OUT_LO, καθορίζει τον τύπο του αρχείου εξόδου και την μεταβλητή MBK_CATAL_NAME που αναφέραμε παραπάνω την λειτουργία της.

Με τις εντολές που ακολουθούν ορίζουμε τις μεταβλητές περιβάλλοντος.

```
export MBK_CATAL_NAME=CATAL_NAME_ASIMUT_VASY
export MBK_IN_LO=vst
export MBK_OUT_LO=vst
```

Τελειώνοντας με την ορισμό των μεταβλητών είμαστε έτοιμοι να εκτελέσουμε το εργαλείο ASIMUT για την προσομοίωση του σχεδιασμού μας. Στην Εικόνα 6.4 απεικονίζονται τα αποτελέσματά του.

Γράφουμε την ακόλουθη εντολή στο τερματικό για να τρέξουμε το ASIMUT:

```
asimut -b multi4 multi4 res_vasy_1
```

- Επιλογή **b**: Περιγραφή του κυκλώματος είναι behavioral.
- Το πρώτο αρχείο είναι το multi4.vbe που αποκτήθηκε από το VASY και είναι η περιγραφή συμπεριφοράς του κυκλώματος, το δεύτερο αρχείο είναι το multi4.pat είναι το αρχείο pattern με τις εισόδους και όλους τους πιθανούς συνδυασμούς και το αρχείο res_vasy_.pat είναι το αρχείο εξόδου με τα αποτελέσματα των εισόδων.

```
[cadence@sonetto alliance]$ asimut -b multi4 multi4 res_vasy_1

                                A SIMUlation Tool

                    Alliance CAD System 5.0,          asimut v3.02
                    Copyright (c) 1991...1999-2021, ASIM/LIP6/UPMC
                    E-mail        :    alliance-users@asim.lip6.fr

            Paris, France, Europe, Earth, Solar system, Milky Way, ...
initializing ...
searching `multi4` ...
BEH : Compiling `multi4.vbe` (Behaviour) ...
making GEX ...

searching pattern file : `multi4` ...
restoring ...

linking ...
executing ...
###----- processing pattern 0 : 10000 ps -----###
###----- processing pattern 1 : 20000 ps -----###
###----- processing pattern 2 : 30000 ps -----###
###----- processing pattern 3 : 40000 ps -----###
###----- processing pattern 4 : 50000 ps -----###
###----- processing pattern 5 : 60000 ps -----###
###----- processing pattern 6 : 70000 ps -----###
###----- processing pattern 7 : 80000 ps -----###
###----- processing pattern 8 : 90000 ps -----###
###----- processing pattern 9 : 100000 ps -----###
###----- processing pattern 10 : 110000 ps -----###
###----- processing pattern 11 : 120000 ps -----###
###----- processing pattern 12 : 130000 ps -----###
###----- processing pattern 13 : 140000 ps -----###
###----- processing pattern 14 : 150000 ps -----###
###----- processing pattern 15 : 160000 ps -----###
###----- processing pattern 16 : 170000 ps -----###
###----- processing pattern 17 : 180000 ps -----###
###----- processing pattern 18 : 190000 ps -----###
###----- processing pattern 19 : 200000 ps -----###
###----- processing pattern 20 : 210000 ps -----###
###----- processing pattern 21 : 220000 ps -----###
###----- processing pattern 22 : 230000 ps -----###
. . . .
###----- processing pattern 448 : 4490000 ps -----###
###----- processing pattern 449 : 4500000 ps -----###
###----- processing pattern 450 : 4510000 ps -----###
###----- processing pattern 451 : 4520000 ps -----###
###----- processing pattern 452 : 4530000 ps -----###
###----- processing pattern 453 : 4540000 ps -----###
###----- processing pattern 454 : 4550000 ps -----###
###----- processing pattern 455 : 4560000 ps -----###
```

Εικόνα 6 4

Η έξοδος των αποτελεσμάτων έχει περικοπή λόγο της έκτασής της.

Βλέπουμε ότι υπολογίζεται κάθε pattern για κάθε πιθανό σχεδιασμό καθώς και ότι απεικονίζεται και ανά πόσο χρονικό διάστημα παίρνει την επόμενη τιμή.

Ανοίγοντας το αρχείο res_vasy_.pat που αποκτήθηκε από το ASIMUT και κοιτώντας ενδεικτικά κάποια αποτελέσματα όπως φαίνεται στην Εικόνα 6.5 διαπιστώνουμε ότι ο πολλαπλασιαστής μας λειτουργεί σωστά και κάνει την πράξη του πολλαπλασιασμού.

Εικόνα 6.5

Προχωράμε την διαδικασία σύνθεσης με την χρήση του εργαλείου βελτιστοποίησης BOOM. Θα χρησιμοποιήσουμε το εργαλείο στο αρχείο συμπεριφοράς multi4.vbe που αποκτήσαμε με το VASY. Όπως και σε προηγούμενα βήματα έτσι και εδώ πριν από την χρήση του εργαλείου θα ορίσουμε τις απαραίτητες μεταβλητές περιβάλλοντος. Ορίζουμε την μεταβλητή MBK_CATAL_NAME όπως παρουσιάζεται στην ακόλουθη γραμμή:

```
export MBK_CATAL_NAME= CATAL
```

Χρησιμοποιούμε το BOOM με την παρακάτω εντολή και η έξοδος του παρουσιάζεται στην Εικόνα 6.6.

```
boom -VP multi4 multi4_o
```

- Επιλογή **V**: Λειτουργία Verbose ενεργοποιημένη. Κάθε βήμα της βελτιστοποίησης εμφανίζεται στην έξοδο.
- Επιλογή **P**: Χρησιμοποιεί ένα αρχείο .**boom** που περιγράφει οδηγίες βελτιστοποίησης και περιορισμούς.
- Το πρώτο αρχείο είναι το multi4.vbe και το δεύτερο είναι το βελτιστοποιημένο αρχείο εξόδου multi4_o.vbe.

```
@@@@@@@                          @@@     @@@
  @@   @@                          @@     @@
  @@    @@                         @@@   @@@
  @@    @@     @@@       @@@      @@@   @@@
  @@   @@    @@   @@   @@   @@    @ @@ @ @@
  @@@@@@    @@     @@ @@     @@   @ @@ @ @@
  @@    @@  @@     @@ @@     @@   @ @@@  @@
  @@     @@ @@     @@ @@     @@   @  @@  @@
  @@     @@ @@     @@ @@     @@   @  @   @@
  @@    @@   @@   @@   @@   @@    @      @@
@@@@@@@@       @@@       @@@     @@@    @@@@

                BOOlean Minimization

Alliance CAD System 5.0,           boom 5.0
Copyright (c) 2000-2021,     ASIM/LIP6/UPMC
Author(s):                  Ludovic Jacomme
E-mail        : alliance-users@asim.lip6.fr

--> Parse BEH file multi4.vbe

--> Check figure multi4

--> Parse parameter file multi4.boom
    keep signal 'rtlcarry_0 7'
    keep signal 'rtlcarry_0 6'
    keep signal 'rtlcarry_0 5'
    keep signal 'rtlcarry_0 4'
    keep signal 'rtlcarry_0 3'
    keep signal 'rtlcarry_0 2'
    keep signal 'rtlcarry_0 1'
    keep signal 'rtlcarry_3 5'
    keep signal 'rtlcarry_3 4'
    keep signal 'rtlcarry_3 3'
    keep signal 'rtlcarry_3 2'
    keep signal 'rtlcarry_3 1'
    keep signal 'rtlcarry_4 5'
    keep signal 'rtlcarry_4 4'
    keep signal 'rtlcarry_4 3'
    keep signal 'rtlcarry_4 2'
    keep signal 'rtlcarry_4 1'

--> Optimization parameters
    Algorithm : simulated annealing
    Keep aux  : no
    Area      : 100 %
    Delay     :   0 %
    Level     :   0

--> Initial cost
    Surface  : 216250
    Depth    : 25
    Literals : 194

--> Translate Abl to Bdd
    Total Bdd nodes 518

--> Optimization % 100

--> Final cost
    Surface  : 267750
    Depth    : 24
    Literals : 232

--> Post treat figure multi4

--> Drive BEH file multi4_o
```

Εικόνα 6.6

Ακριβώς με τον ίδιο τρόπο που λειτουργήσαμε στα προηγούμενα κεφάλαια έτσι και εδώ μετά το BOOM θα χρησιμοποιήσουμε το BOOG. Το BOOG θα πραγματοποιήσει περισσότερη βελτιστοποίηση στις καθυστερήσεις, τον χρωματισμό της κρίσιμης περιοχής καθώς και χαρτογράφηση. Τα αποτελέσματα του BOOG απεικονίζονται στην Εικόνα 6.7. Πριν εκτελέσουμε το BOOG στην μεταβλητή περιβάλλοντος TARGET_LIB θα ορίσουμε την βιβλιοθήκη στόχο sxlib στην οποία βασίζεται το BOOG για να δημιουργήσει την structural περιγραφή του σχεδιασμού μας. Ορίζουμε την μεταβλητή περιβάλλοντος με την εντολή export TARGET_LIB=$ALLIANCE_TOP//home/cadence/alliance/src/cells/src/sxlib.

Είμαστε έτοιμοι να τρέξουμε το εργαλείο BOOG με την εντολή που ακολουθεί:

boog multi4_o

```
[cadence@sonetto alliance]$ boog multi4_o
```

```
                Binding and Optimizing On Gates

             Alliance CAD System 5.0, boog 5.0 [2003/01/09]
             Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
             Author(s):                      Fran\ufffdois Donnet
             E-mail       :    alliance-users@asim.lip6.fr

       MBK_VDD        : vdd
       MBK_VSS        : vss
       MBK_IN_LO      : vst
       MBK_OUT_LO     : vst
       MBK_WORK_LIB   : .
       MBK_TARGET_LIB : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
50% area - 50% delay optimization
Reading file 'multi4_o.vbe'...
Controlling file 'multi4_o.vbe'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Mapping Warning: Cell 'noa3ao322_x4' isn't supported
Mapping Warning: Cell 'noa2ao222_x4' isn't supported
Mapping Warning: Cell 'halfadder_x4' isn't supported
Mapping Warning: Cell 'nts_x2' isn't supported
Mapping Warning: Cell 'oa2a22_x4' isn't supported
Mapping Warning: Cell 'halfadder_x2' isn't supported
Mapping Warning: Cell 'noa2a22_x4' isn't supported
Mapping Warning: Cell 'fulladder_x4' isn't supported
Mapping Warning: Cell 'nmx3_x4' isn't supported
Mapping Warning: Cell 'nao2o22_x4' isn't supported
Mapping Warning: Cell 'inv_x4' isn't supported
Mapping Warning: Cell 'no3_x4' isn't supported
Mapping Warning: Cell 'oa2a2a23_x4' isn't supported
Mapping Warning: Cell 'fulladder_x2' isn't supported
Mapping Warning: Cell 'buf_x8' isn't supported
Mapping Warning: Cell 'mx3_x4' isn't supported
Mapping Warning: Cell 'noa2a2a23_x4' isn't supported
Mapping Warning: Cell 'na4_x4' isn't supported
Mapping Warning: Cell 'ao22_x4' isn't supported
Mapping Warning: Cell 'nao22_x4' isn't supported
Mapping Warning: Cell 'o3_x4' isn't supported
Mapping Warning: Cell 'an12_x4' isn't supported
Mapping Warning: Cell 'ts_x8' isn't supported
Controlling lib '/home/cadence/alliance/install/cells/sxlib'...
Preparing file 'multi4_o.vbe'...
Capacitances on file 'multi4_o.vbe'...
Unflattening file 'multi4_o.vbe'...
Mapping file 'multi4_o.vbe'...
Saving file 'multi4_o.vst'...
Quick estimated critical path (no warranty)...3947 ps from 'x 0' to 'r 4'
Quick estimated area (with over-cell routing)...214250 lambda\ufffd
Details...
        xr2_x1: 32
        inv_x2: 28
        a2_x2: 15
        na2_x1: 7
        o3_x2: 6
        nao22_x1: 5
        no3_x1: 4
        oa2a22_x2: 4
        mx2_x2: 4
        nao2o22_x1: 4
        nxr2_x1: 4
        na3_x1: 4
        noa22_x1: 4
        no2_x1: 4
        zero_x0: 1
        one_x0: 1
        mx3_x2: 1
        buf_x2: 1
        noa2a22_x1: 1
        no4_x1: 1
        an12_x1: 1
        a3_x2: 1
        oa2ao222_x2: 1
        ao22_x2: 1
        oa22_x2: 1
        noa2ao222_x1: 1
        nmx3_x1: 1
        nmx2_x1: 1
        o2_x2: 1
        Total: 140
Saving critical path in xsch color file 'multi4_o.xsc'...
End of boog...
```

Εικόνα 6.7

Κοιτάζοντας την εντολή με την οποία χρησιμοποιήσαμε το BOOG αλλά και την εικόνα με την έξοδο του παρατηρούμε ότι δεν έχουμε δώσει επιλογές στο εργαλείο με αποτέλεσμα να χρησιμοποιεί τις προεπιλεγμένες επιλογές. Έτσι πραγματοποιεί 50% βελτιστοποίηση στην περιοχή και 50% βελτιστοποίηση στην καθυστέρηση(επιλογή -m 2). Ανατρέχοντας σε χρήσεις του BOOG σε προηγούμενα κεφάλαια παρατηρούμε ότι χρησιμοποιούσαμε την επιλογή **m 4** για βελτιστοποίηση μόνο της καθυστέρησης. Βλέπουμε και εδώ όπως και στις προηγούμενες χρήσεις του BOOG ότι πραγματοποιεί χαρτογράφηση με βάση την βιβλιοθήκη sxlib και χρωματίζει την κρίσιμη περιοχή.

Προχωράμε στο εργαλείο που ακολουθεί το BOOG το LOON το οποίο και έχει περίπου την ίδια λειτουργία με αυτή του BOOG. Σε αυτό το σημείο δεν χρειάζεται να ορίσουμε κάποια μεταβλητή περιβάλλοντος καθώς το LOON χρησιμοποιεί ακριβώς τις ίδιες μεταβλητές με τις ίδιες παραμέτρους.

Χρησιμοποιούμε το LOON στο τερματικό με την εξής εντολή:

```
loon multi4_o multi4
```

- Το αρχείο multi4_o.vst είναι το βελτιστοποιημένο αρχείο που αποκτήθηκε από το BOOG
- Το αρχείο multi4.vst είναι το βελτιστοποιημένο αρχείο εξόδου του LOON.

Στην Εικόνα 6.8 που ακολουθεί παρουσιάζεται η έξοδος από την εκτέλεση του εργαλείου LOON.

```
[cadence@sonetto alliance]$ loon multi4_o multi4
```

```
                Local optimization on Nets

             Alliance CAD System 5.0, loon 5.0 [2003/12/07]
             Copyright (c) 2000-2021,        ASIM/LIP6/UPMC
             Author(s):                      Fran\ufffdois Donnet
             E-mail       :    alliance-users@asim.lip6.fr

      MBK_IN_LO       : vst
      MBK_OUT_LO      : vst
      MBK_TARGET_LIB  : /home/cadence/alliance/install/cells/sxlib

Reading default parameter...
50% area - 50% delay optimization
Reading file 'multi4_o.vst'...
Reading lib '/home/cadence/alliance/install/cells/sxlib'...
Capacitances on file 'multi4_o.vst'...
Delays on file 'multi4_o.vst'...6992 ps
Area on file 'multi4_o.vst'...214250 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 32 (33%)
        inv_x2: 28 (9%)
        a2_x2: 15 (8%)
        na2_x1: 7 (3%)
        o3_x2: 6 (4%)
        nao22_x1: 5 (3%)
        no3_x1: 4 (2%)
        oa2a22_x2: 4 (4%)
        mx2_x2: 4 (4%)
        nao2o22_x1: 4 (3%)
        nxr2_x1: 4 (4%)
        na3_x1: 4 (2%)
        noa22_x1: 4 (2%)
        no2_x1: 4 (1%)
        zero_x0: 1 (0%)
        one_x0: 1 (0%)
        mx3_x2: 1 (1%)
        buf_x2: 1 (0%)
        noa2a22_x1: 1 (0%)
        no4_x1: 1 (0%)
        an12_x1: 1 (0%)
        a3_x2: 1 (0%)
        oa2ao222_x2: 1 (1%)
        ao22_x2: 1 (0%)
        oa22_x2: 1 (0%)
        noa2ao222_x1: 1 (0%)
        nmx3_x1: 1 (1%)
        nmx2_x1: 1 (0%)
        o2_x2: 1 (0%)
        Total: 140
Worst RC on file 'multi4_o.vst'...357 ps
Inserting buffers on critical path for file 'multi4.vst'...5 buffers inserted -> 6803 ps
Improving RC on critical path for file 'multi4.vst'...6747 ps
Improving all RC for file 'multi4.vst'...
Worst RC on file 'multi4.vst'...357 ps
Area on file 'multi4.vst'...219750 lamda\ufffd (with over-cell routing)
Details...
        xr2_x1: 32 (32%)
        inv_x2: 26 (8%)
        a2_x2: 15 (8%)
        na2_x1: 7 (3%)
        buf_x2: 6 (2%)
        o3_x2: 6 (4%)
        nao22_x1: 5 (3%)
        no3_x1: 4 (2%)
        oa2a22_x2: 4 (4%)
        mx2_x2: 4 (4%)
        nao2o22_x1: 4 (3%)
        nxr2_x1: 4 (4%)
        na3_x1: 4 (2%)
        noa22_x1: 4 (2%)
        no2_x1: 4 (1%)
        inv_x4: 2 (0%)
        zero_x0: 1 (0%)
        one_x0: 1 (0%)
        mx3_x2: 1 (1%)
        noa2a22_x1: 1 (0%)
        no4_x1: 1 (0%)
        an12_x1: 1 (0%)
        a3_x2: 1 (0%)
        oa2ao222_x2: 1 (1%)
        ao22_x2: 1 (0%)
        oa22_x2: 1 (0%)
        noa2ao222_x1: 1 (0%)
        nmx3_x1: 1 (1%)
        nmx2_x1: 1 (0%)
        o2_x2: 1 (0%)
        Total: 145
Critical path (no warranty)...6747 ps from 'x 0' to 'r 7'
Saving file 'multi4.vst'...
Saving critical path in xsch color file 'multi4.xsc'...
End of loon...
```

Εικόνα 6.8

Και εδώ βλέπουμε ότι έχουμε 50% βελτιστοποίηση της περιοχής και της καθυστέρησης καθώς δεν ορίζουμε εμείς κάποια επιλογή και χρησιμοποιεί το εργαλείο τις προκαθορισμένες επιλογές. Διακρίνουμε ότι έχουμε εισαγωγή 5 buffers για βελτιστοποίηση των καθυστερήσεων και φαίνεται ότι υπάρχει βελτιστοποίηση RC καθυστερήσεων. Πιο καθαρά μπορούμε να διακρίνουμε τις βελτιστοποιήσεις των καθυστερήσεων ενδεικτικά στην Εικόνα 6.9 που ακολουθεί, από 33% βελτιστοποιήθηκε στο 32% η καθυστέρηση και από 9% βελτιστοποιήθηκε στο 8% για λογικές πύλες xr2_x1 και inv_x2 αντίστοιχα.

```
...
xr2_x1: 32 (32%)
inv_x2: 26 (8%)
...
```

Εικόνα 6.9

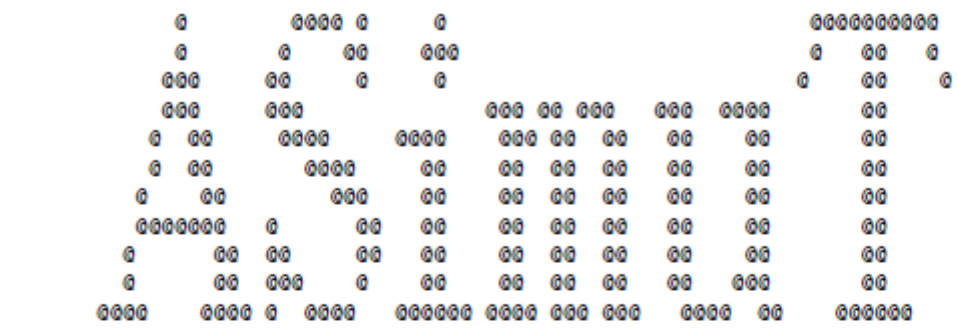
Έχοντας αποκτήσει το βελτιστοποιημένο αρχείο multi4.vst του σχεδιασμού μας έχουμε ολοκληρώσει και το στάδιο της διαδικασίας σύνθεσης και μπορούμε να προχωρήσουμε στο στάδιο της τοποθέτησης και δρομολόγησης.

Πριν προχωρήσουμε στην τοποθέτηση του σχεδιασμού μας θα χρησιμοποιήσουμε το εργαλείο ASIMUT για να ελέγξουμε αν η περιγραφή multi4.vst που μόλις αποκτήσαμε εξακολουθεί να λειτουργεί σωστά. Επιπρόσθετα για να χρησιμοποιήσουμε το ASIMUT χρειαζόμαστε και το pattern αρχείο multi4.pat που περιέχει τις τιμές εισόδου καθώς και όλους τους πιθανούς συνδυασμούς αυτών. Θα πρέπει πρώτα να αλλάξουμε την μεταβλητή περιβάλλοντος MBK_CATAL_NAME και να την ορίσουμε την βιβλιοθήκη cell sxlib βάση της οποίας έχει δημιουργηθεί ο σχεδιασμός με το BOOG και LOON. Ορίζουμε την μεταβλητή με την εντολή export MBK_CATAL_LIB=$TARGET_LIB. Στην Εικόνα 6.10 που ακολουθεί απεικονίζονται τα αποτελέσματα του εργαλείου ASIMUT.

Χρησιμοποιούμε το ASIMUT με ακόλουθη εντολή:

```
asimut multi4 multi4 res_synth_1
```

- Το πρώτο αρχείο είναι το αρχείο multi4.vst που αποκτήθηκε από το LOON, το δεύτερο αρχείο είναι το αρχείο pattern multi4.pat και το αρχείο res_synth_1.pat είναι αρχείο pattern που βγάζει σαν έξοδο το ASIMUT.



```
                          A SIMUlation Tool

                Alliance CAD System 5.0 20040928, asimut v3.02
                Copyright (c) 1991...1999-2005, ASIM/LIP6/UPMC
                E-mail        :    alliance-users@asim.lip6.fr

        Paris, France, Europe, Earth, Solar system, Milky Way, ...
initializing ...
searching 'multi4' ...
compiling 'multi4' (Structural) ...

flattening the root figure ...

searching 'ao2o22_x2' ...
BEH : Compiling 'ao2o22_x2.vbe' (Behaviour) ...
making GEX ...
...
searching 'buf_x2' ...
BEH : Compiling 'buf_x2.vbe' (Behaviour) ...
making GEX ...

searching pattern file : 'multi4' ...
restoring ...
linking ...
executing ...
###----- processing pattern 0 : 10000 ps -----###
...
###----- processing pattern 455 : 4560000 ps -----###
```

Εικόνα 6.10

Η έξοδος των αποτελεσμάτων έχει περικοπή λόγο της έκτασής της.

Βλέπουμε ότι υπολογίζεται κάθε pattern για κάθε πιθανό σχεδιασμό καθώς και ότι απεικονίζεται και ανά πόσο χρονικό διάστημα παίρνει την επόμενη τιμή.

Ανοίγοντας το αρχείο res_vasy_.pat που αποκτήθηκε από το ASIMUT και κοιτώντας ενδεικτικά κάποια αποτελέσματα όπως φαίνεται στην Εικόνα 6.11 διαπιστώνουμε ότι ο πολλαπλασιαστής μας λειτουργεί σωστά και κάνει την πράξη του πολλαπλασιασμού.

```
440000 ps>                                    : 2  4  ?08  0 1 ;
450000 ps>                                    : 2  5  ?08  0 1 ;
460000 ps>                                    : 2  5  ?0a  0 1 ;
470000 ps>                                    : 2  6  ?0a  0 1 ;
```

Εικόνα 6.11

Παρατηρούμε ότι ο πολλαπλασιασμός πραγματοποιείται σωστά αλλά στο δεύτερο πέρασμα απεικονίζεται μία θέση πιο κάτω στο επόμενο αποτέλεσμα σε σχέση με το αντίστοιχο pattern αρχείο που είχαμε σαν έξοδο προηγουμένως για behavioral περιγραφή. Αυτό συμβαίνει γιατί στην περίπτωση της behavioral περιγραφής η καθυστέρηση που χρησιμοποιήθηκε ήταν μηδέν. Σε αυτή την φάση έχουμε χρησιμοποιήσει τις καθυστερήσεις των αντίστοιχών κελιών που χρησιμοποιούνται για την υλοποίηση του σχεδιασμού μας. Κοιτώντας την Εικόνα 6.8 την έξοδο του LOON και την καθυστέρηση της κρίσιμης διαδρομής βλέπουμε ότι αυτή υπολογίζεται στα 6,4 ns ,έχοντας ως αποτέλεσμα την στιγμή της αλλαγή των εισόδων η αντίστοιχη έξοδος να περνάει στην έξοδο της συγκεκριμένης εισόδου.

Με την εκτέλεση και του ASIMUT έχουμε διαπιστώσει ότι ο σχεδιασμός μας εξακολουθεί να λειτουργεί σωστά είμαστε έτοιμοι να τοποθετήσουμε τον σχεδιασμό μας.

6.2 ΤΟΠΟΘΕΤΗΣΗ ΚΑΙ ΔΡΟΜΟΛΟΓΗΣΗ

Σε αυτό το βήμα δεν απαιτείται να ορίσουμε κάποια μεταβλητή περιβάλλοντος.

Χρησιμοποιούμε το OCP με την εντολή που ακολουθεί και η έξοδός του απεικονίζεται στις Εικόνες 6.13, 6.14.

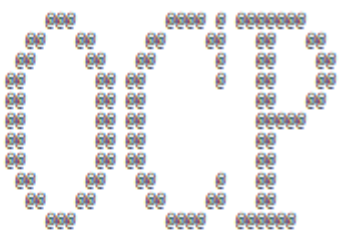
```
ocp -v -gnuplot -ioc multi4 multi4 multi4_p
```

- Επιλογή **v**: Λειτουργία Verbose ενεργοποιημένη.
- Επιλογή **gnuplot**: Δημιουργεί αρχεία gnuplot για την επεξεργασία στατιστικών.
- Επιλογή **ioc**: Η τοποθέτηση των αρχείων θα γίνει σύμφωνα με όσα ορίζονται στο αρχείο multi4.ioc. Το συγκεκριμένο αρχείο παρέχει στο σχεδιασμό μας τα pin εισόδου-εξόδου για το πάνω μέρος αυτό της εισόδου των δύο αριθμών x και y για αυτό είναι από το 0 έως το 3 καθώς είναι 4 bit, και για το κάτω μέρος αυτό του αποτελέσματος το οποίο είναι 8 bit για αυτό είναι από το 0 έως το 7. Στην Εικόνα 6.12 παρουσιάζεται ο αντίστοιχος κώδικας.

```
TOP ( # IOs are ordered from left to right
   (IOPIN x(3).0 );
   (IOPIN x(2).0 );
   (IOPIN x(1).0 );
   (IOPIN x(0).0 );
   (IOPIN y(3).0 );
   (IOPIN y(2).0 );
   (IOPIN y(1).0 );
   (IOPIN y(0).0 );
)
BOTTOM ( # IOs are ordered from left to right
   (IOPIN r(7).0 );
   (IOPIN r(6).0 );
   (IOPIN r(5).0 );
   (IOPIN r(4).0 );
   (IOPIN r(3).0 );
   (IOPIN r(2).0 );
   (IOPIN r(1).0 );
   (IOPIN r(0).0 );
)
IGNORE ( # IOs are ignored(not placed) by IO Placer
)
```

Εικόνα 6.12

```
[cadence@sonetto alliance]$ ocp -v -gnuplot -ioc multi4 multi4 multi4_p
```



```
                    Placer for Standards Cells

                Alliance CAD System 5.0,            ocp 5.0
                Copyright (c) 2001-2021,     ASIM/LIP6/UPMC
                E-mail         : alliance-users@asim.lip6.fr

o ALLIANCE environment:
  o ALLIANCE_TOP    : /home/cadence/alliance/install
o MBK environment:
  o MBK_IN_LO       : vst
  o MBK_OUT_LO      : vst
  o MBK_IN_PH       : ap
  o MBK_OUT_PH      : ap
  o MBK_VSS         : vss
  o MBK_VDD         : vdd
  o MBK_CATAL_NAME  : CATAL
  o MBK_CATA_LIB    : .
                      /home/cadence/alliance/install/cells/sxlib
                      /home/cadence/alliance/install/cells/dp_sxlib
                      /home/cadence/alliance/install/cells/rflib
                      /home/cadence/alliance/install/cells/rf2lib
                      /home/cadence/alliance/install/cells/ramlib
                      /home/cadence/alliance/install/cells/romlib
                      /home/cadence/alliance/install/cells/pxlib

 o Special Net detected : vss
 o Special Net detected : vdd
 o Number total of instances is ....  145
 o Number of instances to place is ....  145
 o Number of instances already placed is ....  0
 o Number of nets is .... 153
 o Sum of instances to place widths is ... 879
 o Computing Initial Placement ...
 o User Margin : 20%
 o Number of Rows : 10
 o Real Margin : 16.2857%
 o Width of the abutment box : 105
 o Height of the abutment box : 100
 o conspace : 13.125 1st connector : 6.5625
 o adding connector  : x 3 x : 6 y : 100
 o adding connector  : x 2 x : 19 y : 100
 o adding connector  : x 1 x : 32 y : 100
 o adding connector  : x 0 x : 45 y : 100
 o adding connector  : y 3 x : 59 y : 100
 o adding connector  : y 2 x : 72 y : 100
 o adding connector  : y 1 x : 85 y : 100
 o adding connector  : y 0 x : 98 y : 100
 o conspace : 13.125 1st connector : 6.5625
 o adding connector : r 7 x : 6 y : 0
 o adding connector : r 6 x : 19 y : 0
 o adding connector : r 5 x : 32 y : 0
 o adding connector : r 4 x : 45 y : 0
 o adding connector : r 3 x : 59 y : 0
 o adding connector : r 2 x : 72 y : 0
 o adding connector : r 1 x : 85 y : 0
 o adding connector : r 0 x : 98 y : 0
 o Initial Placement Computing ... done.
 o Beginning global placement ....
 o Initial RowCost = 65.4
 o Initial BinCost = 70.4629
 o Initial NetCost = 13709.5
 o Initial Cost = 1
 o Computing Initial Temperature ...
 o bins size 879
 o bins capa 879
 o subrows capa 879
Loop = 1, Temperature = 0.185489, Cost = 1.03399
  RowCost = 63.2, BinCost = 181, NetCost = 14175.5
  Success Ratio = 99.4127%, Dist = 1, Delta = 0.5
.....
```

Εικόνα 6.13

```
Loop = 49, Temperature = 8.59898e-05, Cost = 0.42394
  RowCost = 87.2, BinCost = 175.554, NetCost = 5812
  Success Ratio = 3.57715%, Dist = 0.1, Delta = 0.682434
 o Total impossible movements = 67327
 o 0.295572 % suroccupied target
 o 79.0188 % source equal target
 o 20.6856 % impossible exchange
 o Global Placement finished .....
 o Gain for RowCost      = -36.6972%
 o Gain for BinCost      = -144.798%
 o Gain for NetCost      = 57.606%
 o NetCost Estimated = 5812
 o Movements Stats ?!
 o 187445 Tried Moves
 o 2.22305 % of accepted simple instance move
 o 36.1722 % of accepted instance exchange
 o 1.98298 % of rejected simple instance move
 o 59.6218 % of rejected instance exchange
 o Impossible Movements Stats ....
 o If you find these values interesting, call a doctor...
 o Total impossible movements = 70198
 o 0.283484 % suroccupied target
 o 79.1547 % source equal target
 o 20.5618 % impossible exchange
 o Final Optimization in process ...
 o Net Cost before Final Optimization... 6243
 o Final Optimization succeeded ...
 o Final Net Cost ..... 4998.5
 o Final Net Cost Optimization ..... 19.9343%
 o Total Net Optimization .... 63.5399%

Ocp : placement finished
gnuplot files created - in order to use them, please type :
gnuplot afterglobal.gpl
gnuplot binsafterglobal.gpl
gnuplot init.gpl
gnuplot instances-init.gpl
gnuplot stat.gpl
gnuplot view.gpl
NO PREPLACEMENT GIVEN
 o Destruction of DATABASE ....
```

Εικόνα 6.14

Ο σχεδιασμός έχει τοποθετηθεί και έχοντας αποκτήσει το τοποθετημένο αρχείο multi4_p.ap προχωράμε στην δρομολόγησή του με το εργαλείο δρομολόγησης NERO. Δεν χρειάζεται να οριστεί κάποια μεταβλητή περιβάλλοντος καθώς το NERO χρησιμοποιεί ίδιες μεταβλητές με το OCP. Η μόνη διαφοροποίηση είναι ότι σε αυτό το σημείο θα πρέπει να οριστεί ο αριθμός των επιπέδων. Ορίζουμε την μεταβλητή με την εντολή export METAL_LEVEL=2.

Χρησιμοποιούμε το NERO με την ακόλουθη εντολή:

```
nero -V $metal_level -p multi4_p multi4 multi4
```

- Επιλογή **V**: Λειτουργία Verbose ενεργοποιημένη.
- Επιλογή **$metal_level**: Ο αριθμός των επιπέδων που ορίστηκε στην μεταβλητή.
- Επιλογή **p**: Καθορίζει ένα όνομα για το αρχείο τοποθέτησης διαφορετικό από το όνομα netlist.
- Το αρχείο multi4_p είναι το τοποθετημένο αρχείο multi4_p.ap που αποκτήθηκε από το OCP, το δεύτερο αρχείο είναι το αρχείο multi4.vst που είναι το βελτιστοποιημένο αρχείο από το LOON και το τρίτο αρχείο είναι το δρομολογημένο αρχείο από την χρησιμοποίηση του NERO.

Στην Εικόνα 6.15 παρουσιάζεται η έξοδος από την εκτέλεση του εργαλείου δρομολόγησης NERO.

```
[cadence@sonetto alliance]$ nero -V $metal_level -p multi4_p multi4 multi4
```

```
                Negotiating Router

Alliance CAD System 5.0,           nero 5.0
Copyright (c) 2002-2021,     ASIM/LIP6/UPMC
E-mail         : alliance-users@asim.lip6.fr

                S/N 20120503.1
```

```
o  MBK environment :

   MBK_IN_LO       := vst
   MBK_OUT_LO      := vst
   MBK_IN_PH       := ap
   MBK_OUT_PH      := ap
   MBK_WORK_LIB    := .
   MBK_CATA_LIB    := .
                      /home/cadence/alliance/install/cells/sxlib
                      /home/cadence/alliance/install/cells/dp_sxlib
                      /home/cadence/alliance/install/cells/rflib
                      /home/cadence/alliance/install/cells/rf2lib
                      /home/cadence/alliance/install/cells/ramlib
                      /home/cadence/alliance/install/cells/romlib
                      /home/cadence/alliance/install/cells/pxlib
   MBK_CATAL_NAME  := CATAL
   MBK_VDD         := vdd
   MBK_VSS         := vss
   MBK_SEPAR       := .

o  Loading netlist "multi4"...
o  Loading layout "multi4_p"...
o  Flattening layout...
o  Flattening netlist...
o  Building netlist dual representation (lofigchain)...
o  Binding logical & physical views...

o  Loading design into grid...
   o  Using seed cell "a2_x2_2_ins" (model "a2_x2").
   o  Grid offset : (0,0) [adjust (0,0)]
   o  Small design, global routing disabled.
   o  Allocating grid size [106,101,3].
   o  Loading external terminals.
   o  Finding obstacles.
   o  Loading nets into grid.
   o  Allocating the net scheduler.
   o  Reading power grid.

o  Local routing stage.
   - [ 154] (hp :=     0) "vdd"
   .......
   - [   0] (hp :=   135) "y 3"

o  Routing stats :
   - routing iterations    := 231003
   - re-routing iterations := 26863
   - ratio                 := 10.4174%.

o  Dumping routing grid.
o  Saving MBK figure "multi4".
o  Saving layout as "multi4"...
```

Εικόνα 6.15

Η έξοδος για λόγους χώρου καθώς ήταν αρκετά μεγάλη έχει περικοπή και παρουσιάζονται τα πιο σημαντικά σημεία της. Βλέπουμε ότι φορτώνονται τα δύο αρχεία η netlist multi4.vst και το τοποθετημένο αρχείο multi4_p.ap που αποτελεί ουσιαστικά ένα συμβολικό layout.

Στο στάδιο της τοπικής δρομολόγησης δρομολογούνται όλα τα στοιχεία που απαρτίζουν τον πολλαπλασιαστή ξεκινάει από το [154] που είναι η πηγή μέχρι και το [0] που είναι μία είσοδος y.

Αποκτώντας και το δρομολογημένο αρχείο multi4.ap έχουμε τελειώσει την διαδικασία τοποθέτησης και δρομολόγησης και προχωράμε στον έλεγχο της ορθότητας του σχεδιασμού.

6.3 ΕΛΕΓΧΟΣ ΟΡΘΟΤΗΤΑΣ ΣΧΕΔΙΑΣΜΟΥ

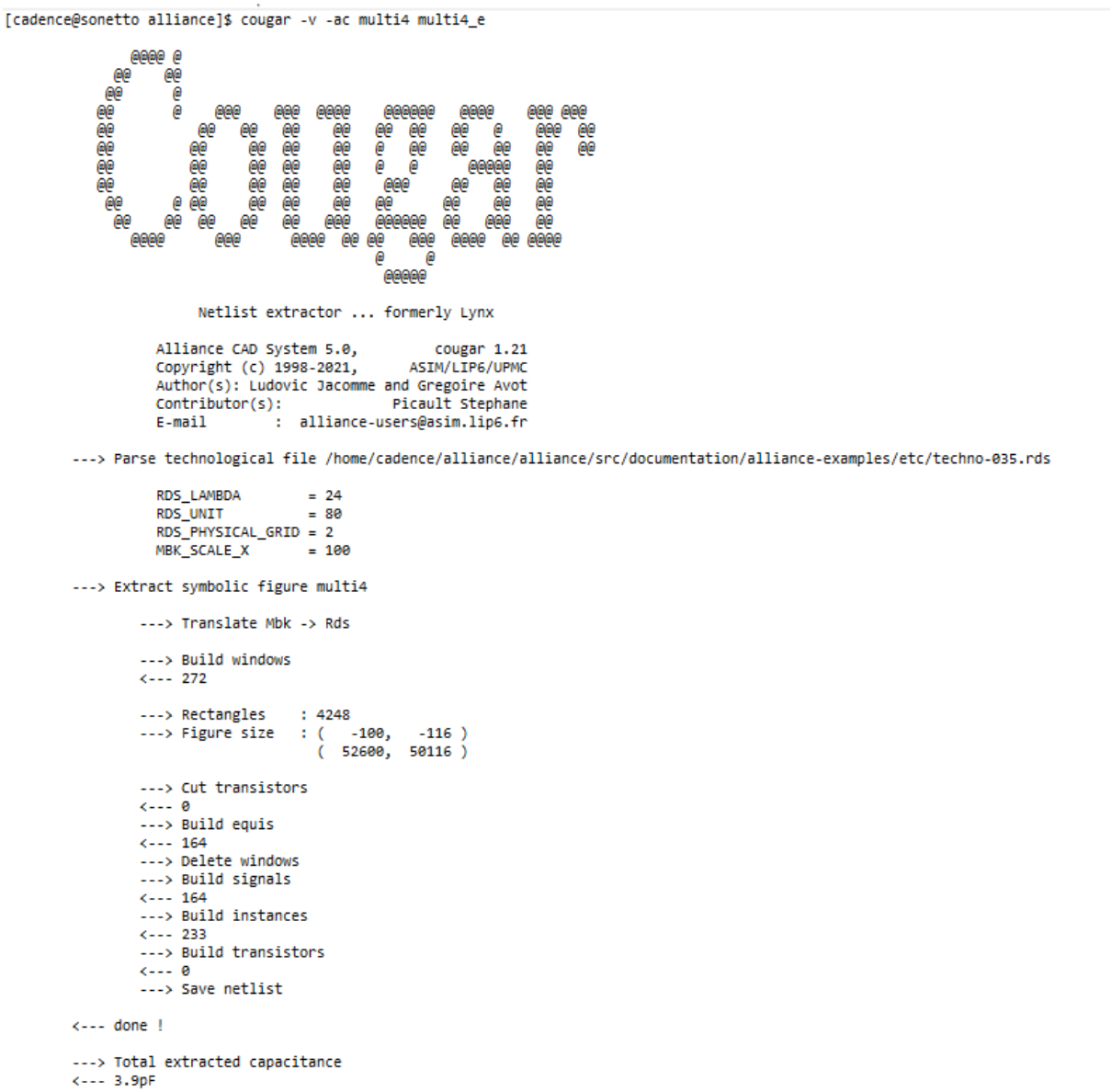
Για τον έλεγχο ορθότητας θα χρησιμοποιήσουμε δύο εργαλεία το COUGAR και το LVX.

Αρχικά θα χρησιμοποιήσουμε το εργαλείο COUGAR. Όπως έχουμε ήδη αναφέρει το συγκεκριμένο εργαλείο χρησιμοποιείται για την επαλήθευση σχεδίων. Για να εφαρμοστεί το COUGAR σε ένα πραγματικό layout παρέχουμε την αντίστοιχη τεχνολογία (RDS FILE), αλλά επίσης παράγει netlist με παρασιτικές πληροφορίες λαμβάνοντας φυσικές παραμέτρους από το αρχείο techno-035.rds. Για το λόγο αυτό θα πρέπει να ορίσουμε την μεταβλητή περιβάλλοντος RDS_TECHO_NAME. Ορίζουμε την συγκεκριμένη μεταβλητή με τις εξής εντολές export RDS_TECHNO=/home/cadence/alliance/alliance/src/documentation/alliance-examples/etc/techno-035.rds και export RDS_TECHNO_NAME=$RDS_TECHNO. Επειδή τα αρχεία που θα χρησιμοποιήσει ως είσοδο καθώς και το αρχείο που θα βγάλει ως έξοδο το COUGAR είναι της μορφής .**ap** πρέπει να αλλάξουμε και τις μεταβλητές MBK_IN_LO και MBK_OUT_LO. Ορίζουμε τις μεταβλητές αντίστοιχα ως εξής export MBK_IN_LO=al και MBK_OUT_LO=al. Ακόμα ορίζουμε και την μεταβλητή MBK_CATAL_LIB ως εξής MBK_CATAL_LIB=$TARGET_LIB.

Χρησιμοποιούμε το COUGAR με την παρακάτω εντολή και τα αποτελέσματά του παρουσιάζονται στην Εικόνα 6.16.

```
cougar -v -ac multi4 multi4_e
```

- Επιλογή **v**: Λειτουργία Verbose ενεργοποιημένη. Κάθε βήμα της εξαγωγής εμφανίζεται στην έξοδο, με μερικά στατιστικά στοιχεία.
- Επιλογή **ac**: Αποσυνδέει την χωρητικότητα από την γείωση σε περίπτωση απώλειας.
- Το πρώτο αρχείο είναι το αρχείο multi4.ap που αποκτήθηκε από το NERO και το δεύτερο είναι το αρχείου εξόδου multi4_e.al του COUGAR.



Εικόνα 6.16

Όπως μπορούμε να δούμε στην εικόνα το COUGAR χρησιμοποιεί το αρχείο τεχνολογία όπως έχουμε αναφέρει και παραπάνω και εξάγει επίσης και τις παρασιτικές πληροφορίες.

Ακόμα με το COUGAR μπορούμε να εξάγουμε και αρχείο SPICE ακριβώς όπως είχαμε κάνει και στο προηγούμενο κεφάλαιο. Θυμίζουμε ότι μια spice netlist είναι μια αναπαράσταση ενός κυκλώματος που βασίζεται σε κείμενο. Με την προβολή της netlist μας βοηθά να μάθουμε για την σύνταξη και την προσομοίωση του spice. Μας βοηθά ακόμα σε αρκετές περιπτώσεις στον εντοπισμό σφαλμάτων προσομοίωσης. Μια netlist επιπέδου spice είναι η network netlist RC(Resistance and Capacitance) που περιέχει πληροφορίες για τις τάσεις τροφοδοσίας και την γείωση. Το spice netlist χρησιμοποιείται για σχεδίαση

γραφήματος ρεύματος αντί για γραφήματος τάσης. Το spice είναι παρά πολύ συσχετισμένο με την πραγματική συμπεριφορά του κυκλώματος.[7][8]

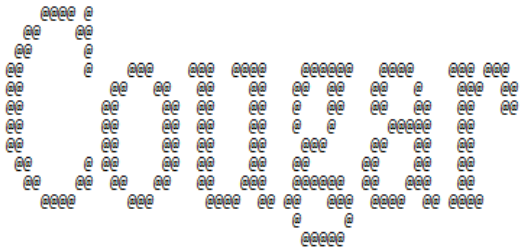
Για να αποκτήσουμε αρχείο SPICE πρέπει να ορίσουμε τις μεταβλητές περιβάλλοντος MBK_IN_LO και MBK_OUT_LO σε μορφή **spi**. Ορίζουμε τις μεταβλητές με τις εξής εντολές αντίστοιχα export MBK_IN_LO=spi export και MBK_OUT_LO=spi. Ακόμα πρέπει να ορίσουμε την μεταβλητή SPI_MODEL για να ορίσουμε το αρχείο για το μοντέλο spice κάτι αντίστοιχο με το αρχείο τεχνολογίας rds. Ορίζουμε και τις μεταβλητές MBK_SPI_MODEL, MBK_SPI_ONE_NODE_NORC και MBK_SPI_NAMEDNODES. Με τις ακόλουθες εντολές ορίζουμε τις μεταβλητές αντίστοιχα export SPI_MODEL=$ALLIANCE_TOP//home/cadence/alliance/alliance/src/mbu/etc/spimodel.cfg, export MBK_SPI_MODEL=$SPI_MODEL, export MBK_SPI_ONE_NODE=”true”, export MBK_SPI_NAMEDMODES=”true”.

Αφού έχουμε ορίσει όλες τις απαραίτητες μεταβλητές προχωράμε στην χρησιμοποίηση του COUGAR με την παρακάτω εντολή:

```
cougar -v -ac multi4 multi4_e
```

Οι επιλογές που χρησιμοποιούμε είναι ακριβώς ίδιες με εκείνες που χρησιμοποιήσαμε ακριβώς στην προηγούμενη χρήση του εργαλείου. Τα αποτελέσματα από την χρήση του COUGAR απεικονίζονται στην Εικόνα 6.17.

```
[cadence@sonetto alliance]$ cougar -v -ac multi4 multi4_e
```



```
                Netlist extractor ... formerly Lynx

            Alliance CAD System 5.0,         cougar 1.21
            Copyright (c) 1998-2021,      ASIM/LIP6/UPMC
            Author(s): Ludovic Jacomme and Gregoire Avot
            Contributor(s):             Picault Stephane
            E-mail        :  alliance-users@asim.lip6.fr

    ---> Parse technological file /home/cadence/alliance/alliance/src/documentation/alliance-examples/etc/techno-035.rds

            RDS_LAMBDA        = 24
            RDS_UNIT          = 80
            RDS_PHYSICAL_GRID = 2
            MBK_SCALE_X       = 100

    ---> Extract symbolic figure multi4

        ---> Translate Mbk -> Rds

        ---> Build windows
        <--- 272

        ---> Rectangles    : 4248
        ---> Figure size   : (   -100,   -116 )
                             (  52600,  50116 )

        ---> Cut transistors
        <--- 0
        ---> Build equis
        <--- 164
        ---> Delete windows
        ---> Build signals
        <--- 164
        ---> Build instances
        <--- 233
        ---> Build transistors
        <--- 0
        ---> Save netlist

    <--- done !

    ---> Total extracted capacitance
    <--- 3.9pF
|
```

Εικόνα 6.17

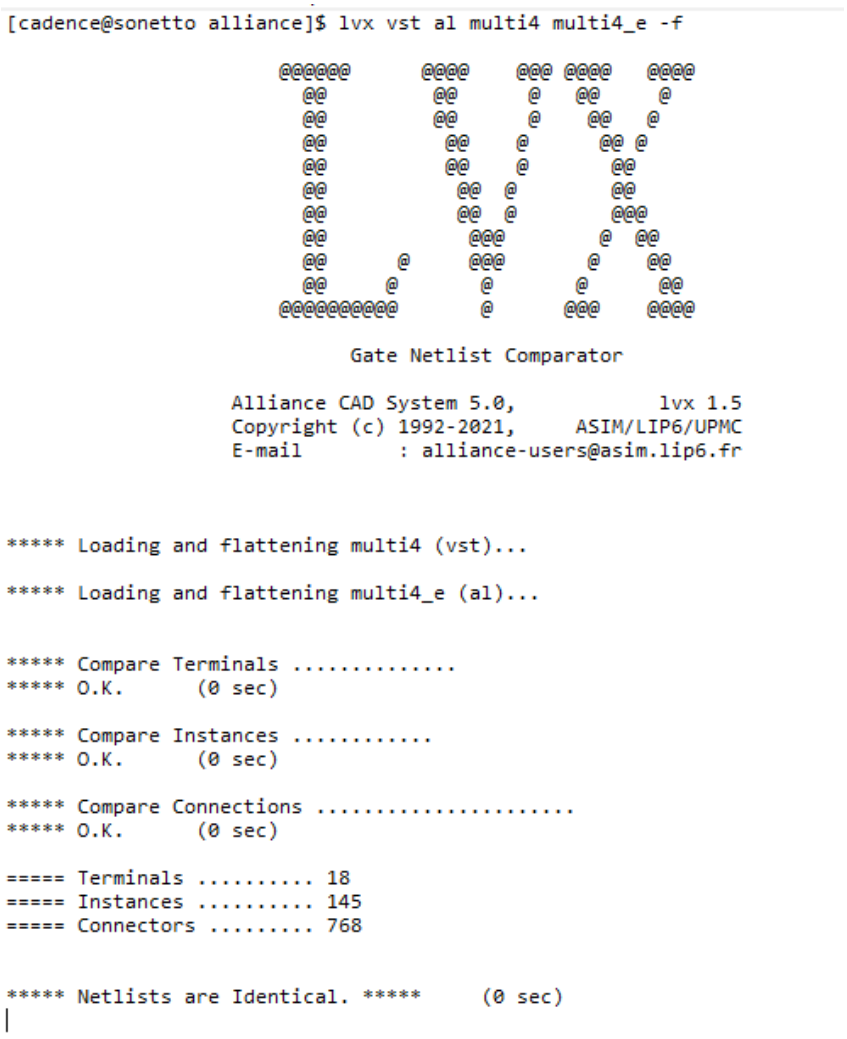
Έχουμε αποκτήσει και τα δύο αρχεία από το COUGAR και πάμε στο επόμενο εργαλείο για τον έλεγχο της ορθότητας του σχεδιασμού μας το LVX. Με το LVX θα συγκρίνουμε την structural περιγραφή multi4.vst που αποκτήθηκε από το LOON με το αρχείο multi4_e.al που αποκτήσαμε με το COUGAR. Θα πρέπει αρχικά να ορίσουμε τις μεταβλητές MBK_IN_LO και MBK_OUT_LO σε μορφή .**vst** καθώς η είσοδος του ενός αρχείου θα είναι structural. Ορίζουμε τις μεταβλητές με τις εντολές export MBK_IN_LO=vst export και MBK_OUT_LO=vst.

Χρησιμοποιούμε το LVX με την ακόλουθη εντολή και η έξοδος του φαίνεται στην Εικόνα 6.18.

lvx vst al multi4 multi4_e -f

- Επιλογή **vst** για το αρχείο multi4.vst και η επιλογή **al** για το αρχείο multi4_e.al.

- Επιλογή **f**: Οι δύο netlist είναι πεπλατυσμένες στα κυτταρικά φύλλα που περιέχονται στο αρχείο καταλόγου.



Εικόνα 6.18

Βλέπουμε ότι φορτώνονται τα δύο αρχεία και γίνεται σύγκριση στις συνδέσεις, τα τερματικά και μας εμφανίζει το μήνυμα ότι οι δυο netlist είναι πανομοιότυπες πράμα που σημαίνει ότι ο σχεδιασμός είναι σωστός.

Έχουμε ολοκληρώσει σε αυτό το σημείο και το στάδιο του ελέγχου της ορθότητας του σχεδιασμού και μπορούμε να προχωρήσουμε στο επόμενο στάδιο κανόνων σχεδιασμού.

6.4 ΕΛΕΓΧΟΣ ΚΑΝΟΝΩΝ ΣΧΕΔΙΑΣΜΟΥ

Θα χρησιμοποιήσουμε το εργαλείο DRUC για να ελέγξουμε το συμβολικό layout που έχουμε αποκτήσει από το εργαλείο NERO. Πριν από αυτό όμως όπως και κάθε φορά θα πρέπει να ορίσουμε κάποιες μεταβλητές περιβάλλοντος οι οποίες είναι RDS_TECHNO_SYMB και RDS_TECHNO_NAME με τις οποίες ορίζεται το αρχείο

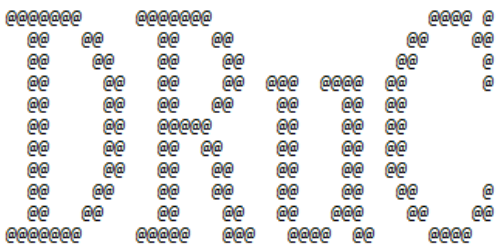
techno-symb.rds από το οποίο το εργαλείο χρησιμοποιεί τους κανόνες που θα ελέγξει. Ορίζουμε τις μεταβλητές με τις ακόλουθες εντολές:

export RDS_TECHNO_SYMB=/home/cadence/alliance/alliance/src/documentation/alliance-examples/etc/techno-symb.rds και export RDS_TECHNO_NAME=$RDS_TECHNO_SYMB

Χρησιμοποιούμε το DRUC με την εντολή που ακολουθεί στο τερματικό και τα αποτελέσματα της εξόδου του φαίνονται στην Εικόνα 6.19.

druc multi4

```
[cadence@sonetto alliance]$ druc multi4
```



```
                Design Rule Checker

 Alliance CAD System 5.0,           druc 5.0
 Copyright (c) 1993-2021,     ASIM/LIP6/UPMC
 E-mail         : alliance-users@asim.lip6.fr

Flatten DRC on: multi4
Delete MBK figure : multi4
Load Flatten Rules : /home/cadence/alliance/alliance/src/documentation/alliance-examples/etc/techno-symb.rds

Unify : multi4

Create Ring : multi4_rng
Merge Errorfiles:

Merge Error Instances:
instructionCourante :  56
End DRC on: multi4
Saving the Error file figure
Done
  0

File: multi4.drc is empty: no errors detected.
```

Εικόνα 6.19

Βλέπουμε ότι από τον έλεγχο των κανόνων σχεδιασμού δεν υπάρχουν σφάλματα όποτε το συμβολικό layout που έχουμε δημιουργήσει είναι σωστό και έτσι μπορούμε να προχωρήσουμε στο τελευταίο στάδιο αυτό της μετατροπής του συμβολικού layout σε πραγματικό.

6.5 ΜΕΤΑΤΡΟΠΗ ΣΥΜΒΟΛΙΚΟΥ LAYOUT ΣΕ ΠΡΑΓΜΑΤΙΚΟ LAYOUT

Φτάσαμε στο τελικό στάδιο του σχεδιασμού μας αυτό της παραγωγής του πραγματικού layout. Χρησιμοποιώντας το εργαλείο S2R θα μετατρέψουμε το συμβολικό layout multi4.ap σε ένα πραγματικό layout. Αρχικά πρέπει να ορίσουμε την μεταβλητή περιβάλλοντος RDS_TECHNO_NAME ώστε στο εργαλείο να χρησιμοποιήσει το απαραίτητο αρχείο τεχνολογίας για την μετατροπή του layout. Ορίζουμε την μεταβλητή με την εντολή export RDS_TECHNO_NAME=$RDS_TECHNO. Το S2R θα βγάλει ως έξοδο το αρχείο **multi4.cif.** Το αρχείο CIF (Calteck Interchange Format) είναι ένα από τα γνωστά αρχεία που χρησιμοποιούνται για το Tapeout, το οποίο είναι τα αποτελέσματα της διαδικασίας σχεδίασης για το σύστημά μας δημιουργώντας ένα αρχείο που περιέχει τις γεωμετρίες και τις πληροφορίες που χρειάζεται ο κατασκευαστής για να φτιάξει τον επεξεργαστή.

Χρησιμοποιούμε το S2R με την παρακάτω εντολή:

```
s2r -v multi4
```

- Επιλογή **v**: Λειτουργία Verbose ενεργοποιημένη.

Εικόνα 6.20

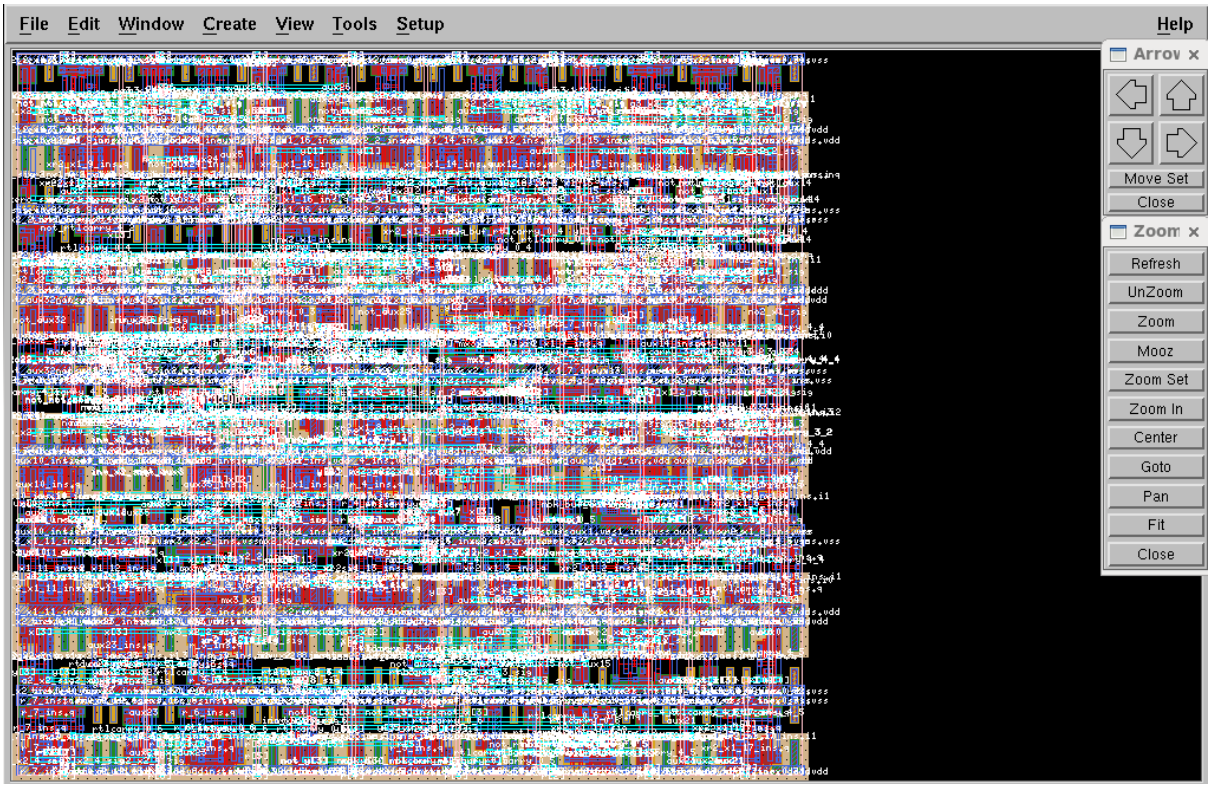
Όπως φαίνεται στην Εικόνα 6.20 κάθε component του πολλαπλασιαστή συγχωνεύεται σε ένα ορθογώνιο το οποίο αποτελείται από Nwell, Pwell, Nimp, Pimp,active,poly,ALU τα οποία θα αναλυθούν σε παρακάτω παρουσιάζοντας και ένα χρωματικό πίνακα για το που αντιστοιχεί το κάθε ένα μέσα στο layout του σχεδιασμού.

Μένει να χρησιμοποιήσουμε μόνο το εργαλείο απεικόνισης DREAL για να απεικονίσουμε το layout του πολλαπλασιαστή που έχουμε υλοποιήσει.

Χρησιμοποιούμε το DREAL με την εντολή που ακολουθεί:

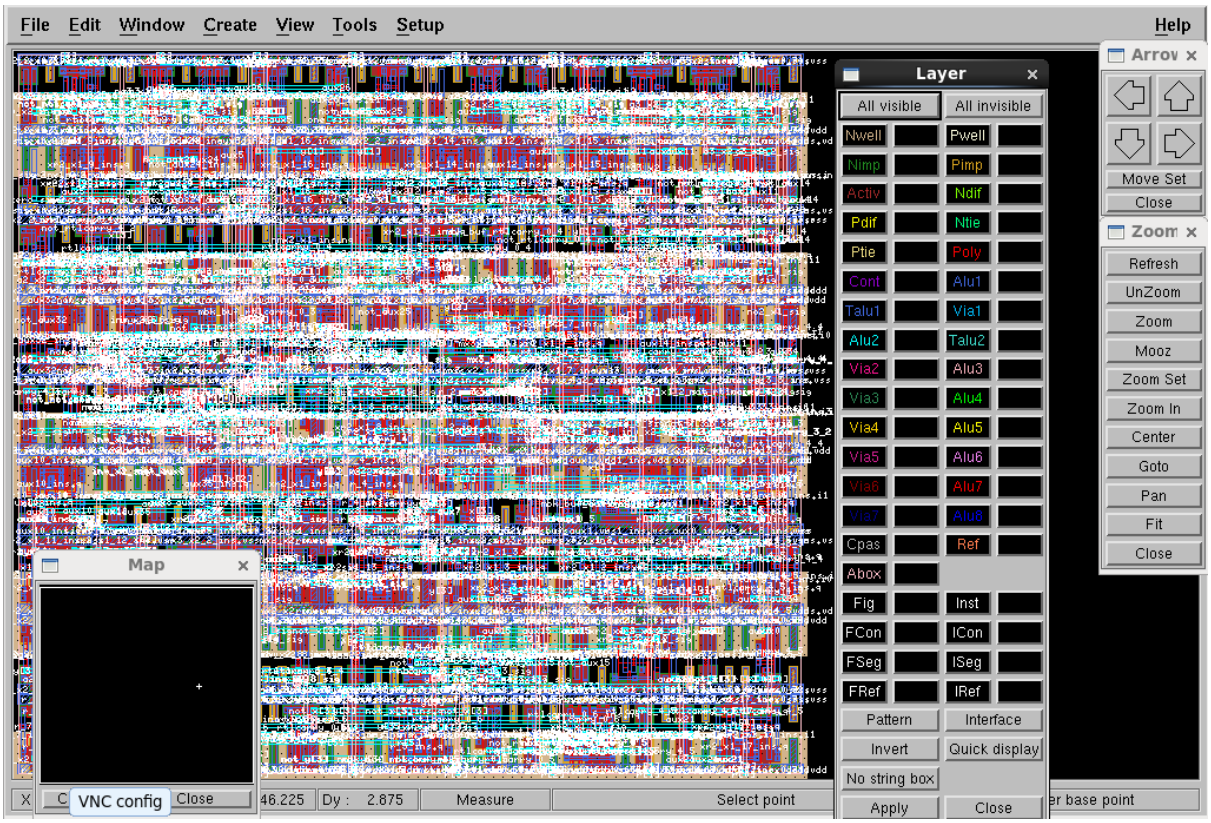
```
dreal
```

Μόλις τρέξουμε την εντολή μας εμφανίζει ένα παράθυρο πηγαίνουμε στο File, πατάμε Open και επιλέγουμε το αρχείο multi4.cif που αποκτήσαμε από το S2R. Το πραγματικό layout του σχεδιασμού μας απεικονίζεται στην Εικόνα 6.21.



Εικόνα 6.21

Στα δεξιά του παραθύρου βλέπουμε το παράθυρο Zoom από εκεί μπορούμε να κάνουμε ζουμ για να δούμε καλύτερα το layout του πολλαπλασιαστή, επίσης από την μπάρα εργαλείων μπορούμε να επιλέξουμε View→Map και από το παράθυρο που θα εμφανιστεί μπορούμε να κινούμαστε πιο εύκολα μέσα στο layout του σχεδιασμού μας. Από την μπάρα εργαλείων επιλέγοντας View→Layers εμφανίζεται μία παλέτα με όλα τα στοιχεία Nwell, Pwell, Nimp, Pimp,active,poly,ALU καθώς και τα αντίστοιχα χρώματά τους. Στην Εικόνα 6.22 φαίνεται η παλέτα καθώς και το χάρτης που αναφέραμε πιο πάνω.



Εικόνα 6.22

Χρησιμοποιώντας την παλέτα με τα χρώματα και τα ονόματα των στοιχείων καθώς και τον χάρτη και κάνοντας ζουμ μπορεί ο χρήστης να εντοπίσει που χρησιμοποιείται κάθε στοιχείο καθώς και να κάνει κάποια επαλήθευση στον σχεδιασμό αν υπάρχει κάποιο σφάλμα. Βέβαια λόγο της μικρής κλίμακας και των παρόμοιων χρωματισμών κάτι τέτοιο είναι αρκετά δύσκολο καθώς απαιτεί πολύ χρόνο πολύ προσεκτική παρατήρηση. Εμείς σε αυτό θα αρκεστούμε στην εξήγηση αυτών των στοιχείων.

- **Nwell-Pwell** : Στην τεχνολογία Nwell ένα well τύπου N διαχέεται σε υπόστρωμα τύπου p ενώ στο P είναι στο αντίστροφο.
- **Nimp – Pimp**: Για ένα Nmos η διάχυση θα είναι Nimp, ενώ για ένα Pmos η διάχυση θα είναι Pimp.
- **Activ**: Η ενεργή μάσκα ορίζει όλες τις περιοχές όπου πρόκειται να τοποθετηθεί διάχυση τύπου n ή τύπου p, ή τις περιοχές όπου πρόκειται να τοποθετηθούν οι πύλες των τρανζίστορ.
- **Ndif-Pdif**: Διάχυση πηγής και διάχυση γείωσης αντίστοιχα.
- **Cont**: Επαφές μεταξύ contract layer με πηγή, γείωση, υποστρώματος και well επαφές μέσω μεταλλικών βυσμάτων.
- **Ntif-Ptif:** Nwell πόλωση και Pwell πόλωση υποστρώματος αντίστοιχα.
- **Poly**: Καλώδιο polysilicon

- **Alu1-Alu2-…**: Πρώτο επίπεδο μετάλλου(ALU1), δεύτερο επίπεδο(ALU2) και ου το καθεξής
- **Via1-Via2-…**: Επαφή μεταξύ ALU1-ALU2(VIA1), επαφή μεταξύ ALU2-ALU3 και ου το καθεξής.[14][18][19]

6.6 ΠΑΡΟΥΣΙΑΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ ΕΝΑ ΕΚΤΕΛΕΣΙΜΟ ΑΡΧΕΙΟ

Θα παρουσιάσουμε το πρόγραμμα που έχουμε υλοποιήσει σε ένα εκτελέσιμο αρχείο. Αρκεί μόνο αυτό το αρχείο για να υλοποιήσουμε τον σχεδιασμό μας. Οι εντολές που γράψαμε σε αυτό το πρόγραμμα είναι όλες οι εντολές που έχουμε αναφέρει σε όλο το κεφάλαιο 6 καθώς και όλες οι μεταβλητές περιβάλλοντος. Η μόνη διαφοροποίηση είναι η χρήση κάποιων εντολών σε ορισμένα εργαλεία ώστε να ξέρουμε αν έχει τρέξει το συγκεκριμένο εργαλείο. Προσθέτουμε τις συγκεκριμένες εντολές διότι τώρα το κάθε εργαλείο δεν θα το τρέχουμε εμείς βήμα-βήμα στο τερματικό και έτσι να γνωρίζουμε αν κάθε εργαλείο έχει εκτελεστεί. Τα εργαλεία στα οποία θα προσθέσουμε την επιπλέον αυτή εντολή είναι το BOOM,BOOG,LOON,LVX και DRUC. Η εντολή αυτή είναι για κάθε εργαλείο αντίστοιχα touch boom.done, touch boog.done, touch loon.done, touch lvx.done, touch druc.done. Η εντολή αυτή μπαίνει ακριβώς κάτω από την εντολή με την οποία εκτελείται το κάθε εργαλείο και έχει σκοπό όπως αναφέραμε να μας ενημερώνει αν το κάθε εργαλείο έχει τρέξει.

Στην Εικόνα 6.23 παρουσιάζεται ο κώδικας του προγράμματος με όλες τις εντολές των εργαλείων, τις μεταβλητές περιβάλλοντος καθώς και τις εντολές που περιγράψαμε σε αυτό το κεφάλαιο.

```
#!/bin/sh
source /home/cadence/alliance/install/etc/profile.d/alc_env.sh



export MBK_CATAL_NAME=NO_CATAL
#multi4.vbe : multi4.vhdl
  vasy -a -B -o -p -I vhdl multi4
export MBK_CATAL_NAME=CATAL_NAME_ASIMUT_VASY
export MBK_IN_LO=vst
export MBK_OUT_LO=vst
#res_vasy_1.pat : multi4.vbe
  asimut -b multi4 multi4 res_vasy_1
export MBK_CATAL_NAME=CATAL


#multi4_o.vbe : multi4.vbe multi4.boom res_vasy_1.pat
      boom -VP multi4 multi4_o
#boom.done : multi4.vbe
  touch  boom.done
export TARGET_LIB=$ALLIANCE_TOP//home/cadence/alliance/alliance/src/cells/src/sxlib


#multi4_o.vst : multi4_o.vbe
        boog multi4_o
#boog.done : multi4_o.vst
   touch boog.done


#multi4.vst : multi4_o.vst
      loon multi4_o multi4
#loon.done : multi4.vst
  touch loon.done
export MBK_CATAL_LIB=$TARGET_LIB
#res_synth_1.pat : multi4.vst
  asimut multi4 multi4 res_synth_1
#multi4_p.ap : res_synth_1.pat
      ocp -v -gnuplot -ioc multi4 multi4 multi4_p
export METAL_LEVEL=2

#multi4.ap : multi4_p.ap multi4.vst
       nero -V $metal_level -p multi4_p multi4 multi4
export RDS_TECHNO=/home/cadence/alliance/alliance/src/documentation/alliance-examples/etc/techno-035.rds
export RDS_TECHNO_NAME=$RDS_TECHNO
export MBK_IN_LO=al
export MBK_OUT_LO=al
export MBK_CATAL_LIB=$TARGET_LIB
#multi4_e.al : multi4.ap
    cougar -v -ac multi4 multi4_e
export MBK_IN_LO=spi
export MBK_OUT_LO=spi
export SPI_MODEL=$ALLIANCE_TOP//home/cadence/alliance/alliance/src/mbk/etc/spimodel.cfg
export MBK_SPI_MODEL=$SPI_MODEL
export MBK_SPI_ONE_NODE_NORC="true"
export MBK_SPI_NAMEDNODES="true"
#multi4_e.spi : multi4.ap
      cougar -v -ac multi4 multi4_e
export MBK_IN_LO=vst
export MBK_OUT_LO=vst

#lvx.done : multi4.vst multi4_e.al multi4_e.spi
      lvx vst al multi4 multi4_e -f
      touch lvx.done
export RDS_TECHNO_SYMB=/home/cadence/alliance/alliance/src/documentation/alliance-examples/etc/techno-symb.rds
export RDS_TECHNO_NAME=$RDS_TECHNO_SYMB
#druc.done : lvx.done multi4.ap
  druc multi4
  touch druc.done
export RDS_TECHNO_NAME=$RDS_TECHNO
#multi4.cif : druc.done
  s2r -v multi4
  dreal multi4.cif
```

Εικόνα 6.23

Στις γραμμές 20,27,33,62,68 διακρίνουμε τις εντολές που έχουμε προσθέσει και είναι η μόνη διαφορά σε ότι αφορά τις εντολές που έχουμε χρησιμοποιήσει σε όλο το κεφάλαιο 6.

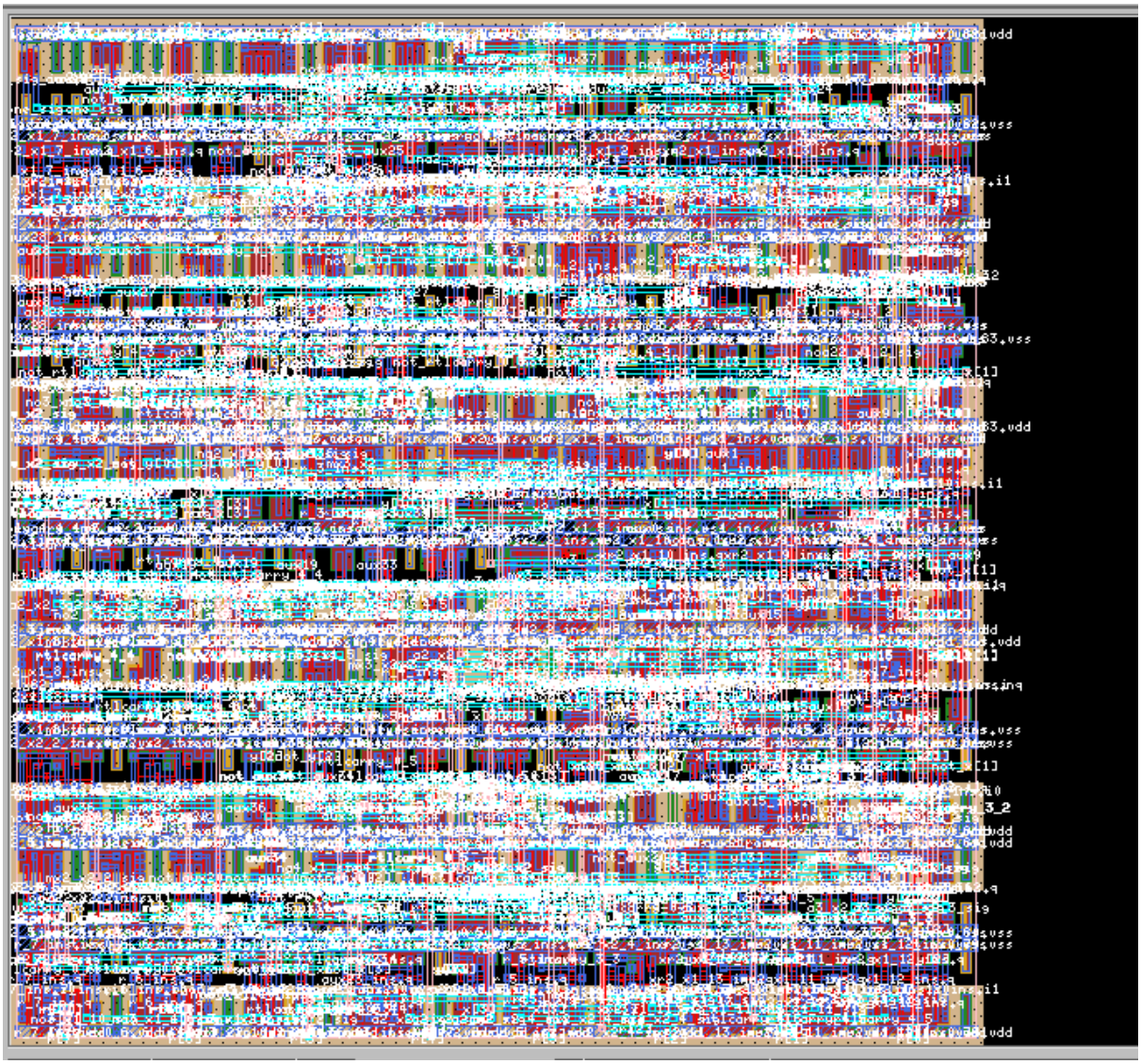
Έχουμε ολοκληρώσει το πρόγραμμα μας και είμαστε έτοιμοι να το τρέξουμε στο τερματικό. Με αυτόν τον τρόπο υλοποιούμε πολύ πιο γρήγορα τον πολλαπλασιαστή μας και έχουμε

την δυνατότητα να επεξεργαζόμαστε τον κώδικα για τυχόν αλλαγές ή βελτιστοποιήσεις που επιθυμούμε στο μέλλον.

Τρέχουμε το πρόγραμμά μας στο τερματικό με την ακόλουθη εντολή:

```
./test.sh
```

Τρέχοντας το πρόγραμμα μπορούμε να διακρίνουμε ότι υλοποιείται κάθε εργαλείο και μας ανοίγει όταν ολοκληρωθεί το εργαλείο απεικόνισης DREAL. Από την μπάρα εργαλείων επιλέγουμε File→Open→multi4.cif και το αποτέλεσμα φαίνεται στην Εικόνα 6.24. Από εκεί και πέρα ακολουθούμε τα βήματα που έχουμε ήδη αναφέρει πιο πάνω.



Εικόνα 6.24

7. ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ ΣΤΗΝ ΣΧΕΔΙΑΣΗ ΟΛΟΚΛΗΡΩΜΕΝΩΝ ΚΥΚΛΩΜΑΤΩΝ

7.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο παρόν κεφάλαιο της πτυχιακής εργασίας παρατίθενται συμπεράσματα στην σχεδίαση ολοκληρωμένων κυκλωμάτων αλλά και της πτυχιακής.

Στο πρώτο μέρος της πτυχιακής μάθαμε για την ιστορία των κυκλωμάτων και πως αυτά εξελίχθηκαν στον χρόνο και ότι είχαν ως αφετηρία το πρώτο τρανζίστορ το οποίο ανακαλύφθηκε το 1947. Μέσα από συνεχή εξέλιξη έχουμε φτάσει στην σημερινή εποχή όπου τα τρανζίστορ έχουν γίνει αισθητά μικρότερα και κατά συνέπεια και οι επεξεργαστές και έχουν αυξηθεί κατακόρυφα η ταχύτητα, η απόδοση και έχει μειωθεί η απαιτούμενη ισχύς.

Σκοπός της πτυχιακής ήταν η υλοποίηση και η επαλήθευση ψηφιακού κυκλώματος και πιο συγκεκριμένα ενός πολλαπλασιαστή 4-bit με στόχο της κατανόηση της σχεδίασης ολοκληρωμένων κυκλωμάτων. Χρησιμοποιήσαμε την πλατφόρμα Alliance και μέσω αυτής μάθαμε την λειτουργία των εργαλείων της και πως αυτά χρησιμοποιούνται και με ποιο τρόπο και ποια σειρά να τα χρησιμοποιούμε ώστε υλοποιούμε τους σχεδιασμούς μας. Επίσης κατανοήσαμε τις τεχνολογίες που χρησιμοποιήσαμε οι οποίες είναι γλώσσες περιγραφής υλικού VHDL και Verilog, RTL σχεδίαση, Κανόνες σχεδίασης αλλά και την ροή που απαιτείται να ακολουθούμε για την σωστή υλοποίηση ενός κυκλώματος.

Αρχικά αναλύσαμε και εξηγήσαμε ένα-ένα όλα τα εργαλεία που διαθέτει το Alliance και είδαμε όλες τις επιλογές που είναι διαθέσιμες για κάθε εργαλείο αντίστοιχα. Χρησιμοποιήσαμε και ένα παράδειγμα ενός πολυπλέκτη για να μας βοηθήσει να κατανοήσουμε σε ένα αρχικό και βασικό στάδιο τα εργαλείο και πως αυτά χρησιμοποιούνται αλλά και να είμαστε σε θέσει να ερμηνεύουμε κάθε έξοδο του κάθε εργαλείου και πως αυτά σταδιακά υλοποιούν ένα ψηφιακό κύκλωμα.

Έχοντας αποκτήσει οικειότητα με το Alliance, τα εργαλεία του καθώς και την ροή υλοποίησης προχωρήσαμε στην υλοποίηση του βασικού μα σχεδιασμού του πολλαπλασιαστή 4-bit. Χρησιμοποιήσαμε 3 βασικά μπλοκ που συνθέτουν τον πολλαπλασιαστή την γεννήτρια μερικών γινομένων, το δέντρο αθροιστών και τον αθροιστή εξόδου. Κάθε μπλοκ υλοποιήθηκε ξεχωριστά και στο τέλος ενώθηκαν όλα μαζί συνθέτοντας έτσι τον τελικό σχεδιασμό μας ο οποίος επαληθεύτηκε με αριθμητικές τιμές και επιπρόσθετα επαληθεύτηκε και ως προς την ορθότητα σχεδίασης του.

Φτάνοντας στο τελικό στάδιο του σχεδιασμού και έχοντας κατανοήσει κάθε μπλοκ που τον απαρτίζει αλλά και κάθε component αυτού υλοποιήσαμε το πραγματικό layout. Ακόμα δημιουργήσαμε ένα πρόγραμμα με όλες τις απαραίτητες εντολές και μεταβλητές περιβάλλοντος για την υλοποίηση του σχεδιασμού με στόχο την ταχύτερη υλοποίηση, πιο εύκολη διόρθωση σε κάποιο σημείο και τυχόν μελλοντικές βελτιώσεις.

7.1 ΠΡΟΤΑΣΕΙΣ ΣΤΗΝ ΣΧΕΔΙΑΣΗ ΚΥΚΛΩΜΑΤΩΝ

Στην τελευταία αυτή ενότητα παρουσιάζονται κάποιες προτάσεις οι οποίες μπορούν να βοηθήσουν στην σχεδίαση και επαλήθευση ψηφιακών κυκλωμάτων. Ο σχεδιασμός ψηφιακών κυκλωμάτων αποτελεί μια αρκετά σύνθετη, δύσκολή και χρονοβόρα διαδικασία καθώς επίσης απαιτεί και πολύ γνώση κάθε εξαρτήματος του κυκλώματος αλλά και πολύ καλή γνώση κάθε βήματος της ροής σχεδίασης από το αρχικό μέχρι και το τελικό στάδιο.

Παρακάτω παρουσιάζονται κάποιες συμβουλές που καλό θα ήταν να λαμβάνονται υπόψιν ειδικά από αρχάριους ως προς την σχεδίαση ψηφιακών κυκλωμάτων χρήστες. Ένα σωστό ψηφιακό κύκλωμα πρέπει να λειτουργεί σε βέλτιστα επίπεδα, να είναι όσο το δυνατών πιο μικρό και λειτουργικό γίνεται αλλά ταυτόχρονα να είναι εύκολα ευανάγνωστο και να επιτρέπει στον εκάστοτε μηχανικό να προβεί σε διορθώσεις και βελτιώσεις.

Κάποιες από αυτές τις προτάσεις είναι:

- Κρατάμε σημειώσεις σε κάθε βήμα, τι δεδομένα έχουμε, που θέλουμε να καταλήξουμε και ποιες είναι οι διαθέσιμες επιλογές που έχουμε ώστε να φτάσουμε στο τελικό στόχο.
- Δημιουργούμε ένα διάγραμμα ροής της διαδικασίας και χωρίζουμε τις διαδικασίες σε επιμέρους διαδικασίες όπως RTL σύνθεση, τοποθέτηση και δρομολόγηση, έλεγχος ορθότητας σχεδιασμού, έλεγχος κανόνων σχεδιασμού και τέλος μετατροπή συμβολικού layout σε πραγματικό.
- Σε κάθε αρχείο που δημιουργούμε τα δίνουμε ονόματα που να σχετίζονται με το συγκεκριμένο βήμα ή τα component που απαρτίζουν το εκάστοτε βήμα του σχεδιασμού μας. Αυτό έχει ως στόχο κάθε στιγμή να ξέρουμε που βρισκόμαστε και να είναι πιο εύκολη η μεταφορά σε κάθε σημείο του σχεδιασμού για τυχόν προσθήκες ή διορθώσεις.
- Όπως είδαμε ένα από τα πιο βασικά στοιχεία της σχεδίασης με προγράμματα όπως το Alliance είναι η πολύ καλή γνώση των εργαλείων πως αυτά εκτελούνται στην γραμμή εντολών αλλά επιπρόσθετα και των επιλογών τους καθώς με βάσει αυτές το εκάστοτε εργαλείο πραγματοποιεί και διαφορετικές λειτουργίες ανάλογο το τι εμείς χρειαζόμαστε.

- Η δημιουργία ενός προγράμματος που θα περιέχει όλα τα απαραίτητα όπως εργαλεία, μεταβλητές περιβάλλοντος εξοικονομεί χρόνο, δίνει μεγαλύτερη ευελιξία στον μηχανικό για τροποποίηση του σχεδιασμού και για μελλοντικές βελτιώσεις. Και ίσως το πιο σημαντικό ευκολότερη και ταχύτερη διόρθωση λαθών με την προϋπόθεση βέβαια όπως ήδη αναφέραμε και πιο πάνω να έχουμε χρησιμοποιήσει τα κατάλληλα ονόματα στα στοιχεία του σχεδιασμού μας.

  Συνοψίζοντας, έχοντας ως οδηγό τις παραπάνω απλές όπως φαίνονται προτάσεις και με καλή γνώση μπορούμε να σχεδιάσουμε ψηφιακά κυκλώματα αρκετά υψηλού επιπέδου.[16][17]

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΕΛΛΗΝΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ολοκληρωμένο Κύκλωμα. Ανακτήθηκε από διαδικτυακό τόπο στις 10/10/2020. https://el.wikipedia.org/wiki/%CE%9F%CE%BB%CE%BF%CE%BA%CE%BB%CE%B7%CF%81%CF%89%CE%BC%CE%AD%CE%BD%CE%BF_%CE%BA%CF%8D%CE%BA%CE%BB%CF%89%CE%BC%CE%B1#cite_note-%CF%88%CE%B7%CF%86%CE%B9%CE%B1%CE%BA%CE%AE_%CF%83%CF%87%CE%B5%CE%B4%CE%AF%CE%B1%CF%83%CE%B7-2

[2] ΕΥΑΓΓΕΛΟΣ ΚΥΡΙΤΣΗΣ. Σχεδίαση και Υλοποίηση Φίλτρων FIR Βασισμένων στον Αλγόριθμο Karatsuba. Ανακτήθηκε από διαδικτυακό τόπο στις 10/10/2020. https://core.ac.uk/download/pdf/45649529.pdf

[3] Λελιγκού Ελένη Αικατερίνη, Βολιότης Σταμάτης, Καραρούντας Αθανάσιος. Η ΛΟΓΙΚΗ ΣΧΕΔΙΑΣΗ ΣΤΟ ΕΡΓΑΣΤΗΡΙΟ. Ανακτήθηκε από διαδικτυακό τόπο στις 11/10/2020. https://repository.kallipos.gr/handle/11419/6440?locale=en

[4] ΣΑΝ ΣΗΜΕΡΑ ΣΤΟΥΣ ΥΠΟΛΟΓΙΣΤΕΣ. 23/12/1947 | ΤΡΑΝΖΙΣΤΟΡ. Ανακτήθηκε από διαδικτυακό τόπο στις 13/10/2020. https://sansimeracomputers.wordpress.com/2012/12/23/dec23-1947/

[5] Neil H. E. Weste, David M. Harris. Σχεδίαση Ολοκληρωμένων Κυκλωμάτων CMOS VLSI. 2011, Εκδόσεις Παπασωτηρίου.

ΞΕΝΟΓΛΩΣΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

[6] Very Large Scale Integration. Ανακτήθηκε από διαδικτυακό τόπο στις 13/10/2020. https://en.wikipedia.org/wiki/Very_Large_Scale_Integration

[7] Y. Leblebici. DESICN OF VLSI SYSTEMS CHAPTER 1 INTRODUCTION TO VLSI SYSTEMS. Ανακτήθηκε από διαδικτυακό τόπο στις 13/10/2020. http://emicroelectronics.free.fr/onlineCourses/VLSI/ch01.html#1.2

[8] George Gustav Savii. Integrated Circuit Design. Ανακτήθηκε από διαδικτυακό τόπο στις 13/10/2020. https://www.sciencedirect.com/topics/computer-science/integrated-circuit-design

[9] Jean-Jacques Delisle. What is Digital IC Design?. Ανακτήθηκε από διαδικτυακό τόπο στις 19/10/2020. https://www.allaboutcircuits.com/technical-articles/what-is-digital-ic-design/

[10] K. Sripath Roy, K. Abhiram, M. Arun Sumanth, Jaishree Jaishankar, P. Abhishek, B. Nikhil Prabhat, L. Gnana Teja. Development of graphical user interface for open source VLSI digital synthesis tool Qflow. Ανακτήθηκε από διαδικτυακό τόπο στις 19/10/2020. file:///C:/Users/alexd/Downloads/IJET-12649%20(1).pdf

[11] Qflow 1.3: An Open-Source Digital Synthesis Flow. Ανακτήθηκε από διαδικτυακό τόπο στις 19/10/2020. http://opencircuitdesign.com/qflow/

[12] Icarus Verilog. Ανακτήθηκε από διαδικτυακό τόπο στις 25/10/2020. http://iverilog.icarus.com/

[13] Icarus Verilog From Wikipedia. Ανακτήθηκε από διαδικτυακό τόπο στις 25/10/2020. https://en.wikipedia.org/wiki/Icarus_Verilog

[14] Carlos Silva Cardenas, Takeo Yoshida, Alberto Palacios Pawlovsky 18 August 2006. Introduction to VLSI CMOS Circuits Design. Ανακτήθηκε από διαδικτυακό τόπο στις 4/3/2020

[15] Wallace tree From Wikipedia. Ανακτήθηκε από διαδικτυακό τόπο στις 2/11/2020. https://en.wikipedia.org/wiki/Wallace_tree

[16] 8 Circuit Design Tips by Nichole Heydenburg, 26/ 9/ 2018. Ανακτήθηκε από το διαδικτυακό τόπο στις 14/11/2020. https://www.eeweb.com/8-circuit-design-tips/

[17] Logic Unused Gate Termination. Ανακτήθηκε από διαδικτυακό τόπο στις 14/11/2020. https://www.electronics-notes.com/articles/digital-embedded-processing/logic-circuits-design/design-guidelines-unused-gate-input-termination.php
[18] VLSI Lab Tutorial 3. Ανακτήθηκε από διαδικτυακό τόπο στις 05/12/2020. http://unixlab.sfsu.edu/~necrl/files/cadence%20tutorials/Cadence%20Virtuso%20Layout%20Editor.pdf
[19] Kyusun Choi. CMPEN 411 VLSI Digital Circuits,Lecture 05: IC Manufacturing. Ανακτήθηκε από διαδικτυακό τόπο στις 05/12/2020. http://www.cse.psu.edu/~kxc104/class/cmpen411/15s/lec/C411L05fabrication.pdf
[20] BY ADMINISTRATOR 29 JUNE 2015. Binary Multiplication Methods. Ανακτήθηκε από διαδικτυακό τόπο στις 6/12/2020. https://www.electronicshub.org/binary-multiplication/
[21] Moore's Law From Wikipedia. Ανακτήθηκε από διαδικτυακό τόπο στις 15/01/2021. https://en.wikipedia.org/wiki/Moore%27s_law

ΒΙΒΛΙΟΓΡΑΦΙΑ ΕΙΚΟΝΩΝ

[1] Πρώτο τρανζίστορ. Ανακτήθηκε από διαδικτυακό τόπο στις 10/10/2020. https://www.noesis.edu.gr/%CE%B5%CF%80%CE%B9%CF%83%CF%84%CE%AE%CE%BC%CE%B7-%CE%BA%CE%B1%CE%B9-%CF%84%CE%B5%CF%87%CE%BD%CE%BF%CE%BB%CE%BF%CE%B3%CE%AF%CE%B1/%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CF%84%CE%AD%CF%82/%CF%84%CE%B5%CF%87%CE%BD%CE%BF%CE%BB%CE%BF%CE%B3%CE%AF%CE%B1/%CF%84%CF%81%CE%B1%CE%BD%CE%B6%CE%AF%CF%83%CF%84%CE%BF%CF%81/
[2] Το πρώτο ολοκληρωμένο κύκλωμα. Ανακτήθηκε από διαδικτυακό τόπο στις 11/10/2020. http://users.sch.gr/pagiats/Electronics/first_IC.htm
[3] Moore's Law From Wikipedia. Ανακτήθηκε από διαδικτυακό τόπο στις 15/01/2021.https://en.wikipedia.org/wiki/Moore%27s_law
[4] Jean-Jacques Delisle. What is Digital IC Design?. Ανακτήθηκε από διαδικτυακό τόπο στις 19/10/2020. https://www.allaboutcircuits.com/technical-articles/what-is-digital-ic-design/
[5] Carlos Silva Cardenas, Takeo Yoshida, Alberto Palacios Pawlovsky 18 August 2006. Introduction to VLSI CMOS Circuits Design. Ανακτήθηκε από διαδικτυακό τόπο στις 4/3/2020.
[6] Selective Non-Default Rules Based Clock Tree Synthesis using open-source EDA. Ανακτήθηκε από διαδικτυακό τόπο στις 15/10/2020.

https://www.vlsisystemdesign.com/selective-non-default-rules-based-clock-tree-synthesis-using-open-source-eda/

[7] A Verilog simulation and synthesis tool that operates as a compiler. Ανακτήθηκε από διαδικτυακό τόπο στις 17/10/2020. https://icarus-verilog.software.informer.com/0.8/