Τ.Ε.Ι. Κρήτης
Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων

Πτυχιακή Εργασία

Ανάπτυξη υποστηρικτικού λογισμικού για Network on Chip

Μοτάκης Αντώνιος
Α.Μ. 899

Επιβλέπων καθηγητής:
Κορνάρος Γεώργιος

15 Απριλίου 2011

Ευχαριστίες

Η εργασία αυτή κατέστη δυνατή χάρη στην συμβολή του επιβλέποντα καθηγητή μου Κορνάρο Γεώργιο, και στην ομάδα ανάπτυξης του *Spidergon STNoC* στην *STMicroelectronics*, Marcello Coppola, Riccardo Locatelli, Valerio Catalano, Michael Soulié, Giuseppe Maruccia, Florentine Dubois, και Lorenzo Pieralisi.

Abstract

As the number of components embedded on a piece of silicon increases, so are the bandwidth requirements of a System on Chip. As traditional interconnects no longer meet current and future requirements in terms of performance, scalability, and energy efficiency, Systems on Chip move towards exploitation of Networks on Chip, which are based on network topologies [1].

Spidergon STNoC is such a Network on Chip developed by the AST labs of the company STMicroelectronics. It supports many topologies which can be designed in a graphical environment thanks to its architecture based on simple, non-programmable routers, and flexible Network Interfaces which function as access points to the interconnect, and provides programmable services such as changing the routing information, quality of service (QoS), and security [1].

The purpose of this work is to develop the necessary supporting software that allows a system designer to exploit the dynamic features of *Spidergon STNoC*. This includes the Java plugin necessary for the design environment of *Spidergon STNoC*, *iNoC*, which will generate a driver for a Linux kernel. The driver provides a userspace interface to the programmable services provided by the designed Network on Chip, and general information about it. In addition, a portable C library uses the interfaces of the driver to provide a homogeneous C interface to applications and higher level functionality, such as determining appropriate QoS parameters based on specific requirements on bandwidth.

Finally, this software ecosystem will be presented running with tools & examples on typical Linux environments (ARM and x86) and Android.

Σύνοψη

Καθώς ο αριθμός των components που ενσωματώνονται σε ένα κομμάτι πυριτίου αυξάνεται συνεχώς, και οι απαιτήσεις σε bandwidth σε ένα System on Chip αυξάνονται. Καθώς τα παραδοσιακά interconnects τύπου δίαυλου δεν ανταποκρίνονται πια στις σύγχρονες και μελλοντικές απαιτήσεις από άποψης επιδόσεων, scalability, και ενεργειακής απόδοσης, τα System on Chip κινούνται πλέον προς την εκμετάλλευση των Network on Chip, τα οποία βασίζονται σε τοπολογίες δικτύου [1].

Το *Spidergon STNoC* είναι ένα τέτοιο Network on Chip που αναπτύσσεται από τα εργαστήρια AST της εταιρείας STMicroelectronics. Υποστηρίζει πληθώρα τοπολογιών, οι οποίες μπορούν να σχεδιαστούν σε γραφικό περιβάλλον, μέσω της αρχιτεκτονικής του που βασίζεται σε απλά, μη προγραμματιζόμενα routers, αλλά και ευέλικτα Network Interface που αποτελούν σημεία πρόσβασης στο interconnect, και παρέχουν προγραμματιζόμενα services όπως αλλαγή του routing, quality of service (QoS), και security [1].

Σκοπός της εργασίας αυτής είναι η ανάπτυξη του απαραίτητου υποστηρικτικού software που θα επιτρέπει στον σχεδιαστή ενός συστήματος να εκμεταλλευτεί την δυναμικότητα του *Spidergon STNoC*. Αυτό περιλαμβάνει το απαραίτητο Java plugin για το γραφικό περιβάλλον σχεδίασης του *Spidergon STNoC*, *iNoC*, που θα κάνει generate έναν driver για περιβάλλον linux. Ο driver θα παρέχει ένα userspace interface στα προγραμματιζόμενα services του σχεδιασμένου Network on Chip, και γενικές πληροφορίες για αυτό. Επιπλέον, μία portable C βιβλιοθήκη, θα χρησιμοποιεί τα interfaces του driver για να παρέχει ένα ομογενές C interface στις εφαρμογές, με υψηλότερου επιπέδου λειτουργικότητα, όπως ο καθορισμός των κατάλληλων QoS παραμέτρων με βάση συγκεκριμένων απαιτήσεων σε bandwidth.

Τέλος, αυτό το software οικοσύστημα, θα παρουσιαστεί να υποστηρίζει εργαλεία & παραδείγματα σε περιβάλλον Linux (ARM και x86) και Android.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος των registers

Listings

1 Εισαγωγή

Σε αυτή την εργασία, αναπτύξαμε το απαραίτητο βοηθητικό λογισμικό, έτσι ώστε το λογισμικό ενός System on Chip που χρησιμοποιεί το *Spidergon STNoC* interconnect, να μπορεί να προγραμματίζει κατά τον χρόνο εκτέλεσης του συστήματος τα χαρακτηριστικά του interconnect. Η λύση που αναπτύχθηκε περιλαμβάνει έναν Linux kernel driver και μια user-space βιβλιοθήκη που εκθέτει τη λειτουργικότητα των υπηρεσιών του *Spidergon STNoC* μέσω ενός απλού, ανεξαρτήτου λειτουργικού, portable C API. Το λογισμικό του συστήματος δεν χρειάζεται εκτεταμένες αλλαγές τόσο στο kernel space τόσο και στο user space κώδικα για να επωφεληθούν από τις δυνατότητες run time επανάπρογραμματισμού του Network on Chip· μικρές μη διεισδυτικές προσθήκες στον κώδικα μιας εφαρμογής αρκούν για να αποκτήσει τον πλήρη έλεγχο του υλικού και να προσφέρει τη βέλτιστη εμπειρία τελικού χρήστη. Επιπλέον, το ανεξάρτητου συστήματος C API που χρησιμοποιείται επιτρέπει στον κώδικα να διατηρεί ένα επίπεδο αποσύνδεσης από κατώτερα στρώματα του stack υλικού/λογισμικού, κάτι το οποίο σημαίνει η δυνατότητα επαναχρησιμοποίησης κώδικα δεν θυσιάζεται.

Επιπροσθέτως, ένα υψηλότερου επιπέδου software component για την επαναρύθμιση του Network on Chip μας επιτρέπει να παρέχουμε πιο πολύπλοκες λειτουργίες οικοδομημένα πάνω στα primitives που παρέχονται από το υλικό. Ο σχεδιαστής του λογισμικού ενός συστήματος, αντί να αφιερώσει το χρόνο του για να καθοριστεί τις σωστές τιμές για το *QoS* μιας εφαρμογής, μπορεί να χρησιμοποιήσει τις λειτουργίες υψηλού επιπέδου που υπολογίζουν τις σωστές ρυθμίσεις για ένα δεδομένο σύνολο απαιτήσεων. Με αυτόν τον τρόπο, ο σχεδιαστής μπορεί να διατηρήσει ακόμη περισσότερο το επίπεδο της αποσύνδεσης, ακόμα και να ξαναχρησιμοποιήσει τον ίδιο κώδικα για πολλαπλά συστήματα με διαφορετικές *Spidergon STNoC* υλοποιήσεις. Επιπλέον, στις υψηλού επιπέδου λειτουργίες περιλαμβάνεται η δυνατότητα αποθήκευσης των τρέχων ρυθμίσεων του Network on Chip σε *profiles* που μπορούν να εναλλάσσονται κατά το χρόνο εκτέλεσης, καθώς επίσης και εργαλεία διαγνωστικά και debugging.

Το λογισμικό μας αναπτύχθηκε και λειτουργεί σε περιβάλλοντα Linux και Android, και περιλαμβάνει εκτός από τον απαραίτητο driver και βιβλιοθήκη, και τις ανάλογες εφαρμογές διαχείρισης του Network on Chip που επιδεικνύουν την λειτουργικότητα του συστήματος μας.

1.1 Κίνητρο για την διεξαγωγή της εργασίας

Η ενσωμάτωση δεκάδων ή και εκατοντάδων cores σε ένα μόνο chip έχει επιτρέψει να προκύψουν multi-core Systems on Chip είτε με τη μορφή των chip multiprocessors (CMPs) ή, γενικότερα, custom-made για ορισμένες εφαρμογές. Καθώς τα multi-core συστήματα γίνονται όλο και πιο απαιτητικά από άποψης επικοινωνίας, χρησιμοποιούνται application specific Networks on Chip (NoCs), τα οποία έχουν διερευνηθεί διεξοδικά [2], και προσαρμόζονται συνήθως κατά το χρόνο σχεδιασμού. Ωστόσο, μια δυναμική προσέγγιση είναι ως επί το πλείστον επιθυμητή στη βελτίωση της ενεργειακής απόδοσης του συστήματος, προσαρμόζοντας μείζονες πόρους του System on Chip, όπως buffering, ή virtual channels, ώστε να ταιριάζουν καλύτερα με διαφορετικές ανάγκες κατά τη διάρκεια της εκτέλεσης. Επιπλέον, οι αναδυόμενες virtualization τεχνολογίες απαιτούν μηχανισμούς για τη δυναμική προσαρμογή των πόρων του συστήματος.

Σύγχρονες λύσεις interconnect, όπως το *Spidergon STNoC* [1] παρέχουν όλο και περισσότερες επιλογές και ρυθμίσεις. Αυτές περιλαμβάνουν QoS παραμέτρους προκειμένου να βελτιστοποιηθεί η απόδοση εφαρμογών καθώς το σύστημα λειτουργεί, ή να επιτύχουν καλύτερη διαχείριση ενέργειας σε επίπεδο συστήματος, και υπηρεσίες security, φιλτράροντας την κυκλοφορία παρόμοια με ένα firewall. Η επαναρύθμιση του routing μπορεί επιπροσθέτως να χρησιμοποιηθεί για να επιτευχθεί καλύτερη αξιοποίηση του Network on Chip ή για λόγους fault tolerance (π.χ. να αποφευχθούν κατεστραμμένες συνδέσεις), ή για την εξάλειψη παρεμβολών της κυκλοφορίας.

Τυπικά, τα σύγχρονα Networks on Chip περιλαμβάνουν Network Interface στα όρια τους, τα οποία διαχωρίζουν την κυκλοφορία σε πακέτα που διαβιβάζονται μέσω ενός stateless δικτύου από routers. Αυτή η οργάνωση επιτρέπει την ανάπτυξη ενός σύνολου υπηρεσιών στα Network Interface που ανταποκρίνονται στις απαιτήσεις των NoC-based multicore συστημάτων.

Η πλήρης εκμετάλλευση αυτού του βαθμού ευελιξίας κατά το run time απαιτεί ένα υψηλότερο βαθμό integration του λογισμικού με το υλικό. Ο προγραμματισμός των υπηρεσιών του *Spidergon STNoC* είναι εφικτός άμεσα (bare metal access), ωστόσο, ένα abstraction layer μπορεί να μειώσει την πολυπλοκότητα υλοποίησης. Επιπλέον, διαφορετικά συστήματα που στοχεύουν σε διαφορετικές αγορές μπορεί να συνδυάζουν διαφορετικούς συνδυασμούς λειτουργικών συστημάτων, πλατφορμών και εφαρμογών, που μπορούν να καταστήσουν τον προγραμματισμό των υπηρεσιών του Network on Chip για κάθε διαφορετικό σύστημα μη πρακτικό και χρονοβόρο για τον σχεδιαστή του λογισμικού του συστήματος.

1.2 Σκοπός και Στόχοι Εργασίας

Οι στόχοι αυτής της εργασίας είναι οι εξής:

- Ανάπτυξη Linux driver για τον επαναπρογραμματισμό του *Spidergon STNoC*.

- Ανάπτυξη βιβλιοθήκης που επιτρέπει τον επαναπρογραμματισμό του *Spidergon STNoC*, χωρίς να χρειαστεί να χρησιμοποιηθεί άμεσα ο driver και να χρειάζεται γνώση του χαμηλού επιπέδου τρόπο προγραμματισμού του *Spidergon STNoC*.
- Υψηλού επιπέδου λειτουργικότητα, όπως ο υπολογισμός των κατάλληλων ρυθμίσεων *QoS* για τις απαιτήσεις μιας εφαρμογής.
- Παρουσίαση εργαλείων που επιδεικνύουν αυτή την λειτουργικότητα σε Linux και Android.

1.3 Δομή Εργασίας

Το υπόλοιπο αυτής της εργασίας είναι οργανωμένο ως εξής:

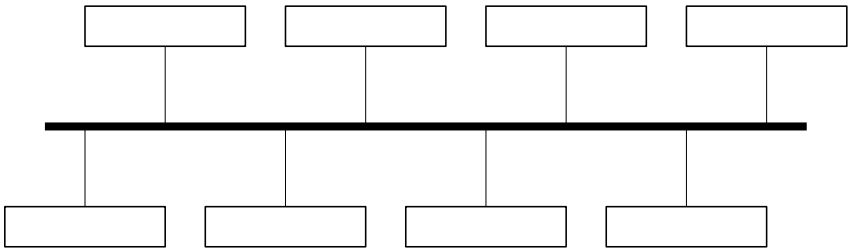
- Στο κεφάλαιο 2 δίνουμε μια σύντομη εισαγωγή πάνω στις τεχνολογίες στις οποίες υλοποιείται η εργασίας μας: η τεχνολογία Network on Chip, το *Spidergon STNoC*, και οι υπηρεσίες που προσφέρει.
- Στο κεφάλαιο 3 αναλύουμε τις απαιτήσεις του λογισμικού κληθήκαμε να αναπτύξουμε.
- Στο κεφάλαιο 4 περιγράφουμε την αρχιτεκτονική και τρόπο υλοποίησης των βασικών στοιχείων του λογισμικού μας: ο Linux driver, με το ανάλογο plugin για το iNoC, και η βιβλιοθήκη *libstnoc*.
- Στο κεφάλαιο 5 περιγράφουμε την αρχιτεκτονική και τον τρόπο υλοποίησης των εργαλείων που αναπτύξαμε και χρησιμοποιούν την βιβλιοθήκη *libstnoc*: το εργαλείο *stnoctl*, η υποστήριξη για Android και το ανάλογο GUI.
- Το κεφάλαιο 6 περιγράφει τον τρόπο χρήσης του λογισμικού που αναπτύξαμε, οδηγίες εγκατάστασης και τεκμηρίωση των APIs και εργαλείων.
- Τέλος στο κεφάλαιο 7 δίνουμε τα συμπεράσματα μας και κατευθύνσεις για μελλοντική ανάπτυξη.

2 Γενική εισαγωγή στα Network on Chip και στο Spidergon

Αν και το αντικείμενο της εργασίας αυτής δεν είναι η ανάπτυξη των NoC καθαυτή, η κατανόηση του περιβάλλοντος το οποίο θα υποστηρίζει κάποιο λογισμικό είναι απαραίτητη για την σκιαγράφηση του «problem space» που εξυπηρετεί αυτό το λογισμικό να αντιμετωπίσει. Έτσι αυτό το κεφάλαιο εξυπηρετεί τον ρόλο μιας εισαγωγής στις τεχνολογίες τις οποίες το λογισμικό μας συμπληρώνει και υποστηρίζει.

2.1 Τα παραδοσιακά interconnects

Αρχικά στα SoCs, η ανάγκη για την διασύνδεση μεταξύ των λογικών components υλοποιημένων πάνω σε ένα κομμάτι πυριτίου εξυπηρετήθηκε από λύσεις τύπου bus, οι οποίες αποτελούνται ουσιαστικά από κάποια ομάδα «καλωδίων» (σχήμα 2.1). Σε αυτές του τύπου υλοποιήσεις, πολλαπλοί χρήστες master ή slave συνδέονται πάνω στο bus, και ανά πάσα στιγμή ένας arbiter καθορίζει ποιος χρήστης θα μεταδίδει δεδομένα και προς τα ποιον.[3]

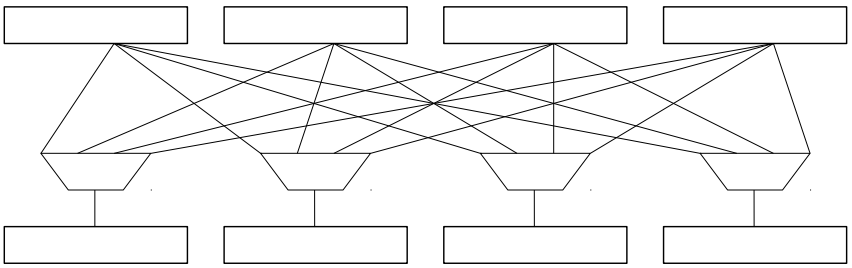


Σχήμα 2.1: Η διασύνδεση bus όπου τα IP Cores μοιράζονται έναν σύνδεσμο για την επικοινωνία τους

Οι περιορισμοί αυτής της προσέγγισης είναι αρκετοί, για παράδειγμα το bandwitdh του bus μοιράζεται απο τους χρήστες του, και ο σχεδιαστής του interconnect καλείται να ισορροπήσει διαφορετικούς παράγοντες λειτουργίας· το μήκος του bus περιορίζει την συχνότητα λειτουργίας, η οποία είναι ανάλογη με το bandwidth που μπορεί να υποστηρίξει το interconnect. Τεχνικές pipelining μπορούν να χρησιμοποιηθούν, κάτι το οποίο εισάγει άλλες δυσκολίες και επηρεάζει το latency κατά μη τετριμμένο τρόπο. Επιπλέον η ανάγκη για υποστήριξη διαφορετικών αναγκών από άποψης latency και bandwidth οδήγησε σε λύσεις όπου πολλαπλές ιεραρχίες απο buses εξυπηρετούν διαφορετικούς τύπους

κίνησης.

Αντιμετωπίζοντας όλο και αυξανόμενες ανάγκες από τα IP που καλούνται να εξυπηρετήσουν όλο και ποιο απαιτητικούς χρήστες, δομές τύπου crossbar (σχήμα 2.2) άρχιζαν να εμφανίζονται στα interconnects, αυξάνοντας το συνολικό bandwidth και την αξιοπιστία του συστήματος από άποψης latency, χρησιμοποιόντας όμως σημαντικά αυξημένο αριθμό wires στο τσίπ, αφού πλέον κάθε master μπορεί να συνδεθεί ανεξάρτητα με κάθε slave.



Σχήμα 2.2: Η διασύνδεση τύπου crossbar

Σύγχρονες λύσεις όπως το AMBA AXI της ARM [4] καθορίζουν το interface ενός IP με το interconnect, αποκρύπτοντας την αρχιτεκτονική του και την φιλοσοφία λειτουργίας με τα οποία έχει σχεδιαστεί. Όμως οι σύγχρονες απαιτήσεις ενός ενσωματωμένου συστήματος πάνω σε ένα System on Chip κάνουν εξαιρετικά περίπλοκη εργασία να σχεδιαστεί ένα interconnect που να αντιμετωπίζει όλους τους προαναφερθέν περιορισμούς, δεδομένου ήδη υπαρχόντων περιορισμών όπως για παράδειγμα όσον αφορά το layout των components ενός τσιπ. Έτσι είναι απαραίτητο το πρόβλημα της διασύνδεσης των διαφόρων IPs να αντιμετωπιστεί με πιο σύγχρονες μεθόδους.

2.2 Τα Network on Chip

Σήμερα οι σχεδιαστές των System on Chip αντιμετωπίζουν σωρεία προκλήσεων κατά την σχεδίαση του interconnect σε ένα τσιπ, οι οποίες δεν περιορίζονται μονάχα στις ολοένα αυξανόμενες απαιτήσεις των IP Cores για επιδόσεις, αλλά περιλαμβάνουν έναν αριθμό απαιτήσεων σε διαχείριση ενέργειας, πολλαπλά clock domains σε διαφορετικά «isles» του τσιπ, διαχείριση σφαλμάτων και ασφαλείας. Οργανώνοντας την φυσική διάταξη του τσιπ, αυτές οι απαιτήσεις μπορεί να προκαλέσουν δυσκολίες, έρχομενες παράλληλα σε σύγκρουση και με απαιτήσεις χώρου wiring, συχνοτήτων, και τάσεων [1].

Το κλειδί για την αντιμετώπιση αυτών των θεμάτων στο interconnect είναι να αποσυνδεθεί το Transport Layer από το Physical Layer. Αυτό επιτυγχάνεται σχεδιάζοντας το on-chip interonnect βασιζόμενο σε ένα packet based transport πρωτόκολλο, όπου τα αιτήματα (και απαντήσεις) των IP Cores έχουν μορφοποιηθεί σε απλά πακέτα, που αποτελούνται από μια επικεφαλίδα, και ωφέλιμο φορτίο, και μεταφέρονται στο interconnect μέσω πολλαπλών απλών ενδιάμεσων συνδέσεων. Η αποσύνδεση μεταξύ του Transport

Layer και το Physical Layer επιτυγχάνεται με δύο τρόπους: λογικά, διότι τα πακέτα σε τοπικό επίπεδο μεταφέρονται μέσω φυσικών συνδέσεων, με τη κατάλληλη διαμόρφωση για κάθε ενδιάμεση σύνδεση· και, φυσικά, επειδή οι φυσικές ιδιότητες του κάθε συνδέσμου (πλάτος, συχνότητα) μπορεί να επιλεγεί σύμφωνα με τις απαιτήσεις της σύνδεσης σε bandwidth και latency [3].

Χρησιμοποιώντας την τεχνολογία Network on Chip, το interconnect αποτελείται από έναν αριθμό στοιχείων, switches και συνδέσμους σε μια καθορισμένη από τον σχεδιαστή τοπολογία. Αυτή η πραγματική Network on Chip αρχιτεκτονική έχει πολλά πλεονεκτήματα σε σχέση με τις προσεγγίσεις τύπου bus και crossbar· είναι εξαιρετικά επεκτάσιμη και ευέλικτη. Σε αντίθεση με αυτές τις παραδοσιακές λύσεις, όπου ο αρχιτέκτονας του συστήματος είναι αναγκασμένος να χαρτογραφήσει την αρχιτεκτονική του συστήματος με μορφή buses/crossbars, χρησιμοποιώντας την προσέγγιση Network on Chip, το interconnect μπορεί να κατασκευαστεί με μία καθορισμένη από το χρήστη τοπολογία που ταιριάζει με το όραμα του σχεδιαστή ως προς την βέλτιστη αρχιτεκτονική.

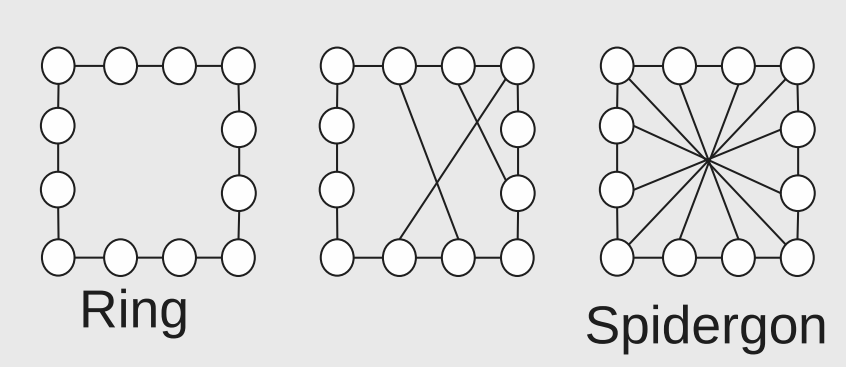
Η αρχιτεκτονική των Network on Chip επιτρέπει μια ορθογώνια προσέγγιση στις προκλήσεις του σχεδιασμού ενός System on Chip επιτρέποντας στον σχεδιαστή να ακολουθήσει μια προσέγγιση όπου οι απαιτήσεις αντιμετωπίζονται ανεξάρτητα. Για παράδειγμα, ένα Network on Chip μπορεί να κατασκευαστεί πρώτα με μια τοπολογία βασισμένη στις απαιτήσεις συνδεσιμότητας, QoS, καθώς και χρονισμού, και στη συνέχεια να βελτιστοποιήσει τους μεμονωμένους συνδέσμους, επιλέγοντας τα βέλτιστα πλάτη και χαρακτηριστικά βάση των αναγκών σε latency και bandwidth.

Λόγω της κομψής αποσύνδεσης του Transport Layer από το Physical Layer, πολύ υψηλές συχνότητες ρολογιού μπορούν να επιτευχθούν. Η πολυπλοκότητα από άποψης λογικών πυλών είναι μειωμένη, δεδομένου ότι κατά τη διάρκεια της μεταφοράς του πακέτου, το περιεχόμενο του φορτίου δεν χρειάζεται να αξιολογηθεί ή να μετατραπεί. Για παράδειγμα, ένας μετατροπέας πλάτους από 64 bit σε 32 bit απλά διαχωρίζει κάθε word των 64 bits σε δύο words των 32-bit, χωρίς ποτέ να εξετάσει το περιεχόμενο του πακέτου (ως word ορίζεται το πλάτος των δεδομένων που μεταφέρεται πάνω από μια σύνδεση κατά τη διάρκεια ενός κύκλου ρολογιού). Δεδομένου ότι οι φυσικές ιδιότητες του κάθε συνδέσμου έχουν ρυθμιστεί αποκλειστικά με βάση τις ειδικές απαιτήσεις του συνδέσμου αυτού, και ανεξάρτητα από άλλες συνδέσεις, ο συνολικός αριθμός των συρμάτων στο interconnect μειώνεται σημαντικά (έως 40%).

Ενώ η ιδέα ενός Network on Chip για το interconnect ενός System on Chip μπορεί να εξακολουθεί να φαίνεται φουτουριστική, η τεχνολογία υιοθετείται ραγδαία απο ένα ευρύ φάσμα Systems on Chip όλων των επιπέδων πολυπλοκότητας. Ο λόγος είναι απλός: Network on Chip υλοποιήσεις των System on Chip interconnects αποδεικνύονται σημαντικά μικρότερες, πιο γρήγορες και πιο ενεργειακά αποδοτικές σε σχέση με παραδοσιακές λύσεις τύπου bus ή crossbar.

2.3 Το Spidergon STNoC και τα services του

Το *Spidergon STNoC* προσφέρει ένα ευρύ φάσμα υπηρεσιών που μπορούν να παραμετροποιηθούν κατά το χρόνο εκτέλεσης· αυτές οι προγραμματιζόμενες παράμετροι, εκτίθενται μέσω μιας σειράς από memory-mapped registers. Οι υπηρεσίες αυτές περιλαμβάνουν επιλογές για τη διαχείριση ενέργειας, το routing, Quality of Service, και την ασφάλεια. [1]



Σχήμα 2.3: Η οικογένεια τοπολογιών *Spidergon STNoC*

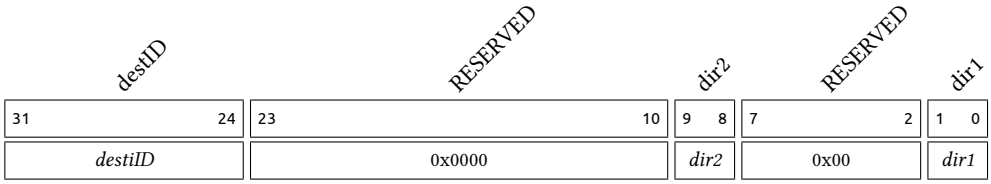
Ένα σύστημα *Spidergon STNoC* μπορεί να διαμορφωθεί κατά τη φάση της σχεδίασης για τη χρήση δύο εναλλακτικών πρωτοκόλλων δρομολόγησης· *Spidergon Routing* και *Source Routing*. Η πρώτη εφαρμόζεται για την οικογένεια τοπολογιών *Spidergon* που αποτελούν επέκταση των τοπολογιών δακτυλίου (σχήμα 2.3)· κάθε κόμβος συνδέεται και στον κόμβο απέναντι του στον δακτύλιο, επιπλέον των κόμβων στα αριστερά και δεξιά του. Εκτός από τους αριστερούς, δεξιούς, και απέναντι συνδέσμους, το *Spidergon Routing* επίσης υποστηρίζει έναν πάνω σύνδεσμο που οδηγεί σε ιεραρχικές *Spidergon* τοπολογίες.

Σε αυτή την οικογένεια τοπολογιών, η δρομολόγηση εκφράζεται με κατευθυντήριο τρόπο. Ένα πεδίο των 2 bits *dir1*, χρησιμοποιείται για την κωδικοποίηση τεσσάρων καταστάσεων: *Left*, *Right*, *Left Acrross* και *Right Accross*. Ένα δεύτερο πεδίο (*dir2*) χρησιμοποιείται όταν χρησιμοποιείται μια ιεραρχική τοπολογία, και τέλος, ένα πεδίο προορισμού των οκτώ bit (*destID*) καθορίζει τον τελικό προορισμό. Αυτά τα πεδία ενσωματώνονται από το Network Interface στην επικεφαλίδα των πακέτων κίνησης πριν εισαχθούν στο Network on Chip, τηρώντας μια φιλοσοφία υλοποίησης υπηρεσιών στα άκρα του δικτύου, χωρίς να υπάρχει υλοποιημένη κάποια ιδιαίτερη λογική στο εσωτερικό και στα routers.

Μπορούμε να επανάπρογραμματίσουμε κατά τον χρόνο εκτέλεσης την δρομολόγηση προς κάποιον προορισμό, γράφοντας σε μιας σειρά από registers που διαθέτει το Network Interface πηγή. Για κάθε πιθανό Network Interface προορισμού, εκτίθεται και ένας προγραμματιζόμενος register με τις παραμέτρους της δρομολόγησης (Register 2.1).

Στο *Source Routing*, ο ίδιος χώρος στην επικεφαλίδα του πακέτου χρησιμοποιείται με διαφορετικό τρόπο, και κωδικοποιεί το *routing path*, δηλαδή ποιον σύνδεσμο πρέπει να

Register 2.1: Register για τον καθορισμό του routing σε *Spidergon Routing* συστήματα



dir1 Καθορίζει την κατεύθυνση που θα ακολουθήσουν τα πακέτα για αυτόν τον προορισμό.

Η default τιμή *dir1* είναι καθορισμένη κατά τον χρόνο σχεδιασμού.

dir2 Καθορίζει την κατεύθυνση που θα ακολουθήσουν τα πακέτα για αυτόν τον προορισμό, μετά από χρήση ενός ιεραρχικού link.

Η default τιμή *dir2* είναι καθορισμένη κατά τον χρόνο σχεδιασμού.

destID Το destID του προορισμού, που τερματίζει την διάδοση του πακέτου μέσα στο Network on Chip, όταν αυτό φτάσει στο Network Interface για το οποίο προορίζεται.

Η default τιμή *destID* είναι καθορισμένη κατά τον χρόνο σχεδιασμού.

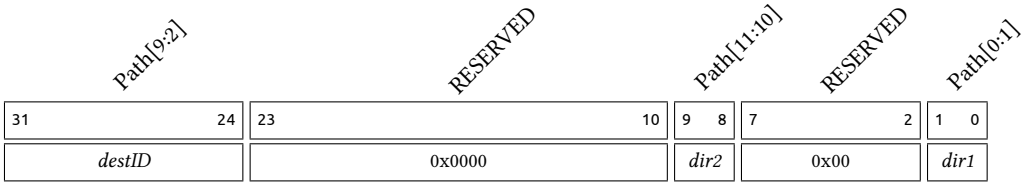
RESERVED Κάποια πεδία είναι δεσμευμένα για μελλοντική χρήση.

ακολουθήσει το πακέτο για κάθε router από το οποίο θα περάσει. Αυτό διευρύνει σημαντικά τον αριθμό των τοπολογιών που μπορεί να υλοποιηθεί με το *Spidergon STNoC*, ενώ εξακολουθεί να τηρεί την ίδια φιλοσοφία με το *Spidergon Routing*. Το format του προγραμματιζόμενου register είναι σχεδόν πανομοιότυπο (Register 2.2).

Και στις δύο αυτές εναλλακτικές πολιτικές δρομολόγησης, οι πληροφορίες δρομολόγησης στα registers του Network Interface, αντιστοιχούνται σε διαφορετικές περιοχές μνήμης σε Network Interface προορισμού κατά τον ίδιο τρόπο, που καθορίζεται κατά τον χρόνο σχεδίασης. Αυτά τα registers μπορούν να επαναπρογραμματιστούν σε run-time από το λογισμικό του συστήματος, ανάλογα με τις τρέχουσες απαιτήσεις συστήματος και των εφαρμογών σε χρήση.

Το *Spidergon STNoC* υλοποιεί ένα πρωτόκολλο *Fair Bandwidth Allocator* προκειμένου να παρέχει δυναμική υποστήριξη QoS, όπου κάθε σε Network Interface θα αντιστοιχείται κίνηση στο Network on Chip ανάλογη με την *FBA* τιμή του. Σε αυτό το σύστημα, οκτώ bit για τον προσδιορισμό του επιπέδου *FBA* και δύο bit για *priority* αποδίδονται σε κάθε *Initiator* Network Interface. Το *Spidergon STNoC* επιτρέπει επίσης να διαμορφωθεί ένας *Initiator* με *Multi QoS* υποστήριξη κατά τον σχεδιασμό του συστήματος, που επιτρέπει σε χωριστά ζεύγη τιμών *FBA* και *priority* για κάθε προσβάσιμη περιοχή μνήμης. Στο *Multi*

Register 2.2: Register για τον καθορισμό του routing σε *Source Routing* συστήματα



Path Καθορίζει την διαδρομή που θα ακολουθήσουν τα πακέτα για κάθε ενδιάμεσο router για αυτόν τον προορισμό.

Η default τιμή *path* (*dir1* & *destID* & *dir2*) είναι καθορισμένη κατά τον χρόνο σχεδιασμού.

Παρατηρούμε ότι πρόκειται για το ίδιο register format με *Spidergon Routing*, με τα bits που διαβάζονται να ερμηνεύονται διαφορετικά από το Network Interface.

RESERVED Κάποια πεδία είναι δεσμευμένα για μελλοντική χρήση.

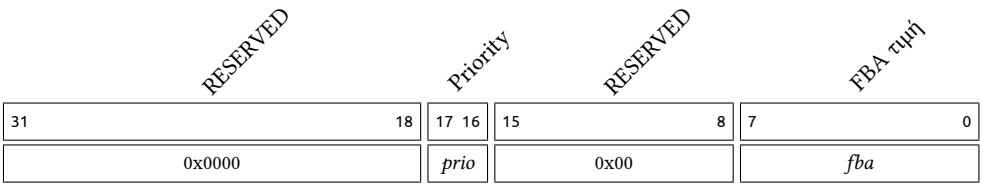
QoS οι registers με τις QoS παραμέτρους αντιστοιχούνται σε περιοχές μνήμες κατά τον ίδιο τρόπο με το routing service (Register 2.3).

Η υποκείμενη υλοποίηση εγγυάται ότι σε κάθε σύνδεσμο, οι ανταγωνιστές για την κίνηση του δικτύου θα εξυπηρετηθούν με εγγυήσεις για bandwidth, ανάλογες με την *FBA τιμή τους*. Όσοι Initiators με υψηλότερη προτεραιότητα *priority* εξυπηρετούνται νωρίτερα, και κατά τα άλλα σε στυλ round robin. Είναι σημαντικό να σημειωθεί ωστόσο, ότι μια μετάδοση πακέτου δεν θα διακοπεί, προκειμένου να διατηρηθούν οι QoS απαιτήσεις· αυτό πρέπει να λαμβάνεται υπόψη κατά τον προγραμματισμό της υπηρεσίας QoS. Η τιμές *FBA* και *priority* μπορούν να επανάπρογραμματιστούν κατά το run-time, καθώς και για οποιαδήποτε διασύνδεση δικτύου που έχει ενεργοποιημένη την υποστήριξη *QoS* ή *Multi QoS* στην φάση του σχεδιασμού. Αυτό προσφέρει εξαιρετική ευελιξία στο σχεδιαστή του συστήματος, να προσαρμόσει την κυκλοφορία στο Network on Chip δυναμικά και ανάλογα με τις τρέχουσες απαιτήσεις.

Διατίθεται επίσης και το service για security, που λειτουργεί σαν firewall του traffic που περνάει από ένα Network Interface. Αυτό προγραμματίζεται από μια σειρά από επανάπρογραμματιζόμενα registers. Η πρώτη ομάδα από registers καθορίζει μια σειρά από περιοχές μνήμης που θέλουμε να φιλτράρουμε την πρόσβαση σε αυτές. Ο αριθμός των περιοχών μνήμης καθορίζεται κατά τον χρόνο σχεδιασμού, και για καθεμία χρησιμοποιούνται 3 registers (Register 2.4). Σε αυτά καθορίζουμε επίσης και τα global δικαιώματα πρόσβασης σε κάθε συγκεκριμένη περιοχή, με 3 bit για δικαιώματα read, write και execute.

Στην συνέχεια οργανώνουμε τα IP Cores για τα οποία θέλουμε να δημιουργήσουμε κανόνες που φιλτράρουν την πρόσβαση, σε ομάδες με βάση το AXI ID τους (Register 2.5). Κάθε ομάδα περιλαμβάνει έως 2 μέλη, και ορίζουμε κατά τον χρόνο σχεδίασης του

Register 2.3: Register για τον καθορισμό του Quality of Service



FBA τιμή Καθορίζει την FBA τιμή για το traffic που αντιστοιχεί σε αυτόν τον register.

Η default τιμή *fba* είναι καθορισμένη κατά τον χρόνο σχεδιασμού.

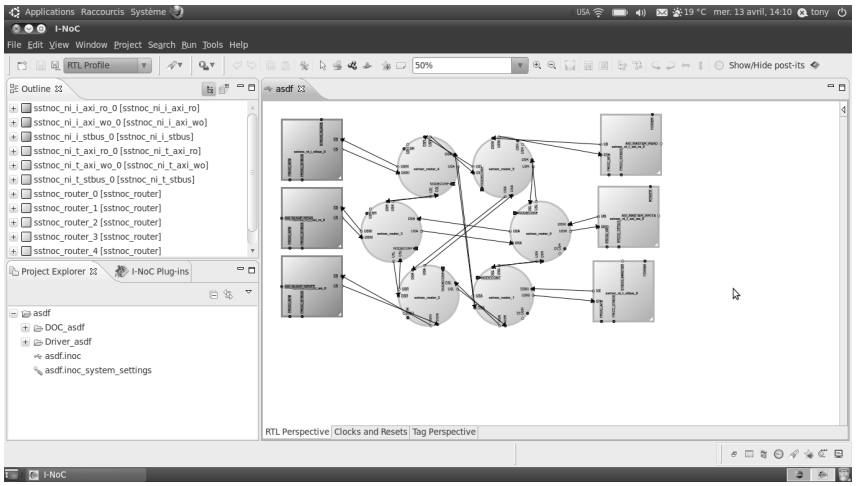
Priority Καθορίζει την προτεραιότητα για το traffic που αντιστοιχεί σε αυτόν τον register.

Η default τιμή *prio* είναι καθορισμένη κατά τον χρόνο σχεδιασμού.

RESERVED Κάποια πεδία είναι δεσμευμένα για μελλοντική χρήση.

Network on Chip έως 8 ομάδες με τον αντίστοιχο αριθμό regiters.

Τέλος δημιουργούμε τους κανόνες που αντιστοιχούν έναν αριθμό περιοχών μνήμης, με δικαιώματα πρόσβασης για κάθε μία από τις ομάδες που έχουμε ορίσει (Register 2.6). Κάθε κανόνας αντιστοιχεί σε έναν register, και ο αριθμός των κανόνων ορίζεται κατά τον χρόνο σχεδίασης.



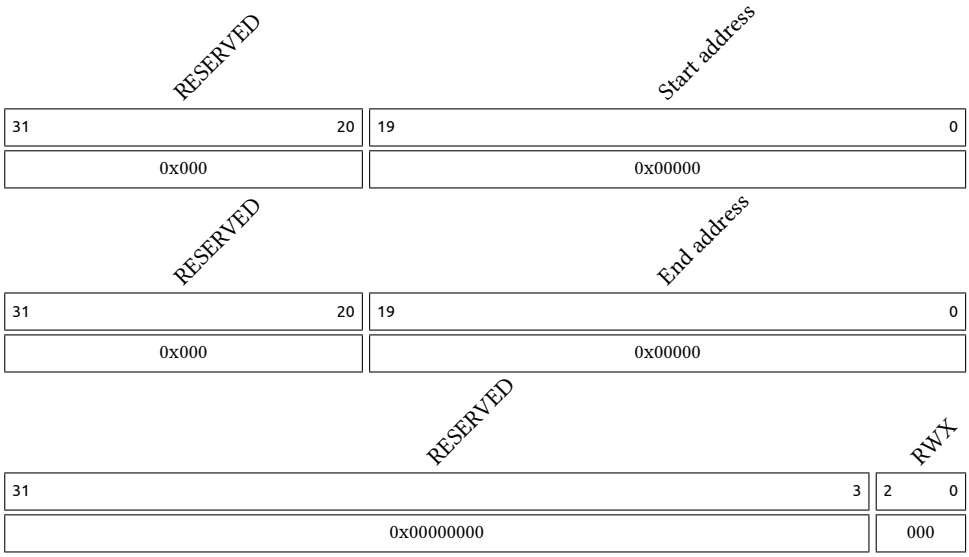
Σχήμα 2.4: Το γραφικό εργαλείο *iNoC*

2.3.1 Το γραφικό εργαλείο iNoC

Για να βοηθηθεί ο σχεδιαστής ενός συστήματος να δημιουργήσει συστήματα βασισμένα στο *Spidergon STNoC* interconnect, διατίθεται το γραφικό εργαλείο *iNoC*. Οι δρομολογητές και οι διασυνδέσεις του δικτύου αντιπροσωπεύονται με εικονίδια και οι συνδέσεις μεταξύ τους μπορούν να επεξεργασθούν με γραφικό τρόπο, επιτρέποντας την εύκολη σχεδίαση οποιασδήποτε τοπολογίας (σχήμα 2.4), ενώ οι παράμετροι σχεδιασμού των υπηρεσιών του Network on Chip και των Network Interfaces μπορούν να ρυθμιστούν. Οι διευθύνσεις μνήμης των προσβάσιμων περιοχών μνήμης από το Network on Chip και τις διευθύνσεις μνήμης των προγραμματιζόμενων registers μέσα σε κάθε Network Interface μπορούν επίσης να ρυθμιστούν από αυτό το εργαλείο.

Το *iNoC* περιλαμβάνει υποστήριξη για plug-ins, με υπάρχοντα plug-ins για την παραγωγή RTL scripts, verification environments, καθώς και τεκμηρίωση που περιγράφει το τρέχων *Spidergon STNoC* σύστημα, με τις ρυθμισμένες παραμέτρους και τις διαθέσιμες προγραμματιζόμενες υπηρεσίες. Στην εργασία αυτή χρησιμοποιούμε αυτή την υποστήριξη για plug-ins, για την επέκταση του *iNoC* με τη δυνατότητα δημιουργίας driver για το Linux εύκολα, κατάλληλο για κάποιο *Spidergon STNoC* system· αυτό επιτρέπει το λογισμικό μας να μπορεί να χρησιμοποιηθεί από τους χρήστες της *Spidergon STNoC* τεχνολογίας χωρίς να επεμβαίνει στα υπάρχοντα workflows.

Register 2.4: Περιοχή μνήμης για χρήση από το security service



Start address Τα 20 πιο σημαντικά bits της διεύθυνσης όπου αρχίζει αυτή η περιοχή μνήμης.

Η default τιμή είναι μηδενισμένη.

End address Τα 20 πιο σημαντικά bits της διεύθυνσης όπου τελειώνει αυτή η περιοχή μνήμης.

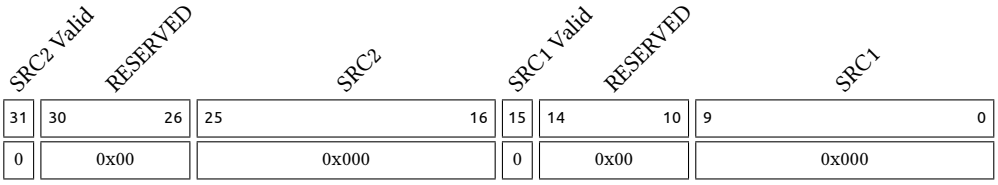
Η default τιμή είναι μηδενισμένη.

RWX Τα global δικαιώματα πρόσβασης προς αυτήν την περιοχή μνήμης.

Η default τιμή είναι μηδενισμένη.

RESERVED Κάποια πεδία είναι δεσμευμένα για μελλοντική χρήση.

Register 2.5: Register με ζευγάρι πηγών για φιλτράρισμα από το security service



SRC1 Το AXI ID του πρώτου IP core μέλος του ζευγαριού.

Η default τιμή είναι μηδενισμένη.

SRC1 Valid Ορίζουμε αυτό το bit ως 1 εάν έχουμε καταχωρίσει μια έγκυρη τιμή στο SRC1.

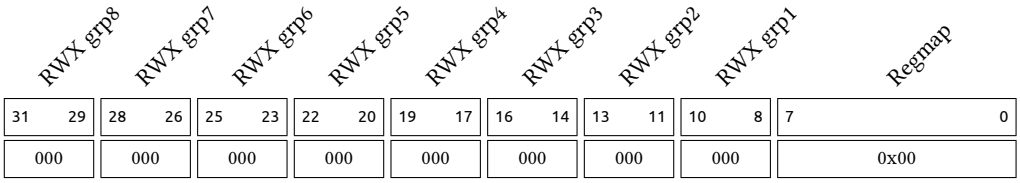
SRC2 Το AXI ID του δεύτερου IP core μέλος του ζευγαριού.

Η default τιμή είναι μηδενισμένη.

SRC2 Valid Ορίζουμε αυτό το bit ως 1 εάν έχουμε καταχωρίσει μια έγκυρη τιμή στο SRC2.

RESERVED Κάποια πεδία είναι δεσμευμένα για μελλοντική χρήση.

Register 2.6: Register με τους κανόνες φιλτραρίσματος για το security service



Regmap Ένα bitmap που αντιστοιχεί με ένα bit ανά memory region που έχουμε ορίσει, εάν ο τρέχων κανόνας ισχύει για αυτό.

Η default τιμή είναι μηδενισμένη.

RWX grp1 Για το πρώτο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

RWX grp2 Για το δεύτερο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

RWX grp3 Για το τρίτο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

RWX grp4 Για το τέταρτο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

RWX grp5 Για το πέμπτο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

RWX grp6 Για το έκτο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

RWX grp7 Για το εύδομο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

RWX grp8 Για το όγδοο ζευγάρι IP Cores με AXI ID που έχουμε ορίσει, τα δικαιώματα πρόσβασης (Read-Write-Execute) στις καθορισμένες memory regions.

3 Γενικές απαιτήσεις συστήματος

Έχοντας δει τα πλεονεκτήματα των Network on Chip, και πώς μια υλοποίηση τους όπως το *Spidergon STNoC* προσφέρει αυξημένες δυνατότητες και ευελιξία, όπως η δυνατότητα επαναπρογραμματισμού του routing μεταξύ δύο nodes σε run time, υπηρεσίες τύπου QoS, security, τίθεται το ζήτημα της αξιοποίησης των νέων αυτών δυνατοτήτων. Καλούμαστε λοιπόν να καλύψουμε ένα κενό, αυτό του software (ή middleware) που επιτρέπει στον σχεδιαστή ενός σύγχρονου System on Chip όπως το *Spidergon STNoC* να αξιοποιήσει το διαθέσιμο hardware με το καλύτερο δυνατό τρόπο.

3.1 Απαιτήσεις από άποψης παρεχόμενων APIs

Αυτό που κληθήκαμε λοιπόν να υλοποιήσουμε, είναι η δημιουργία των εργαλείων, μέσω των οποίων ο σχεδιαστής του λογισμικού ενός ενσωματωμένου συστήματος, θα μπορεί ορίζει τις βέλτιστες παραμέτρους λειτουργίας ενός συστήματος *Spidergon STNoC* ανάλογα με τις ανάγκες των εφαρμογών που τρέχουν εκείνη την στιγμή. Θα πρέπει δηλαδή το λογισμικό του συστήματος, να έχει την δυνατότητα να ορίζει ανά πάσα στιγμή παραμέτρους για οποιαδήποτε από τις υπηρεσίες που παρέχονται από το *Spidergon STNoC*, είτε άμεσα, είτε έμμεσα εν μέσω υψηλότερου επιπέδου λειτουργιών.

Επιπλέον είναι επιθυμητό όποιο λογισμικό εκμεταλλεύεται αυτού του είδους την λειτουργικότητα, να μην προσθέτουμε περισσότερους περιορισμούς στην λειτουργία του. Πρέπει όχι μόνο να καλύψουμε όσο δυνατόν μεγαλύτερο εύρος συστημάτων, τουλάχιστον θεωρητικά, αλλά επίσης όποιο σύστημα αξιοποιεί τις δυνατότητες του *Spidergon STNoC* μέσω του λογισμικού μας, να διατηρεί το όποιο portability διαθέτει. Φυσικά είναι αδύνατον να υλοποιήσουμε και να εξετάσουμε το λογισμικό μας σε κάθε δυνατή πλατφόρμα· στην πράξη το σύστημα μας θα τρέχει πάνω σε Linux - η πλέον κυρίαρχη πλατφόρμα σε embedded συστήματα - αλλά η αρχιτεκτονική μας θα πρέπει να είναι έγκυρη για οποιοδήποτε οικοσύστημα, και η υλοποίηση θα πρέπει να είναι portable προς άλλες πλατφόρμες.

Οι υπηρεσίες στις οποίες θα πρέπει να παρέχουμε απευθείας πρόσβαση περιλαμβάνουν τουλάχιστον τις υπηρεσίες QoS και προγραμματιζόμενου routing, είτε Spidergon Routing είτε Source Routing. Οι πιο προχωρημένες λειτουργίες που καλούμαστε να υλοποιήσουμε είναι λειτουργίες χρήσιμες για debugging και development του συστήματος, και λειτουργίες για αλλαγή μεταξύ «προφίλ» ρυθμίσεων του Network on Chip. Επίσης είναι ζητούμενο να μπορεί ο σχεδιαστής ενός συστήματος, να μπορεί να ορίσει τις επιθυμητές του τιμές για εγγυημένο bandwitdh, και το λογισμικό μας να αναλαμβάνει το

έργο εύρεσης κατάλληλων παραμέτρων για την υπηρεσία QoS, έτσι ώστε αυτή να μην χρειάζεται να προγραμματίζεται άμεσα.

Σημαντικό είναι επίσης, να μπορούμε να εξυπηρετήσουμε το τεράστιο εύρος τοπολογιών που υποστηρίζει το *Spidergon STNoC* μέσω του εργαλείου *iNoC*. Όταν ο σχεδιαστής καθορίζει το Network on Chip για το σύστημα του, θα πρέπει να έχει τα εργαλεία που θα κάνουν integrate το σύστημα του με το λογισμικού που παρέχουμε. Ο καλύτερος τρόπος να γίνει αυτό είναι μέσω των εργαλείων που ήδη χρησιμοποιούνται στον σχεδιασμό, δηλαδή το *iNoC*.

Επιγραμματικά οι κύριες απαιτήσεις των interfaces που σχεδιάζουμε:

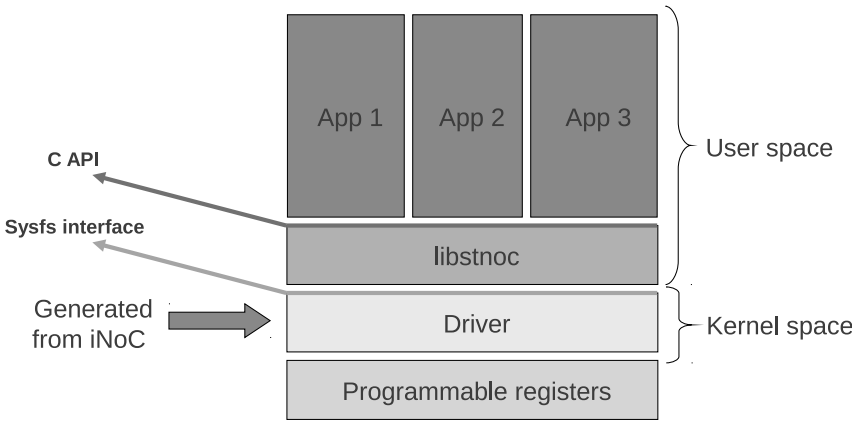
- Αρχική υλοποίηση του συστήματος πάνω σε Linux.
- Συμβατότητα με μεγάλο εύρος συστημάτων, γλωσσών προγραμματισμού και λογισμικών.
- Portability όποιου λογισμικού αξιοποιεί τα services του *Spidergon STNoC*.
- Πρόσβαση σε QoS, routing (είτε Spidergon είτε Source Routing).
- Εργαλεία για τον σχεδιαστή του συστήματος που να του δίνουν εικόνα του Network on Chip κάτα την ανάπτυξη του συστήματος.
- Εναλλαγή μεταξύ σετ ρυθμίσεων (προφίλ) του *Spidergon STNoC*.
- Εύρεση των απαραίτητων ρυθμίσεων QoS με βάση απαιτήσεων σε bandwidth.
- Integration του λογισμικού με το Network on Chip επεκτείνοντας κατάλληλα το *iNoC*.

3.2 Γενική αρχιτεκτονική

Όπως έχουμε δει έως τώρα, το *Spidergon STNoC* παρέχει πρόσβαση στα επαναπρογραμματιζόμενα services του μέσω μιας σειράς από memory mapped registers. Όπως είναι τυπικό, χρειάζεται ένας driver σε μορφή Linux kernel module που θα έχει πρόσβαση στις διευθύνσεις των registers· αν και υπάρχουν δυνατότητες για πρόσβαση αυτών των τιμών από user space, δεν είναι ο στόχος μας η έρευνα εναλλακτικών μοντέλων drivers.

Δεν είναι επιθυμητό ούτε πρακτικό να ενσωματωθεί όλη η ζητούμενη λειτουργικότητα στον driver. Δεδομένου ότι όλος ο κώδικας του driver τρέχει σε kernel space, κάθε επιπλέον feature μπορεί να θεωρηθεί και άλλο ένα vector of attack για έλεγχο του συστήματος, ή άλλο ένα σημείο όπου πιθανά bugs θα απειλείσουν την σταθερότητα του συστήματος. Εξάλλου, το interface που παρέχει ο driver προς user space, περιορίζεται από τα facilities που παρέχει ο πυρήνας του Linux, και μόνο portable προς άλλα λειτουργικά δεν είναι.

Καταλαβαίνουμε λοιπόν ότι είναι απαραίτητα τουλάχιστον 2 layers στην αρχιτεκτονική μας· ότι είναι απαλύτως απαραίτητο να βρίσκεται σε kernel space υλοποιημένο στον driver, και μια βιβλιοθήκη που θα υλοποιεί τις απαιτήσεις μας, λειτουργώντας πλέον σε user space και με portable τρόπο. Η βιβλιοθήκη δηλαδή, αν και θα τρέχει σε Linux πάνω απο τον Linux driver, δεν θα κάνει άλλες υποθέσεις για το σύστημα και θα υλοποιηθεί με τρόπο που να μπορεί να μεταφερθεί σε άλλα συστήματα, και πάνω από κάποιον άλλον driver.



Σχήμα 3.1: Αρχιτεκτονική του λογισμικού σε περιβάλλον Linux

Η βιβλιοθήκη θα παρέχει κάποιο API, ανεξάρτητο από το λειτουργικό σύστημα, έτσι ώστε οι εφαρμογές που τελικά χρησιμοποιούν τις δυνατότητες που ανοίγει το λογισμικό μας να είναι portable προς οποιοδήποτε σύστημα μελλοντικά θα μπορούμε να υποστηρίξουμε.

Generate του driver με το *iNoC* Εφόσον με το *iNoC* μπορούμε να δημιουργήσουμε ένα εύρος από διαφορετικά Network on Chip, πρέπει και η αρχιτεκτονική μας να προσαρμόζεται και σε διαφορετικά συστήματα. Προσεγγίζουμε αυτό τον στόχο, δίνοντας την δυνατότητα στον χρήστη να κάνει generate τον driver μέσα απο το *iNoC*, και απομονώνοντας στον driver όποιες παραμέτρους μπορούν να διαφοροποιούν την λειτουργία του λογισμικού.

Αυτό περιλαμβάνει

- Τον αριθμό, τις ιδιότητες και ένα αναγνωριστικό για κάθε network interface στο Network on Chip.
- Την διεύθυνση των προγραμματιζόμενων registers στην μνήμη.
- Ποια services υποστηρίζει κάθε network interface .
- Τύπος routing (Source ή Spidergon Routing).

- Επιπλέον πληροφορίες για χρήση από υψηλότερου επιπέδου layers του λογισμικού: εναλλακτικές διαδρομές για το routing service για συστήματα που χρησιμοποιούν *Source Routing*.

3.3 Απαιτήσεις τελικών εφαρμογών

Φυσικά, θέλουμε και μία εφαρμογή που να τα αξιοποιεί όλα αυτά· αυτό θα είναι ένα εργαλείο που θα παρέχει ένα πρακτικό command line interface προς στο *Spidergon STNoC*, εύκολο στην χρήση. Αυτό το εργαλείο εξυπηρετεί σκοπούς debugging για τον σχεδιαστή ενός συστήματος βασισμένου σε *Spidergon STNoC*, αλλά επίσης ο πηγαίος κώδικας του να μπορεί να αποτελέσει ένα εξαιρετικό παράδειγμα και proof of concept για την χρήση των API που παρέχουμε.

Επίσης, θέλουμε το σύστημα μας να παρουσιαστεί να τρέχει σε ένα πλήρες οικοσύστημα λογισμικού, την πλατφόρμα Android της Google για κινητά τηλέφωνα. Όλα τα προαναφερθέν εργαλεία πρέπει να τρέχουν πάνω σε Android, αλλά επίσης να παρέχεται η λειτουργικότητα του συστήματος μας και σε εφαρμογές γραμμένες για το Dalvik VM του Android· αυτό σημαίνει ότι δεν υπάρχει πρόσβαση σε κάποιο interface αγνωστικό της πλατφόρμας. Θα πρέπει να υλοποιηθεί ένα επιπλέον layer που θα υλοποιεί ένα επιπλέον interface σε Java, χρησιμοποιώντας όμως και τις κλάσεις του Android.

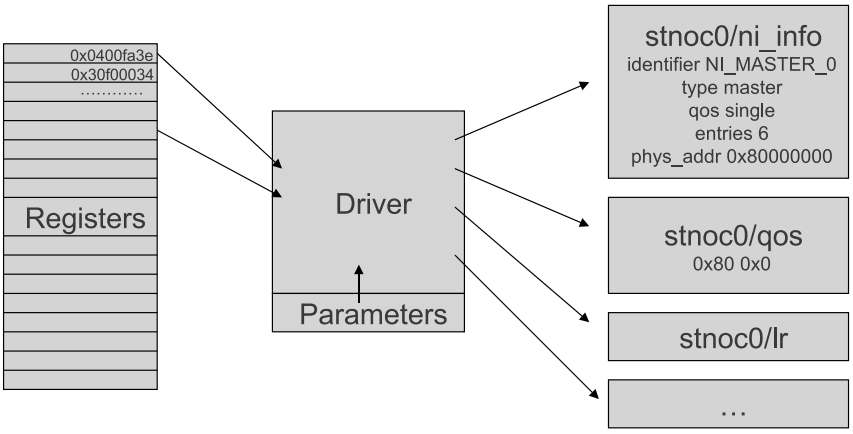
Τέλος η υποστήριξη για Android να ολοκληρώνεται τελικά, παρέχοντας και ένα GUI application που θα δίνει την δυνατότητα ελέγχου των παραμέτρων QoS και routing του *Spidergon STNoC*.

4 Αρχιτεκτονική software driver και services

Ένα τυπικό build του λογισμικού μας αποτελείται από έναν Linux driver προσαρμοσμένο σε ένα δεδομένο *Spidergon STNoC* σύστημα, και την *libstnoc* C βιβλιοθήκη, η οποία λειτουργεί ως επίπεδο αφαίρεσης πάνω από τον driver, παρέχοντας την επιθυμητή λειτουργικότητα. Η υλοποίηση μας περιλαμβάνει ένα plug-in για το *iNoC* το οποίο επιτρέπει στον σχεδιαστή του συστήματος να παράγει το απαιτούμενο πρόγραμμα οδήγησης για ένα δεδομένο *Spidergon STNoC* interconnect (σχήμα 3.1). Η *libstnoc* είναι γραμμένη κατά modular τρόπο, επιτρέποντας τη μεταφορά σε διαφορετικά συστήματα στο μέλλον, τα οποία μπορεί να χρησιμοποιούν διαφορετικό πρόγραμμα οδήγησης με διαφορετικό interface προς το user space.

4.1 Σχεδίαση του Linux driver

Ο driver για συστήματα Linux αξιοποιεί το sysfs filesystem interface [5] [6], ένα interface του πυρήνα προς το user space που βασίζεται σε μια ιεραρχία από καταλόγους και αρχεία στο /sys/, για να παρέχει πρόσβαση στις παραμέτρους λειτουργίας του *Spidergon STNoC*. Αυτό γίνεται δεσμεύοντας τον κατάλογο /sys/devices/system/stnoc/ και διατηρώντας εκεί τα ανάλογα αρχεία που αντιπροσωπεύουν τις παραμέτρους του Network on Chip.



Σχήμα 4.1: Ο Linux driver μετασχηματίζει τις binary τιμές στα προγραμματιζόμενα registers, σε μια μορφή απλού κειμένου στο sysfs filesystem

Στις πληροφορίες που διατίθενται από τον driver περιλαμβάνεται το είδος του routing που χρησιμοποιείται από την υλοποίηση του Network on Chip, οι διευθύνσεις μνήμης των memory-mapped registers που παρέχουν την run time παραμετροποίηση του συστήματος, ο αριθμός και τα «ονόματα» (text IDs) των Network Interfaces, το είδος των Network Interfaces και τι services έχουν ενεργοποιηθεί σε αυτά.

Επιπλέον, ο driver παρακολουθεί τις τιμές που κρατούν οι προγραμματιζόμενοι registers των Network Interfaces, και εκθέτει την λειτουργικότητα τους μέσω εγγράψιμων απλών αρχείων κειμένου. Έτσι υπάρχουν αρχεία στην ιεραρχία sysfs για τις τρέχουσες τιμές QoS, routing, και ασφαλείας. Διαβάζοντας απο αυτά τα αρχεία, λαμβάνουμε τις τρέχουσες ρυθμίσεις σε απλό κείμενο, και μπορούμε να γράψουμε τις νέες ρυθμίσεις με τον ίδιο τρόπο.

Απο το interface εκθέτονται επίσης και η τιμή του *Last Register*, πιθανοί προορισμοί από ένα Network Interface, εναλλακτικές τιμές routing για συστήματα με *Source Routing*, και τέλος οι τιμές των προγραμματιζόμενων registers κάθε Network Interface σε raw (hexadecimal) μορφή.

4.1.1 Κάνοντας Generate τον Linux Driver

Το *Spidergon STNoC* εκθέτει τις run time παραμέτρους του, με μια σειρά από memory mapped προγραμματιζόμενων registers. Τυπικά, ένα πρόγραμμα οδήγησης kernel space απαιτείται για την πρόσβαση αυτών των registers για να παρέχει μια διεπαφή στο user space. Ωστόσο, δεδομένου ότι κάθε υλοποίηση υλικού πρόκειται να χρησιμοποιήσει ένα διαφορετικό *Spidergon STNoC* σύστημα, με διαφορετικές υπηρεσίες και τοπολογίες, μια σειρά παραμέτρων στα header files του οδηγού θα πρέπει να καθοριστούν. Σε αυτές περιλαμβάνονται:

- Το πρωτόκολλο δρομολόγησης που χρησιμοποιείται (*Source Routing* ή *Spidergon Routing*).
- Τον αριθμό των Network Interfaces που υπάρχουν στο Network on Chip.
- Οι διευθύνσεις των memory-mapped registers στην μνήμη.

Επιπλέον, για κάθε Networ Interface χρειάζονται οι εξής παράμετροι:

◇ Network Interface ID (ένα αναγνωριστικό απλού κειμένου).

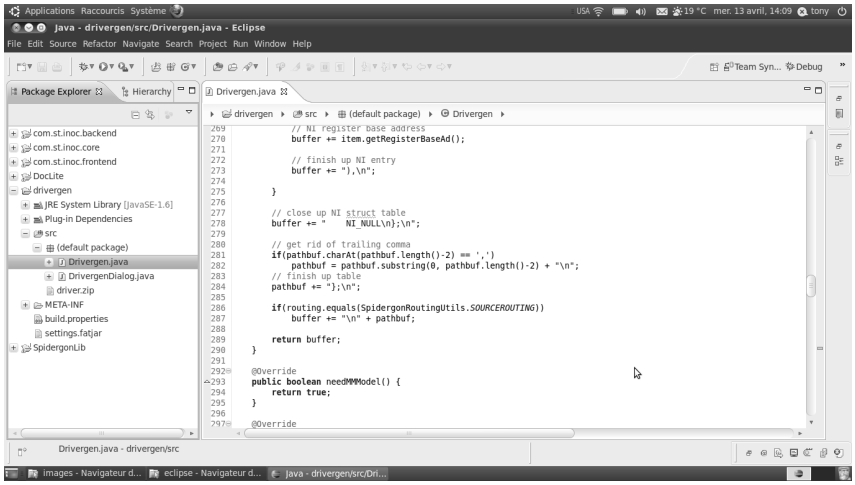
◇ Ο τύπος του κάθε δικτύου (*Master* ή *Slave*).

◇ Ενεργοποιημένες υπηρεσίες (*QoS*, *Multi_QoS*, *Προγραμματιζόμενο Routing*, *Ασφάλεια τύπου firewall*).

◇ Προσβάσιμοι προορισμοί.

◇ Μια λίστα των εναλλακτικών διαδρομών (σε κωδικοποιημένη μορφή) που μπορούν να χρησιμοποιηθούν, για καθένα απο τους προσβάσιμους προορισμούς (για υλοποιήσεις που χρησιμοποιούν *Source Routing*).

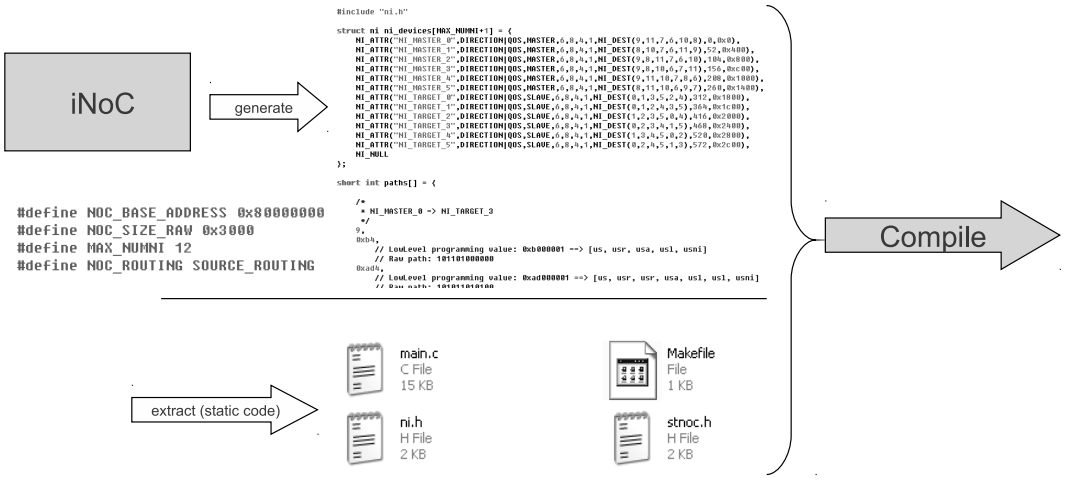
Ορισμένες από αυτές τις παραμέτρους αποτελούν απόλυτη προϋπόθεση για μια λειτουργική υλοποίηση προγράμματος οδήγησης, ενώ άλλες, π.χ. το ID απλώς εκτίθενται όπως είναι στο user space. Αυτές χρησιμοποιούνται από την *libstnoc* για την υλοποίηση πιο εύχρηστων APIs· π.χ. είναι πιο παραγωγικό να αναφερθώ στις διεπαφές δικτύου στον κώδικα εφαρμογής υψηλού επιπέδου με ένα ανθρωπίνως αναγνωρίσιμο ID κειμένου, αντί για ένα αριθμητικό αναγνωριστικό.



Σχήμα 4.2: Η ανάπτυξη του plugin για το *iNoC* σε περιβάλλον Eclipse

Ένας έγκυρος verified οδηγός για ένα δεδομένο *Spidergon STNoC* σύστημα δεν απαιτεί την χειροκίνητη ρύθμιση των παραμέτρων αυτών· έχουμε υλοποιήσει ένα plug-in για το *iNoC* περιβάλλον που επιτρέπει την άμεση δημιουργία του Linux driver, με τον ίδιο τρόπο που ο σχεδιαστής του συστήματος είναι ήδη σε θέση να παραγάγει τα *RTL scripts* για την επιθυμητή τοπολογία. Ο προκύπτων Linux driver μπορεί να γίνει compile σε πρόσφατες εκδόσεις του πυρήνα του Linux, είτε ενσωματωμένος στο image του πυρήνα, ή σαν module του πυρήνα. Το interface του οδηγού εκτίθεται μέσω του sysfs filesystem interface [6] [5].

Ο σχεδιαστής του λογισμικού του συστήματος μπορεί να χρησιμοποιήσει την διεπαφή με τον οδηγό, είτε άμεσα στον κώδικα εφαρμογής, ή κατά το debugging ενός *Spidergon STNoC* συστήματος, δεδομένου ότι βασίζεται στο filesystem based sysfs Linux interface. Ωστόσο, αυτό δεν είναι ιδανικό, ειδικά κατά την ανάπτυξη κώδικα συστήματος· η διαδικασία πρόσβασης των sysfs αρχείων, parsing τους και τη μορφοποίηση νέων παραμέτρων για να γραφτούν πίσω, μπορεί να είναι επαναλαμβανόμενη και χρονοβόρα. Επιπλέον, η εκ νέου εφαρμογή του υπάρχοντα κώδικα σε διάφορα codebases, θα είχε ως αποτέλεσμα την ύπαρξη πολλαπλών εκδόσεων για debugging και maintainance. Επομένως η ανάπτυξη κώδικα χρησιμοποιώντας απευθείας το sysfs interface περιορίζει την επανα-



Σχήμα 4.3: Τα στοιχεία για το driver generation από το *iNoC*

χρησιμοποίηση κώδικα σε διαφορετικά συστήματα με διαφορετικό driver.

4.2 Libstnoc, γενική αρχιτεκτονική

Ο σχεδιαστής του software ενός συστήματος *Spidergon STNoC* δεν χρειάζεται να καταφύγει στο OS specific interface του driver, καθώς παρέχουμε την *libstnoc*, η οποία υλοποιεί ένα portable C API, το οποίο αποκύπτει τις συγκεκριμένες λεπτομέρειες του συστήματος. Ο σχεδιαστής μπορεί να αναφέρεται στα Network Interfaces του Noc βάση των ID κειμένου τους που ορίζονται στο iNoC αντί μιας διεύθυνσης στη μνήμη, να λάβει πληροφορίες για τις υπηρεσίες που έχουν ενεργοποιηθεί και να αλλάξει τα run time parameters τους, χωρίς να θυσιάζει το portability του κώδικα, την αναγνωσιμότητα του και την δυνατότητα συντήρησης (maintainability). Για του λόγου το αληθές, οι εφαρμογές που χρησιμοποιούν την *libstnoc* θα μπορούσαν ακόμη και να γίνουν port μεταξύ διαφορετικών *Spidergon STNoC* υλοποιήσεων, εφ'όσον ο κώδικας είναι σωστά σχεδιασμένος, αποφεύγοντας παραδοχές σχετικά με τις διαφορές μεταξύ των συστημάτων [7] [8].

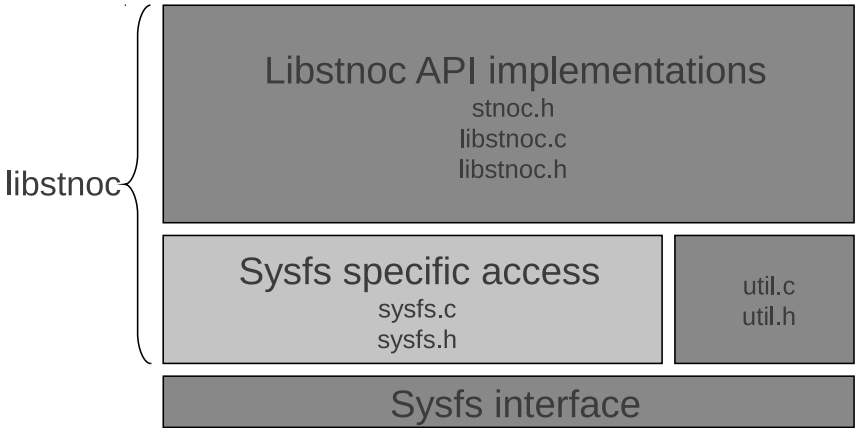
Δεδομένου ότι η *libstnoc* βρίσκεται στο user space, υψηλότερου επίπεδου λειτουργικότητα μπορεί να αναπτυχθεί, όπως διαγνωστικές υπηρεσίες, η αποθήκευση και εκ νέου εφαρμογή προφίλ του QoS κατά τη λειτουργία του συστήματος, και η παραγωγή έγκυρων *QoS* τιμών για ένα σύνολο QoS απαιτήσεων που ενδέχεται να μην είναι γνωστές με ακρίβεια offline.

Η βιβλιοθήκη έχει σχεδιαστεί κατά modular τρόπο, έτσι ώστε να είναι και εύκολα portable· ο κώδικας που χρησιμοποιεί το sysfs interface του driver, είναι διαχωρισμένος από την υλοποίηση του API της βιβλιοθήκης, κάτι που κρατάει στο ελάχιστο τον κώδικα που πρέπει να ξανά υλοποιηθεί με σκοπό το portability σε διαφορετικές πλατφόρμες (σχήμα 4.4).

Το `l`ibstnoc.c υλοποιεί το παρεχόμενο API, με την λογική του και έλεγχο για σφάλματα

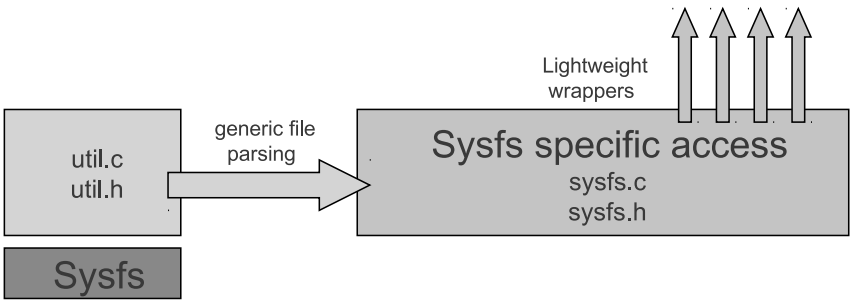
ή μη έγκυρων παραμέτρων εισόδου. Απεναντίας το χαμηλότερο επίπεδο που υλοποιείται στο sysfs.c αποτελεί ένα πολύ απλό στρώμα πρόσβασης στο hardware, χωρίς υλοποιημένη λογική και σχεδόν καθόλου έλεγχο σφαλμάτων, παρά μόνο για σφάλματα συστήματος, π.χ. αν δεν μπορεί να δεσμευτεί αρκετή μνήμη για την λειτουργία, ή αν κάποιο χαμηλότερο επίπεδο επιστρέψει σφάλμα. Παρ'όλα αυτά, αυτά τα σφάλματα δεν αντιμετωπίζονται σε αυτό το επίπεδο, αλλά επιστρέφονται στον καλούντα κώδικα, συνήθως στο libstnoc.c ο οποίος μπορεί να αποφασίσει πως θα αντιμετωπιστούν.

Η βιβλιοθήκη εκτός από έναν υποστηριζόμενο layer προς το hardware (π.χ. driver), απαιτεί και την παρουσία μιας C standard library· η τρέχουσα έκδοση υποστηρίζει τις glibc και Bionic C library. Για την μεταφορά της βιβλιοθήκης σε κάποια άλλη πλατφόρμα, αρκεί να υλοποιηθεί μονάχα ένα αντίστοιχο του sysfs.c.



Σχήμα 4.4: Η αρχιτεκτονική της βιβλιοθήκης *libstnoc*

Ο τρέχον κώδικας για Linux είναι πολύ απλός, και βασίζεται σε ένα ακόμα επίπεδο αφαίρεσης· έχοντας υλοποιήσει εσωτερικά μια γενική συνάρτηση parsing των αρχείων του sysfs interface του driver, το sysfs.c έχει υλοποιηθεί απλώς σαν μια σειρά από απλά wrappers. Έτσι ο κώδικας είναι σύντομος με το duplication να διατηρείται σε πολύ χαμηλά επίπεδα, επιτρέποντας εύκολη συντήρηση και portability.



Σχήμα 4.5: Η υλοποίηση του Linux specific μέρους του κώδικα της *libstnoc*

Πάνω σε αυτή την αρχιτεκτονική, η βιβλιοθήκη *libstnoc* παρέχει ένα υψηλού επιπέδου C API, απομακρυσμένο από τις λεπτομέρειες όπως το format των hardware registers, ή ο τρόπος που αυτά εκθέτονται από τον driver. Οι συναρτήσεις που οι εφαρμογές μπορούν να εκμεταλευτούν, περιλαμβάνουν get/set συναρτήσεις για τιμές routing και QoS, profile switching, και παραγωγή QoS profiles με βάση δεδομένων bandwidth απαιτήσεων.

4.3 Libstnoc, λειτουργικότητα QoS

Η *libstnoc*δεν γνωρίζει το διαθέσιμο εύρος ζώνης στο άκρο κάθε Network Interface· συνεπώς τα constraints εύρους ζώνης που εισάγονται από το user space δεν είναι σε μορφή bytes/s, αλλά με βάση μεριδίων του επιτεύξιμου throughput (1 μετοχή = 1/20000ό της κυκλοφορίας του Network Interface).

Η παροχή API πρόσβασης στις *QoS* παραμέτρους του *Spidergon STNoC* είναι χρήσιμη για τον σχεδιαστή του συστήματος, όταν είναι δυνατόν να προσδιοριστούν εκ των προτέρων οι απαιτήσεις που πρέπει να πληρούνται. Ωστόσο, οι απαιτήσεις του συστήματος μπορεί να αλλάζουν δυναμικά, ενώ το σύστημα λειτουργεί, ανάλογα με τη συμπεριφορά της τρέχουσας εφαρμογής. Η δυνατότητα επαναχρησιμοποίησης του κώδικα είναι επιθυμητή ώστε να μπορεί διατηρηθεί το ίδιο λογισμικό για μια σειρά συστημάτων με ελαφρώς διαφορετικές *Spidergon STNoC* υλοποιήσεις. Τέλος, η εξεύρεση των κατάλληλων *QoS* τιμών για ένα σύστημα απαιτεί μια καλή αντίληψη της υλοποίησης QoS στο *Spidergon STNoC*.

Για την αντιμετώπιση των αναγκών αυτών αναπτύξαμε στην βιβλιοθήκη *libstnoc* ένα σύνολο υψηλών επιπέδου λειτουργιών, που δέχονται μια λίστα περιορισμών ως προς τις απαιτήσεις σε εγγυημένο εύρος ζώνης μεταξύ δύο Network Interfaces. Αν οι περιορισμοί είναι ικανοποιήσιμοι από το Network on Chip, υπολογίζουμε τις απαραίτητες *QoS* τιμές και τις αποθηκεύουμε σε ένα νέο *profile*. Ο κώδικας της εφαρμογής μπορεί στη συνέχεια να εναλλάσσεστε μεταξύ αυτού και οποιουδήποτε άλλου *profile* κατά το χρόνο εκτέλεσης, πράγμα που απλοποιεί εξαιρετικά το έργο των σχεδιαστών λογισμικού του συστήματος.

4.3.1 Αναγνώριση του προβλήματος

Μπορούμε να υπολογίσουμε τις απαιτούμενες *QoS* τιμές, με την εξέταση κάθε εμπλεκόμενου συνδέσμου στο Network on Chip όπου ένας δεδομένος περιορισμός εφαρμόζεται, και να διατυπώσουμε μια εξίσωση που πρέπει να πληρούν οι τιμές *QoS* των ανταγωνιζομένων Network Interfaces. Έτσι για επιτευχθεί ένας περιορισμός που θα εγγυάται μια αναλογία B κίνησης για τον κόμβο C_0, για κάθε εμπλεκόμενη σύνδεση με διαγωνιζόμενους η ακόλουθη εξίσωση πρέπει να ικανοποιείται:

$$C_0 \geq B \sum_{i=0}^{N} C_i$$

Παρατηρούμε ότι λαμβάνουμε υπόψιν μία *QoS* τιμή για κάθε αγωνιζόμενο στις εξισώσεις (*Single QoS*), αλλά η προσέγγιση λειτουργεί πανομοιότυπα και για πολλαπλές *QoS* τιμές ανά διαγωνιζόμενο (*Multi QoS*).

Όπου C_i είναι η *QoS* τιμή για τον κόμβο i. Μπορούμε να μετακινήσουμε όλους τους αγνώστους στην αριστερή πλευρά της εξίσωσης:

$$C_0 - BC_0 - B\sum_{i=1}^{N} C_i \geq 0$$

$$\Rightarrow \quad (1-B)C_0 - B\sum_{i=1}^{N} C_i \geq 0$$

Για παράδειγμα, για να ικανοποιηθεί μια απαίτηση για 70% τοις εκατό εγγυημένου traffic ($B = 0.7$) για τον κόμβο C_0 ο οποίος χρησιμοποιείται επίσης και από τρεις άλλους κόμβους έχουμε:

$$0.3C_0 - 0.7C_1 - 0.7C_2 - 0.7C_3 \geq 0$$

Με πολλαπλούς περιορισμούς για εγγυημένη κατανομή εύρους ζώνης μεταξύ διαφορετικών ζευγών κόμβων, με καθέναν να χρησιμοποιεί έναν αριθμό συνδέσεων, μπορούμε να συγκεντρώσουμε μια σειρά εξισώσεων, όπως η παραπάνω. Μπορούμε να εκφράσουμε πλέον το πρόβλημα ως ένα πρόβλημα *Integer Programming*, όπου επιλέγουμε να ελαχιστοποιηθεί το $\sum_{i=0}^{N} C_i$. Η επίλυση του προβλήματος μας δίνει τις μικρότερες τιμές *QoS* που ικανοποιούν τους περιορισμούς μας.

4.3.2 Υλοποιημένη λύση

Όπως έχουμε δει, το πρόβλημα του υπολογισμού των επιθυμητών παραμέτρων QoS μπορεί να εκφραστεί ως ένα *Integer Programming* πρόβλημα. Αυτού του τύπου τα προβλήματα, περιλαμβάνουν μια σειρά από γραμμικές ανισότητες, με κάποιες άγνωστες μεταβλητές, για τις οποίες θέλουμε να βρεθεί μια λύση όπου οι μεταβλητές είναι ακέραιοι, και όπου μεγιστοποιούμε ή ελαχιστοποιούμε μια γραμμική optimization function. Στην περίπτωση μας μπορούμε να χρησιμοποιήσουμε το άθροισμα των μεταβλητών ως optimization function προς ελαχιστοποίηση, αποφεύγοντας έτσι να λάβουμε τιμές που προκαλούν latency που δεν είναι απαραίτητο, αφού τα FBA values είναι ανάλογα με τον αριθμό των πακέτων που θα μεταδοθούν πριν δοθεί το link προς χρήση για ένα άλλο stream.

Για τα *Integer Programming* προβλήματα υπάρχουν διάφοροι αλγόριθμοι στην βιβλιογραφία όπως Cutting Planes, και Branch & Bound τεχνικές [9]. Όμως μπορούν να αποδειχτούν εξαιρετικά βαριά υπολογιστικά, ακόμα και με μικρό αριθμό μεταβλητών, αφού το γενικό σύνολο των Integer Programming προβλημάτων είναι NP-complete. Προκειμένου να λυθεί αποτελεσματικά ένα τέτοιο πρόβλημα σε ένα σύστημα με περιορισμένους

πόρους, όπως αυτά που μπορεί να τρέξουμε ενδεχομένως, δεν μας ικανοποιεί ένας αλγόριθμος για την γενική περίπτωση των *Integer Programming* προβλημάτων, αλλά θέλουμε κάποιον αλγόριθμο που εκμεταλλεύεται τα συγκεκριμένα χαρακτηριστικά του υποσυνόλου των προβλημάτων που μας ενδιαφέρουν.

Στην *libstnoc* έχουμε εφαρμόσει μια προσέγγιση που εκτιμά μόνο τις τελευταίες συνδέσεις, από το Network Interface μέχρι το συνδεδεμένο IP core. Έχουμε υλοποιήσει έναν απλό solver εκμεταλλευόμενοι τα κοινά χαρακτηριστικά του τύπου του προβλήματος που αντιμετωπίζουμε· κυρίως, αφού $0 \leq B < 1$ υπάρχει πάντα ακριβώς μία θετική μεταβλητή σε κάθε εξίσωση περιορισμών. Υπάρχει επίσης μια μέγιστη τιμή *QoS* την οποία το σύστημα μπορεί να δεχθεί, κάτι που μπορούμε να χρησιμοποιήσουμε ως προϋπόθεση για τον τερματισμό του αλγορίθμου που υλοποιούμε.

Solver για απαιτήσεις QoS

1. Έναρξη με μία τιμής *QoS* ίση με 1 για όλους τους ανταγωνιζόμενους κόμβους.
2. Επανάληψη για κάθε ανεκπλήρωτη εξίσωση περιορισμών:
   a) Αύξηση της τιμής της (μόνο μίας) θετικής μεταβλητή με την ελάχιστη δυνατή τιμή που μπορεί να ικανοποιήσει την εξίσωση.
   b) Αν η νέα τιμή ξεπερνά την επιτρεπόμενη μέγιστη τιμή *QoS*, ο αλγόριθμος τελειώνει εδώ και δηλώνει αποτυχία (οι περιορισμοί δεν μπορούν να εκπληρωθούν).
   c) Επιστροφή στο βήμα 2.
3. Εφόσον πληρούνται όλοι οι περιορισμοί, η λύση έχει βρεθεί.

Δεδομένου ότι το *QoS* πρωτόκολλο διακόπτει ένα stream μόνο στο τέλος της μετάδοσης ενός πακέτου, πρέπει οι μεταβλητές που προκύπτουν να γίνουν normalize κατά το μέγεθος του πακέτου. Το αποτέλεσμα είναι ένα προφίλ από *QoS* τιμές που πληρούν τις αρχικές δεσμεύσεις.

5 Σχεδίαση εφαρμογών & demos

Πάνω στα χαμηλού επιπέδου components του λογισμικού μας, δηλαδή τον Linux driver και την βιβλιοθήκη *libstnoc*, υλοποιούμε χρήσιμα εργαλεία για την ανάπτυξη συστημάτων βασισμένων σε *Spidergon STNoC* τεχνολογία, που εξυπηρετούν διαγνωστικές ανάγκες, ανάγκες debugging, και επιδεικνύουν την λειτουργικότητα της προσέγγισης μας.

5.1 Το εργαλείο stnoctl

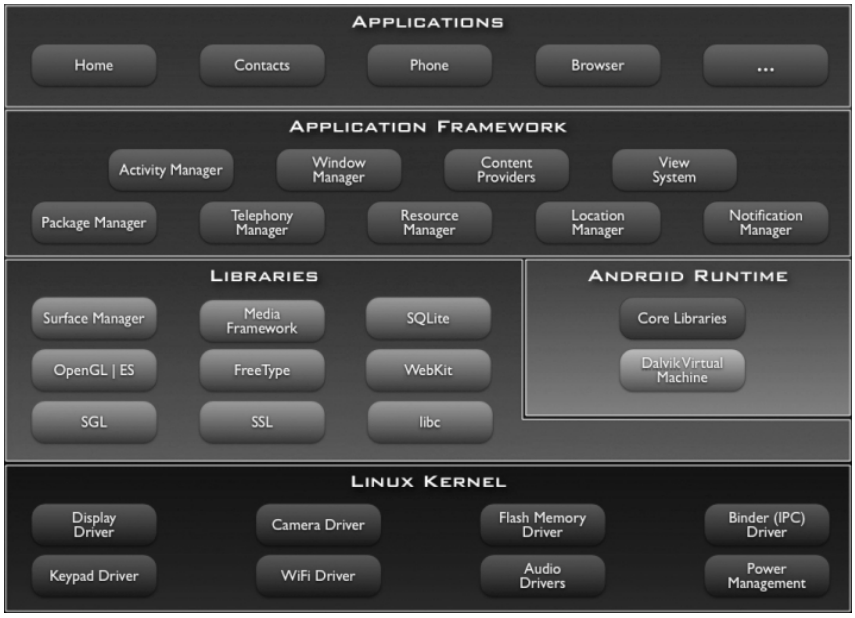
Για την επίδειξη της χρήσης της *libstnoc* και του οδηγού για Linux, έχουμε αναπτύξει το εργαλείο *stnoctl*, ένα περιβάλλον γραμμής εντολών που παρέχει πρόσβαση στο *Spidergon STNoC*, χρήσιμο για debugging ενός συστήματος που βρίσκεται σε λειτουργία. Το *stnoctl* μπορεί να γίνει compile και να τρέξει, με σχεδόν καμία αλλαγή στον κώδικα, σε οποιοδήποτε σύστημα με μια υλοποίηση *Spidergon STNoC* που υποστηρίζεται από την *libstnoc*.

Το *stnoctl* υλοποιεί μία απλή διεπαφή χρήστη προς τις δυνατότητες που παρέχει το *Spidergon STNoC* μέσω της *libstnoc*, όπως ανάγνωση και τον ορισμό των routing και QoS τιμών, αποθήκευση και εναλλαγή μεταξύ *profiles*, δημιουργία περιορισμών από πλευράς απαιτούμενου εύρους ζώνης μεταξύ δύο Network Interfaces και δημιουργία ενός *profile*, που βασίζεται στους εν λόγω περιορισμούς. Αυτές οι δυνατότητες μπορούν να χρησιμοποιηθούν κατά την ανάπτυξη του λογισμικού συστήματος ή κατά το debugging ενός συστήματος σε λειτουργία. Επιπλέον, ο κώδικας του *stnoctl* χρησιμεύει ως ένα πλήρες παράδειγμα μιας *libstnoc* cross-platform enabled εφαρμογής· το *stnoctl* μπορεί να χρησιμοποιηθεί ως έχει σε *οποιοδήποτε* σύστημα *Spidergon STNoC*.

Το *stnoctl* παρέχει αυτές τις δυνατότητες υλοποιώντας ένα απλό interactive κέλυφος που καλεί τις αντίστοιχες συναρτήσεις της *libstnoc*. Το *stnoctl* δεν κάνει υποθέσεις για το Network on Chip πάνω στο οποίο θα τρέξει· κατά την φόρτωση του εργαλείου, ενημερώνεται από τα APIs που έχουμε υλοποιήσει για τον αριθμό, τον τύπο και τα ενεργοποιημένα services των διαθέσιμων Network Interface. Έτσι το *stnoctl* αποτελεί υπόδειγμα του πώς μια εφαρμογή που εκμεταλλεύεται την *libstnoc*, μπορεί να είναι portable και μεταξύ διαφορετικών *Spidergon STNoC* υλοποιήσεων.

5.2 Android port

Έχουμε επίσης κάνει port την *libstnoc* και το *stnoctl* στο Linux-based λειτουργικό σύστημα Android για κινητά τηλέφωνα. Ανοίγοντας ένα κέλυφος γραμμής εντολών σε ένα



Σχήμα 5.1: Αρχιτεκτονική του λειτουργικού συστήματος Android

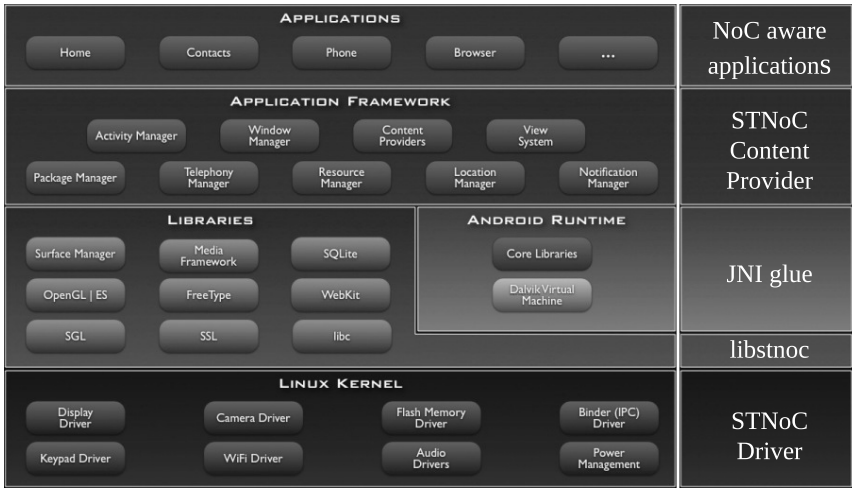
σύστημα Android σε λειτουργία που ενσωματώνει το λογισμικό μας στο Android image, μπορεί κανείς να χρησιμοποιήσει το *stnoctl* με τον ίδιο τρόπο όπως και σε ένα vanilla σύστημα Linux. Εντούτοις, η υποστήριξη για Android προϋποθέτει κάποια στοιχεία ακόμα.

5.2.1 Η αρχιτεκτονική του Android

Στο χαμηλότερο επίπεδο του Android, βρίσκεται ο πυρήνας του Linux. Η έκδοση του Linux που χρησιμοποιεί το Android ενσωματώνει κάποιες αλλαγές σε σχέση με τον vanilla Linux kernel, όμως παραμένει αρκετά παρόμοιος και το driver model είναι σχεδόν απαράλλαχτο. Πάνω από τον πυρήνα του Linux, στο user space πλέον, το Android περιλαμβάνει ένα περιβάλλον από core βιβλιοθήκες, για γραφικά, SQLite κλπ, οι οποίες τρέχουν σε ένα εντελώς διαφορετικό user space περιβάλλον σε σχέση με vanilla Linux συστήματα, και αξιοποιείται μία ελαφριά custom C library, η Bionic η οποία είναι BSD based.

Όμως μεγάλο μέρος του Android είναι βασισμένο σε Java, και τρέχει πάνω σε ένα ελαφρύ Java Virtual Machine που ονομάζεται Dalvik VM· αυτά τα υψηλότερου επιπέδου components και εφαρμογές μπορούν να αλληλεπιδράσουν μόνο με κώδικα Java που τρέχει πάνω στο Dalvik VM. Όσον αφορά τις ανάγκες του λογισμικού μας, οι περισσότερες εφαρμογές Android χειρίζονται την αποθήκευση και τη επεξεργασία δεδομένων μέσω μίας υλοποίησης *ContentProvider* [10].

Σ'αυτό το μοντέλο, οι εφαρμογές χρησιμοποιούν ένα ενιαίο API του Android, για να ζητήσουν πρόσβαση σε κάποια δεδομένα που καθορίζονται σε μορφή URL· τα δεδομένα επιστρέφονται στην εφαρμογή σε μια δομή πίνακα παρόμοια όπως σε μια βάση δεδομένων, και από τον ίδιο πίνακα μπορούν να ενημερώσουν τα δεδομένα. Παρατηρούμε ότι



Σχήμα 5.2: Τα components που έχουμε υλοποιήσει για κάθε επίπεδο του Android stack

μιας και οι εφαρμογές Android τυπικά δεν έχουν πρόσβαση στα χαμηλότερου επιπέδου interfaces, ακόμα και η SQLite στο Android χρησιμοποιείται μέσω του *ContentProvider* interface.

Για την υλοποίηση των ανάλογων interfaces σε Java, των χαμηλότερου επιπέδου components, και γενικότερα για τον ρόλο του integration μεταξύ native και Java κώδικα, το Dalvik VM περιλαμβάνει μία υλοποίηση του Java Native Interface, που επιτρέπει να γραφτεί σε C/C++ κώδικας προσβάσιμος από εφαρμογές πάνω από το Java VM.

5.2.2 Components του λογισμικού στο Android

Ο οδηγός λειτουργεί ως έχει σε Android, εφόσον συμπεριληφθεί στο build του πυρήνα που προορίζεται για μια συσκευή Android, δεδομένου ότι δεν είναι πολύ διαφορετικός από έναν vanilla Linux kernel. [11] Μιας και χρησιμοποιείται ο ίδιος driver, με το ίδιο interface προς το user space, δεν χρειάζεται καινούρια υλοποίηση του system specific μέρους του κώδικα της *libstnoc*. Ωστόσο, το user space περιβάλλον ενός Android συστήματος είναι εντελώς διαφορετικό από ένα κοινό σύστημα Linux, και χρησιμοποιεί μια άλλη BSD based υλοποίηση της standard C library, που ονομάζεται Bionic. Χρειάζονται οπότε κάποιες μικρές αλλαγές στον κώδικα της *libstnoc* έτσι ώστε να λειτουργεί και σε συστήματα Android.

Στο επίπεδο του Dalvik VM, έχουμε υλοποιήσει τον απαραίτητο κώδικα «κόλα» σε C, χρησιμοποιώντας το Java Native Interface, που επιτρέπει σε κώδικα Java να χρησιμοποιεί δυνατότητες της *libstnoc*. Ο κώδικας αυτός λειτουργεί σαν wrapper γύρω από τις συναρτήσεις της *libstnoc*, και μετατρέπει τα δεδομένα που αυτή επιστρέφει σε αντικείμενα που αναγνωρίζονται από το περιβάλλον της Java. Αυτό μας επιτρέπει να υλοποιήσουμε έναν *ContentProvider* για το Android, ο οποίος παρέχει πρόσβαση στις routing και QoS παραμέτρους για ένα σε λειτουργία σύστημα. Έτσι μέσω αυτού του *ContentProvider* interface

μπορούν τυπικές εφαρμογές Android να αποκτήσουν πρόσβαση και να ενημερώσουν, τις τρέχουσες τιμές που καθορίζουν την λειτουργίας του *Spidergon STNoC*.

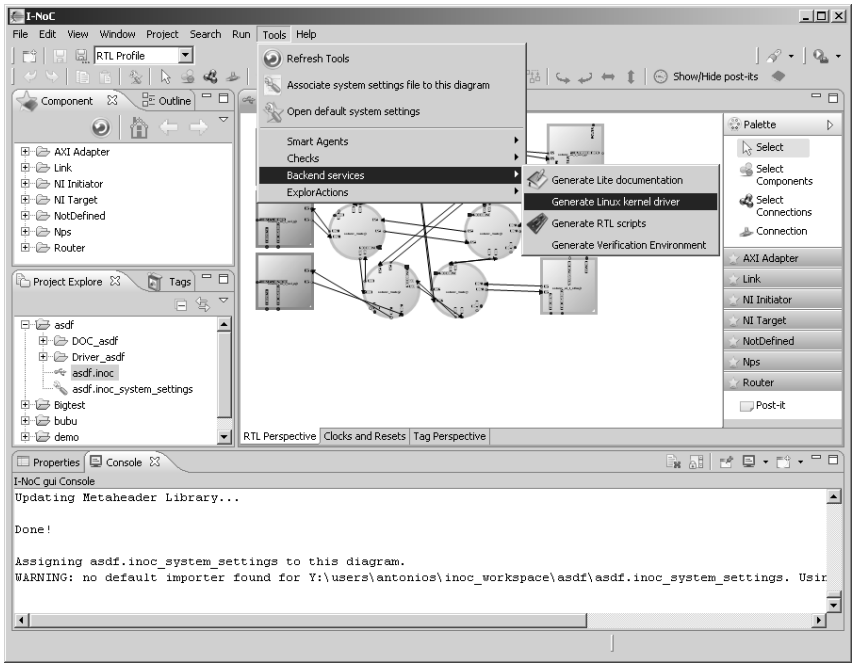
Μια εφαρμογή Android περιλαμβάνεται επίσης, η οποία επιτρέπει τον χειρισμό αυτών των ρυθμίσεων μέσω ενός γραφικού περιβάλλοντος, απευθείας από τη συσκευή. Ενώ είναι απίθανο ότι το λογισμικό μας θα χρησιμοποιηθεί με αυτόν τον τρόπο σε ένα τελικό production σύστημα, δείχνει με σαφήνεια πως μια εφαρμογή στο υψηλότερο επίπεδο του Android stack μπορεί να αξιοποιήσει τις δυνατότητες που παρέχει το *Spidergon STNoC* μέσω των υψηλού επιπέδου interfaces που έχουμε υλοποιήσει.

6 Χρήση του stack

Έχουμε δει στα προηγούμενα κεφάλαια τα components που αναπτύξαμε, πως λειτουργούν και τι ρόλο ικανοποιούν. Σε αυτό το κεφάλαιο περιγράφουμε τον τρόπο χρήσης του λογισμικού μας, από την διαδικασία του compile έως και πλήρη τεκμηρίωση των APIs και εργαλείων που παρέχουμε.

6.1 Linux driver

6.1.1 Κάνοντας generate τον Linux driver

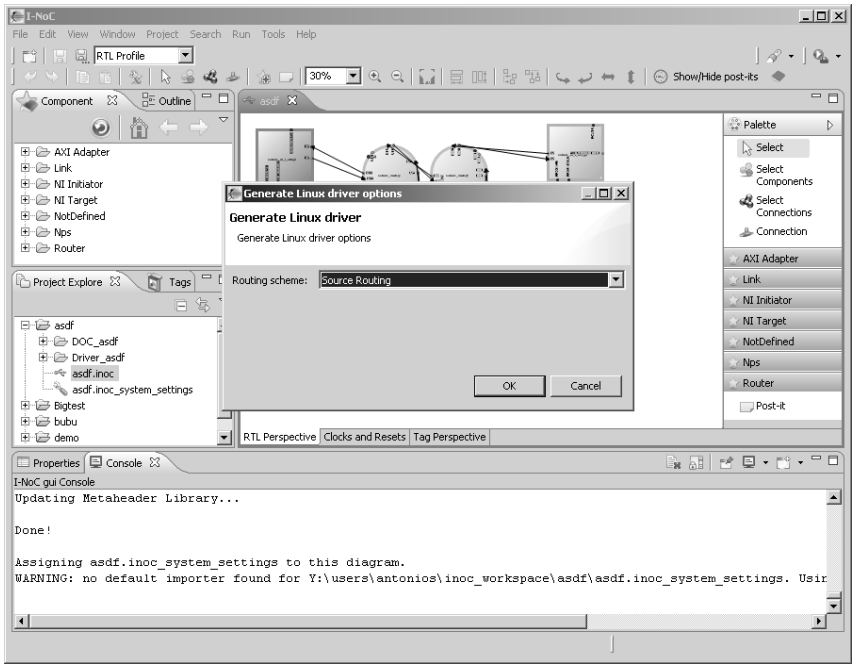


Σχήμα 6.1: Χρησιμοποιόντας το plugin για το iNoC για να γίνει generate ο driver για Linux

Το πρώτο στοιχείο που χρειάζεται να είναι παρών σε ένα σύστημα που αξιοποιεί το λογισμικό μας, είναι ο driver για Linux συστήματα. Όπως έχουμε δει, αυτός ο driver διαφέρει μεταξύ διαφορετικών *Spidergon STNoC* υλοποιήσεων, οπότε θα πρέπει με την χρήση του iNoC plugin που έχουμε αναπτύξει να κάνουμε generate τον ανάλογο driver. Ο driver που θα δημιουργήσουμε με αυτόν τον τρόπο, είναι συμβατός με vanilla συστήματα Linux,

με πρόσφατες εκδόσεις του πυρήνα, και επίσης και με τον πυρήνα που περιλαμβάνει το λειτουργικό σύστημα Android.

Η διαδικασία αυτή είναι πάρα πολύ απλή· μέσα από το ίδιο design του iNoC που έχουμε χρησιμοποιήσει για να κάνουμε generate το Network on Chip που θέλουμε για το System on Chip μας, και εφόσον το σύστημα είναι έγκυρο με όλες τις απαραίτητες ρυθμίσεις, και το plugin μας έχει εγκατασταθεί στο iNoC, μπορούμε να κάνουμε generate τον driver από το menu του iNoC: `Tools > Backend services > Generate Linux kernel driver` (σχήμα 6.1). Στην συνέχεια το λογισμικό θα μας ζητήσει αν το Network on Chip μας χρησιμοποιεί *Spidergon Routing* ή *Source Routing*· είναι σημαντικό η επιλογή που θα κάνουμε εδώ να είναι ίδια με αυτή που κάναμε όταν καναμε generate και το hardware του Network on Chip(σχήμα 6.2).



Σχήμα 6.2: Επιλογή μεταξύ *Spidergon Routing* και *Source Routing* κατά το generation του driver

Ο driver γίνεται generate, και αποθηκεύεται στον φάκελο του iNoC project μας, και απ'όπου μπορούμε να τον αντιγράψουμε προς το σύστημα στο οποίο κάνουμε compile τον πυρήνα του συστήματος μας.

6.1.2 Compilation, loading και unloading

Ο driver μπορεί να γίνει compile όπως οποιοδήποτε kernel module, ή να ενσωματωθεί στο image του πυρήνα, είτε για vanilla Linux συστήματα, είτε για χρήση σε Android. Το ανάλογο Makefile περιλαμβάνεται ήδη, οπότε αυτό είναι συνήθως υπόθεση ενός απλού

make. Όταν ο driver γίνεται compile σαν module, το αποτέλεσμα είναι το αρχείο `stnoc.ko` που μπορούμε να κάνουμε `insmod` στο σύστημα μας.

Στο Makefile περιλαμβάνονται κυρίως 2 παράμετροι που μπορούν να τροποποιηθούν από τον χρήστη. Η πρώτη από αυτές είναι η `FAKEREGS` η οποία μας δίνει την δυνατότητα να κάνουμε compile έναν driver που δεν χρησιμοποιεί το *Spidergon STNoC* hardware· αντ'αυτού, χρησιμοποιείται ένα μέρος κενής μνήμης από το σύστημα. Αυτό είναι χρήσιμο για ανάπτυξη και debugging του λογισμικού πάνω σε συμβατικούς υπολογιστές Linux που δεν περιλαμβάνουν κάποιο Network on Chip.

Επιπλέον, η `SUREGS` μας επιτρέπει να επιλέξουμε αν το sysfs interface του driver θα είναι προσβάσιμο από όλους τους χρήστες του συστήματος, ή μόνο τον root. Καθώς η αλλαγή των παραμέτρων του Network on Chip είναι μια διαδικασία που μπορεί να οδηγήσει σε αστάθεια ή παραβίαση της ασφάλειας του συστήματος, τυπικά πρέπει να γίνεται μόνο από root processes, αλλά η επιλογή αυτή είναι χρήσιμη σε συνδυασμό με την `FAKEREGS` για τον developer του λογισμικού.

6.1.3 Το sysfs interface

Ο driver εκθέτει τις προγραμματιζόμενες παραμέτρους του Network on Chip ως αρχεία απλού κειμένου στην ιεραρχία sysfs του Linux, υπό τον κατάλογο `/sys/devices/-system/stnoc/`. Μπορούμε να διαβάσουμε από τα αρχεία σε αυτόν τον κατάλογο για να λάβουμε τις τρέχουσες τιμές με τις οποίες έχουν ρυθμιστεί τα προγραμματιζόμενα services του *Spidergon STNoC*, ή ακόμα και να γράψουμε, ενημερώνοντας τις παραμέτρους με τις τιμές που θέλουμε.

Αυτό παρέχει ένα απλό interface για την διαχείριση των παραμέτρων του *Spidergon STNoC*, για χρήση κατά την ανάπτυξη του συστήματος ή από υψηλότερου επιπέδου λογισμικό. Παρ'όλα αυτά, δεν συνιστάται αυτό το interface να χρησιμοποιείται άμεσα από κάποιο λογισμικό· η *libstnoc* παρέχει ένα απλούστερο C API, με καλύτερες εγγυήσεις για σταθερότητα του interface από έκδοση σε έκδοση, και επιτρέπει επιπλέον portability και σε περισσότερα συστήματα στο μέλλον. Εξάλλου ούτε καν ο πυρήνας του Linux δεν παρέχει εγγυήσεις για την μελλοντική σταθερότητα του sysfs interface.

Αρχεία στο sysfs interface του driver Τα αρχεία που εκθέτουν τις παραμέτρους του *Spidergon STNoC* είναι διαμορφωμένα κατά 2 τρόπους κυρίως· ο πρώτος είναι ζεύγη κλειδιών - τιμής, με ένα ζευγάρι ανά γραμμή και ένα κενό να διαχωρίζει το κλειδί από την αντίστοιχη τιμή. Για παράδειγμα μια γραμμή «`maxni 12`» αναφέρεται σε 12 διαθέσιμα Network Interface στο Network on Chip.

Ο δεύτερος τρόπος διαμόρφωσης αναφέρεται σε δεδομένα σε μορφή πίνακα, όπου οι στήλες διαχωρίζονται από κενά, και κάθε αλλαγή γραμμής του κειμένου αντιστοιχεί σε διαφορετική γραμμή του πίνακα. Σε αυτή την μορφή εξαγωγής των δεδομένων, η πρώτη στήλη, δηλαδή η πρώτη τιμή κάθε γραμμής, αντιστοιχεί στο index του πίνακα, ξεκινώντας από το μηδέν. Σε αρχεία που επιτρέπουν να γράψουμε για να ενημερώσουμε τις

παραμέτρους του *Spidergon STNoC*, χρησιμοποιούμε αυτό το index για να καθορίσουμε σε ποια γραμμή θέλουμε να αλλάξουμε τιμή, και στην συνέχεια παρέχουμε τις τιμές σε μορφή απλού κειμένου, διαχωρίζοντας τις στήλες με whitespace.

/sys/devices/system/stnoc/stnoc_info

Αρχείο μόνο για ανάγνωση, διαμορφωμένο κατά ζευγάρια κλειδιών - τιμής.

maxni Ο αριθμός των διαθέσιμων Network Interface στο Network on Chip.

routing Η τιμή είναι είτε `spidergon` ή `source`, ανάλογα με τον τύπο του routing που έχει επιλεγεί κατά τον σχεδιασμό του συστήματος.

Αρχεία του sysfs interface για κάθε Network Interface Υπό τον top level κατάλογο του sysfs interface του driver, υπάρχει ένας υποκατάλογος για κάθε Network Interface διαθέσιμο στο σύστημα μας, αριθμημένα κατά αύξοντα αριθμό, π.χ. `stnoc0/` `stnoc1/` `stnoc2/` και ούτω καθ' εξής. Καθένας από αυτούς τους υποκαταλόγους περιλαμβάνει περισσότερα αρχεία με τις παραμέτρους που αντιστοιχούν στο αντίστοιχο Network Interface.

ni_info

Για κάθε Network Interface, αρχείο μόνο για ανάγνωση με ζεύγη κλειδιών - τιμής, με παραμέτρους με τις οποίες έχει σχεδιαστεί το τρέχον Network Interface.

identifier Το αναγνωριστικό identifier που έχει καθοριστεί για αυτό το Network Interface στο *iNoC*.

type Ανάλογα με τον τύπο του Network Interface, `master` ή `slave`.

qos Ανάλογα με τον τύπο του QoS service που έχει ρυθμιστεί για αυτό το Network Interface, `none`, `single`, ή `multi`.

entries Ο αριθμός των Network Interface που είναι προσβάσιμα από αυτό το Network Interface.

phys_addr Η φυσική διεύθυνση στην μνήμη των προγραμματιζόμενων registers για τα services αυτού του Network Interface.

destinations

Για κάθε Network Interface, αυτό το αρχείο μόνο για ανάγνωση παρέχει τους προορισμούς (άλλα Network Interface) που είναι προσβάσιμοι από αυτό το Network Interface. Δεδομένα σε μορφή πίνακα, με τις εξής στήλες:

1. Το index του πίνακα ξεκινώντας από το 0 στην πρώτη γραμμή.

2. Ο άυξων αριθμός που αντιστοιχεί σε αυτό το Network Interface - προορισμό στο sysfs interface του driver.

3. Το αναγνωριστικό identifier που έχει καθοριστεί για αυτό το Network Interface- προορισμό στο *iNoC*.

dir

Για κάθε Network Interface για το οποίο έχει ενεργοποιηθεί το service για προγραμματιζόμενο routing, το αρχείο αυτό μας επιτρέπει να διαβάσουμε ή να ενημερώσουμε τις τιμές που καθορίζουν το routing μέσω του Network on Chip για κάθε Network Interface προορισμό. Τα δεδομένα είναι σε μορφή πίνακα, με διαφορετικές στήλες όμως ανάλογα με το εάν το σύστημα αξιοποιεί *Spidergon Routing* ή *Source Routing*.

Spidergon routing

1. Το index του πίνακα ξεκινώντας από το 0 στην πρώτη γραμμή· κάθε καταχώριση αντιστοιχεί στην αντίστοιχη γραμμή του πίνακα `destinations`.
2. Ένα αναγνωριστικό `destID` των 8 bits, που αντιστοιχεί στον προορισμό στον οποίο καταλήγει το routing των πακέτων.
3. Ο κωδικός κατεύθυνσης `dir1` που καθορίζει την κατεύθυνση που ακολουθούν τα πακέτα, κωδικοποιημένος όπως ορίζεται από το *Spidergon Routing*.
4. Ο κωδικός κατεύθυνσης `dir2` που καθορίζει την κατεύθυνση που ακολουθούν τα πακέτα αφού ακολουθήσουν έναν ιεραρχικό σύνδεσμο, κωδικοποιημένος όπως ορίζεται από το *Spidergon Routing*.

Source routing

1. Το index του πίνακα ξεκινώντας από το 0 στην πρώτη γραμμή· κάθε καταχώριση αντιστοιχεί στην αντίστοιχη γραμμή του πίνακα destinations.
2. Το τρέχον path που καθορίζει το routing σε κάθε ενδιάμεσο router στο Network on Chip (12 bits σε hexadecimal μορφή).

paths

Εάν το Network on Chip έχει επιλεγεί κατά τον σχεδιασμό να αξιοποιεί *Source Routing*, τότε για τα Network Interface που έχουν ενεργοποιημένο το service για προγραμματιζόμενο routing, υπάρχει διαθέσιμο αυτό το μόνο για ανάγνωση αρχείο που παρέχει για τα διαθέσιμα Network Interface προορισμοί, την λίστα από τα έγκυρα paths των 12 bits που μπορούν να χρησιμοποιηθούν για δρομολόγηση των πακέτων προς εκείνο τον προορισμό. Τα δεδομένα είναι μορφοποιημένα σε στυλ πίνακα, με μεταβλητό αριθμό στηλών.

1. Το index του πίνακα ξεκινώντας από το 0 στην πρώτη γραμμή.

2. Ο άυξων αριθμός που αντιστοιχεί σε αυτό το Network Interface - προορισμό στο sysfs interface του driver.

3. Οι υπόλοιπες στήλες περιέχουν ένα εναλλακτικό path των 12 bits ανά στήλη.

qos

Τα Network Interface για τα οποία έχει ενεργοποιηθεί το service για προγραμματιζόμενο QoS (Single ή Multi) διαθέτουν αυτό το αρχείο για πρόσβαση στις QoS παραμέτρους τους. Το αρχείο είναι διαμορφωμένο σε στυλ πίνακα, και μπορεί να γραφτεί.

1. Τιμή `FBA`.
2. Τιμή `priority`.

sec_RID

Εάν ένα Network Interface έχει ενεργοποιημένο το service για προγραμματιζόμενο security, αυτό το αρχείο καθορίζει τις περιοχές της μνήμης που θέλουμε να ασφαλίσουμε. Τα δεδομένα έχουν μορφή πίνακα.

1. Αύξων αριθμός, ξεκινώντας από το μηδέν, της περιοχής μνήμης που προγραμματίζουμε.
2. Διεύθυνση στην μνήμη απ' όπου ξεκινάει η περιοχή μνήμης.
3. Διεύθυνση στην μνήμη όπου τελιώνει η περιοχή μνήμης που ασφαλίζουμε.
4. Καθολικά (global) δικαιώματα πρόσβασης στην καθορισμένη περιοχή μνήμης.

sec_SRC

Για ένα Network Interface με ενεργοποιημένο το security service, ομαδοποιεί τα *Source ID* σε ζευγάρια για τα οποία καθορίζουμε δικαιώματα πρόσβασης στις περιοχές της μνήμης. Διαμορφωμένο σε στυλ πίνακα.

1. Αύξων αριθμός, ξεκινώντας από το μηδέν, των ζευγαριών *Source IDs* για τα οποία θα καθορίσουμε δικαιώματα πρόσβασης.
2. *Source ID* της πρώτης πηγής μέλος του ζευγαριού (μπορεί να είναι κενό).
3. *Source ID* της δεύτερης πηγής μέλος του ζευγαριού (μπορεί να είναι κενό).

sec_RULES

Για ένα Network Interface με ενεργοποιημένο το security service, αποθηκεύουμε κανόνες που αντιστοιχούν δικαιώματα πρόσβασης για κάθε ομάδα με Source IDs, στις περιοχές της μνήμης που έχουμε ορίσει. Τα δεδομένα είναι διαμορφωμένα σε στυλ πίνακα.

1. Αύξων αριθμός του κανόνα, ξεκινώντας από το μηδέν.

2. Bitmap των 8 bits με κάθε bit να αντιστοιχεί σε μια ομάδα μνήμης που έχουμε ορίσει, με το οποίο καθορίζουμε για ποιες περιοχές της μνήμης ισχύει ο κανόνας.

3. Σε octal κωδικοποίηση, 3 bits ανά ομάδα Source IDs που έχουμε καθορίσει· τα 3 αυτά bits αντιπροσωπεύουν δικαιώματα πρόσβασης στην περιοχή μνήμης (Read Write Execute).

lr

Για όποια Network Interface παρέχουμ οποιοδήποτε service που εκθέτει προγραμματιζόμενα registers, υπάρχει επιπλέον διαθέσιμο και αυτό το αρχείο, που βρίσκεται σε κατάσταση είτε `unset` είτε `set`. Δηλώνει την τρέχουσα κατάσταση του *Last Register* του Network Interface. Μπορούμε να γράψουμε `set` στο αρχείο, για να ορίσουμε στον *Last Register* την τιμή 1.

raw

Για κάθε Network Interface με προγραμματιζόμενα registers, αυτό το μόνο για ανάγνωση αρχείο μας παρέχει σε raw hexadecimal μορφή της τιμές όλων των προγραμματιζόμενων registers.

6.2 Η βιβλιοθήκη libstnoc

Η *libstnoc* μπορεί να γίνει compile και να χρησιμοποιηθεί σε vanilla συστήματα Linux ή σε Android, και η αρχιτεκτονική της επιτρέπει να γίνει port σε άλλες πλατφόρμες μελλοντικά, διατηρώντας το ίδιο C API. Η χρήση της από κάποια εφαρμογή γίνεται απλά κάνοντας include στον C κώδικα το `stnoc.h` και αξιοποιώντας το C API που περιγράφουμε στην συνέχεια.

6.2.1 Compilation

Η *libstnoc* σε vanilla συστήματα Linux δεν αξιοποιεί κάποιο build system, αλλά περιλαμβάνει ένα απλό `Makefile` με το οποίο γίνεται compile. Το αποτέλεσμα είναι να γίνει build η *libstnoc* στον τρέχον κατάλογο, εκθέτοντας μόνο τα symbols του C API της. Περιλαμβάνεται επίσης και το αρχείο `Android.mk`, το οποίο επιτρέπει την βιβλιοθήκη να γίνει build από το build system του λειτουργικού Android· σε αυτήν την περίπτωση αρκεί να τοποθετηθεί ένα αντίγραφο του πηγαίου κώδικα της *libstnoc* στον κατάλογο `vendor` του πηγαίου κώδικα του Android.

Δίνοντας κατά το compile στον GCC την παράμετρο `-DDEBUG`, ενεργοποιούνται μέσα στον κώδικα της βιβλιοθήκης και μια σειρά από εντολές εξόδου στο `stderr` που τυπώνουν διάφορες πληροφορίες χρήσιμες για το debugging του κώδικα.

6.2.2 Libstnoc API overview

Εδώ θα δούμε αναλυτικά το C interface που παρέχει η libstnoc στον developer του software ενός συστήματος που θέλει να εκμεταλευτεί τα services του *Spidergon STNoC*.

Γενικό function format Οι περισσότερες συναρτήσεις που εκτίθενται από την *libstnoc* ακολουθούν το εξής function definition format· οι περισσότερες παράμετροι των συναρτήσεων είναι unsigned. Οι τιμές που αναμένονται από τις συναρτήσεις, συμπληρώνονται με τη διαβίβαση ενός δείκτη μνήμης - η τιμή που επιστρέφεται (return value) είναι συνήθως απλά ένας ακέραιος κωδικός σφάλματος. Ορισμένες συναρτήσεις ενδέχεται να επιστρέψουν παρ'όλα αυτά κάποια χρήσιμη τιμή.

```
int stnoc_func_name(unsigned param1, unsigned param2, unsigned *ret1,
    unsigned *ret2)
```

Κωδικοί σφάλματος συναρτήσεων Οι περισσότερες συναρτήσεις επιστρέφουν το αρνητικό ενός από τους ακόλουθους κωδικούς σφάλματος σε περίπτωση αποτυχίας.

```
// error types
#define NOSTNOC         1
#define E_INV_DATA      2
#define E_INV_CONF      3
#define E_SYS_ERR       4
#define E_UNKNOWN       5
#define E_UNFEASIBLE    6
```

Παράδειγμα κλήσης συνάρτησης της *libstnoc*

```
1  #include <stnoc.h>
2
3  void do_something(unsigned source, unsigned dest) {
4          unsigned result;
5          int errcode = stnoc_get_routing_path(source, dest, &result);
6          if(errcode > 0) {
7                  printf("success: path is %x\n", result);
8          } else {
9                  printf("error calling function: %s\n",
10                         stnoc_error_str(errcode)         );
11         }
12 }
```

Διάφορες βοηθητικές συναρτήσεις

```
1  int stnoc_available(void);
```

Επιστρέφει έναν κωδικό σφάλματος, εάν ένα *Spidergon STNoC* σύστημα δεν μπορεί να εντοπιστεί ή υπάρχει ένα σφάλμα στην βιβλιοθήκη *libstnoc* ή στον kernel driver.

```
1 int stnoc_num_ni(void);
```

Επιστρέφει τον αριθμό των Network Interfaces που υπάρχουν στο Network on Chip. Κάθε διαθέσιμο Network Interface αριθμείται αυθαίρετα από το το 0 έως το NUM_NI-1. Ενδέχεται να επιστρέψει έναν κωδικό σφάλματος.

```
1 int stnoc_reachable(unsigned int node, unsigned int dest );
```

Επιστρέφει έναν κωδικό σφάλματος, εάν ο προορισμός dest δεν είναι προσβάσιμος από το node, διαφορετικά επιστρέφει ένα μη αρνητικό return value.

```
1 int stnoc_get_id(unsigned int node, char** id);
```

Γράφει στο **id ένα δείκτη για ένα string που αντιπροσωπεύει το text ID του ζητουμένου node. Αυτό το string ΔΕΝ πρέπει να γίνει deallocate από τον καλούντα της συνάρτησης. Η συνάρτηση επιστρέφει τον ίδιο δείκτη για διαφορετικές κλήσεις για το ίδιο node.

```
1 int stnoc_get_by_id(char* id);
```

Το αντίστροφο της προηγούμενης συνάρτησης.

```
1 int stnoc_qos_type(unsigned int node);
```

Επιστρέφει QOS ή MULTI_QOS ή έναν κωδικό σφάλματος.

```
1 int stnoc_routing_type(void);
```

Επιστρέφει SPIDERGON_ROUTING ή SOURCE_ROUTING ή έναν κωδικό σφάλματος.

```
1 int stnoc_ni_flags(unsigned int node);
```

Επιστρέφει ένα bitmap από flags για ένα δεδομένο Network Interface, ή έναν κωδικό σφάλματος.

```
1 // ni flags
2 #define NONE                    0x0
3 #define MASTER                  0x1
4 #define SLAVE                   0x2
5 #define PROG_ROUTING            0x4
6 #define QOS                     0x8
7 #define MULTI_QOS               0x10
```

```
1 int stnoc_dump(unsigned int node);
2 int stnoc_dump_all(void);
```

Έξοδος στο stdout πληροφοριών σχετικά με τον κόμβο node, ή όλους τους κόμβους.

```
1 const char* stnoc_error_str(int err);
```

Επιστρέφει ένα const char* με μια περιγραφή υπό μορφή κειμένου (string) ενός συγκεκριμένου κωδικού σφάλματος.

Συναρτήσεις get/set τιμών προγραμματιζόμενων registers

```
1  int stnoc_set_qos(      unsigned int node,      unsigned int dest,
2                          unsigned int fba_thr,   unsigned int priority   )
                               ;
3
4  int stnoc_set_spidergon_routing (               unsigned int node,
5                          unsigned int dest,      unsigned int dir1,
6                          unsigned int dir2,      unsigned int destid     )
                               ;
7
8  int stnoc_set_spidergon_routing_dir1(           unsigned int node,
9                          unsigned int dest,      unsigned int dir1       )
                               ;
10
11 int stnoc_set_spidergon_routing_dir2(           unsigned int node,
12                         unsigned int dest,      unsigned int dir2       )
                               ;
13
14 int stnoc_set_spidergon_routing_destid(         unsigned int node,
15                         unsigned int dest,      unsigned int destid     )
                               ;
16
17 int stnoc_set_routing_path(                     unsigned int node,
18                         unsigned int dest,      unsigned int value      )
                               ;
```

Οικογένεια από συναρτήσεις που χρησιμοποιούνται για να οριστεί μια προγραμματιζόμενη τιμή για ένα ζεύγος source, destination· επιστροφή μη αρνητικής τιμής σε περίπτωση επιτυχίας.

```
1  int stnoc_get_qos(      unsigned int node,      unsigned int dest,
2                          unsigned int *fba_thr,  unsigned int *prio      )
                               ;
3
4  int stnoc_get_qos_all(  unsigned int node,      unsigned int **dests,
5                          unsigned int **thrs,    unsigned int **prios);
6
7  int stnoc_get_spidergon_routing(                unsigned int node,
```

```
8                          unsigned int dest,      unsigned int *dir1,
9                          unsigned int *dir2,     unsigned int *destid    )
                               ;
10
11 int stnoc_get_spidergon_routing_all(            unsigned int node,
12                         unsigned int **dests,   unsigned int **dir1,
13                         unsigned int **dir2,    unsigned int **destid   )
                               ;
14
15 int stnoc_get_routing_path(                     unsigned int node,
16                         unsigned int dest,      unsigned int *path      )
                               ;
17
18 int stnoc_get_routing_path_all(                unsigned int node,
19                         unsigned int **dests,   unsigned int **paths    )
                               ;
```

Αυτή η οικογένεια συναρτήσεων γράφει τις ζητούμενες τιμές στους δείκτες που παρέχονται ως είσοδο. Οι _all εκδόσεις των συναρτήσεων, απαιτούν ένα δείκτη σε έναν δείκτη, όπου η συνάρτηση θα κατανείμει μνήμη για πολλαπλές τιμές. Η τιμή return που επιστρέφεται από τις συναρτήσεις αυτές είναι ο αριθμός των εγγραφών που έχουν διαβαστεί. Όποια κατανεμημένη μνήμη πρέπει να απελευθερωθεί από τον καλούντα της συνάρτησης.

```
1 int stnoc_get_alt_paths(unsigned int node, unsigned int dest, unsigned
      int **paths      );
```

Για ένα ζεύγος source, destination εκχωρεί αρκετή μνήμη στην διεύθυνση που παρέχεται ως είσοδο (**paths) και γράφει μία λίστα εναλλακτικών paths που μπορούν να χρησιμοποιηθούν για *Source Routing*, διαμορφωμένα όπως γίνονται δεκτά από την `int stnoc_set_routing_path(unsigned, unsigned, unsigned)`. Η allocated μνήμη πρέπει να απελευθερωθεί από τον καλούντα της συνάρτησης.

Generation & εναλλαγή προφίλ QoS

```
1 STNOC_PROFILE* new_stnoc_profile (void);
2 void stnoc_profile_free(STNOC_PROFILE *prof);
```

Εκχωρεί και ελευθερώνει χώρο για μια μεταβλητή STNOC_PROFILE, η οποία επιτρέπει την αποθήκευση QoS ρυθμίσεων για μεταγενέστερη εναλλαγή μεταξύ διαφορετικών use cases κατά το run time. Παρατηρούμε ότι επί του παρόντος η STNOC_PROFILE κρατάει μόνο τις ρυθμίσεις QoS, αλλά μελλοντικά αναμένεται να καταχωρεί οποιουδήποτε είδους ρυθμίσεις μπορούν να αλλάξουν κατά το run time για διαφορετικά use cases, π.χ. routing.

```
1 int stnoc_profile_copy(STNOC_PROFILE *prof);
```

Αποθηκεύει τις τρέχουσες ρυθμίσεις QoS στην STNOC_PROFILE που καθορίζεται από την παράμετρο *prof.

```
1 int stnoc_profile_apply(STNOC_PROFILE *prof);
```

Εφαρμόζει τις ρυθμίσεις QoS που αποθηκεύτηκαν προηγουμένως στην STNOC_PROFILE που καθορίζεται από την παράμετρο *prof.

```
1 STNOC_CONSTRAINTS* new_stnoc_constraints(void);
2 void stnoc_constraints_free(STNOC_CONSTRAINTS *constr);
```

Εκχωρεί και ελευθερώνει χώρο για μια μεταβλητή STNOC_CONSTRAINTS, η οποία επιτρέπει την αποθήκευση εύρους ζώνης και άλλων απαιτήσεων, με στόχο τη δημιουργία ενός προφίλ που εκπληρώνει αυτούς τους περιορισμούς.

```
1 int stnoc_constraints_bw(STNOC_CONSTRAINTS *constr, unsigned int source,
      unsigned int dest, unsigned int value);
```

Προσθέτει μια απαίτηση εύρους ζώνης στην STNOC_CONSTRAINTS *constr, μεταξύ των source, dest. Δεδομένου ότι η libstnoc δεν είναι ενήμερη για το διαθέσιμο throughput στο εσωτερικό του Network on Chip, αυτή αναπαρίσταται ως ένας αριθμός από συνολικά 20.000 «μετοχές» του εύρους ζώνης στο Network on Chip. Π.χ. μια τιμή 10000 (από 20000) αντιπροσωπεύει το 50% του διαθέσιμου εύρους ζώνης.

```
1 int stnoc_constraints_pksize(STNOC_CONSTRAINTS *con, unsigned nd,
      unsigned value);
```

Δεδομένου ότι η λειτουργία του πρωτοκόλλου FBA επηρεάζεται από το μέγεθος των παραγόμενων πακέτων του traffic, το λογισμικό πρέπει να ενημερώσει την libstnoc για αυτό, προκειμένου να δημιουργηθούν έγκυρες ρυθμίσεις FBA.

```
1 int stnoc_gen_profile(STNOC_CONSTRAINTS *constr, STNOC_PROFILE *prof);
```

Δημιουργία ενός STNOC_PROFILE με QoS ρυθμίσεις που πληρούν την `STNOC_CONSTRAINTS *constr`.

```
1 int stnoc_max_bw(unsigned source, unsigned dest, STNOC_CONSTRAINTS *c,
      unsigned *bw);
```

Για τις τρέχουσες ρυθμίσεις QoS, γράφει στην unsigned *bw το θεωρητικό μέγιστο εγγυημένο εύρος ζώνης μεταξύ των source, dest. Η STNOC_CONSTRAINTS *c απαιτείται ως παράμετρος, γιατί η έξοδος αυτή εξαρτάται από το μέγεθος των παραγόμενων πακέτων.

Πλήρες παράδειγμα χρήσης της libstnoc Λαμβάνοντας υπόψιν ένα τυπικό παράδειγμα, μια εφαρμογή που αποκωδικοποιεί ένα αρχείο ήχου MP3 μπορεί να ζητήσει εγγυήσεις εύρους ζώνης κατά την εκτέλεση σε ένα σύστημα που βασίζεται σε *Spidergon STNoC*. Τυπικά, ο κώδικας θα πρέπει να αντιμετωπίσει μια σειρά προβλημάτων. Πρώτον, θα πρέπει να υπολογίσει τις απαιτούμενες τιμές *QoS* για την *Spidergon STNoC QoS* υπηρεσία και, αφετέρου, τις διευθύνσεις των προγραμματιζόμενων registers των εμπλεκόμενων Network Interfaces. Συνήθως, οι εν λόγω διευθύνσεις, που θα ήταν hard coded ως σταθερές από τον developer στον κώδικα, θα πρέπει να γίνουν map σε διευθύνσεις εντός του χώρου εικονικών διευθύνσεων μιας εφαρμογής και η πρόσβαση να γίνει κατά προκαθορισμένου format. Κάθε ένα από αυτά τα ζητήματα περιλαμβάνουν αρκετές προκλήσεις, επιπλέον του ότι πρέπει να επανα-υλοποιούνται για κάθε διαφορετικό σύστημα, software platform και εφαρμογή.

Αντιθέτως οι εφαρμογές που διαχειρίζονται τις υπηρεσίες *Spidergon STNoC* με τη βοήθεια της βιβλιοθήκης *libstnoc* επιδεικνύουν εξαιρετική απλότητα· ο προγραμματιστής μιας εφαρμογής μπορεί να χρησιμοποιεί ένα API απλών συναρτήσεων χωρίς να ανησυχεί για χαμηλού επιπέδου λεπτομέρειες (όπως φαίνεται στο listing 6.1). Μπορούμε εύκολα να οραματιστούμε εφαρμογές που χρησιμοποιούν αυτές τις συναρτήσεις με πιο δυναμικούς τρόπους, όπως για παράδειγμα για την προσαρμογή του απαιτούμενου εύρους ζώνης, σύμφωνα με το bit-rate των streams που αναπαράγονται.

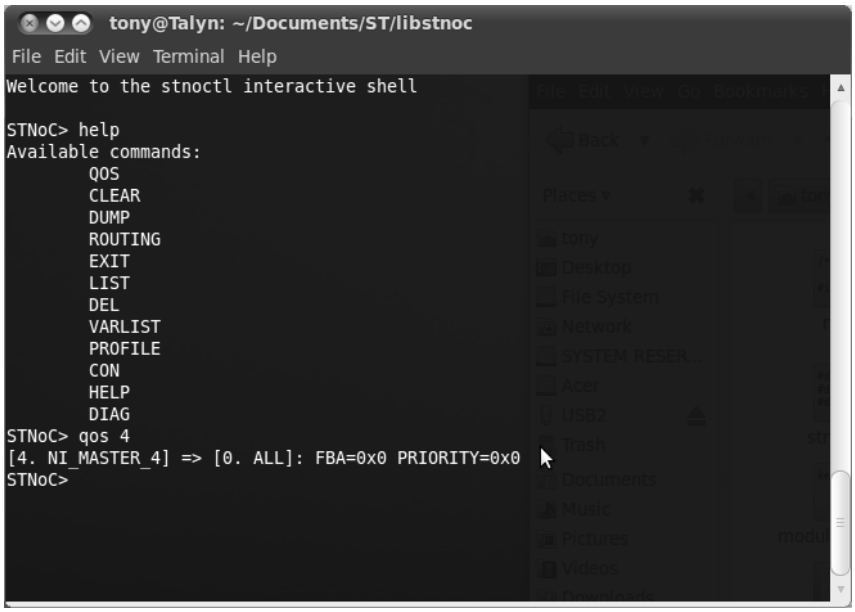
Listing 6.1: Παράδειγμα κώδικα που αξιοποιεί το API της *libstnoc*

```
1  #include "stnoc.h"
2
3  #define DSP_AUDIO  stnoc_get_by_id("NI_INIT_DSP")
4  #define DSP_BUFFER stnoc_get_by_id("NI_TARGET_MEM0")
5
6  // switching between two profiles
7  STNOC_PROFILE *old, *new;
8
9  int play(int someparameter)
10 {
11     stnoc_profile_apply(new);
12     /*  Start playing code */
13 }
14
15 int stop(int someparameter)
16 {
17     stnoc_profile_apply(old);
18     /* Stop playing code */
19 }
20
21 void main(void)
22 {
23     STNOC_CONSTRAINTS *mp3play = new_stnoc_constraints();
24
25     // backup old configuration
26     old = new_stnoc_profile();
27     stnoc_profile_copy(old);
```

```
28
29      // mp3 playing requirements in bandwidth
30      stnoc_constraints_bw(mp3play, DSP_BUFFER, DSP_AUDIO, 8000);
31      // package size (in bytes) of generated traffic
32      stnoc_constraints_pksize(mp3play, DSP_BUFFER, 16);
33
34      // generate a valid configuration
35      new = new_stnoc_profile();
36      stnoc_gen_profile(mp3play, new);
37      // get rid of unused variable
38      stnoc_constraints_free(mp3play);
39
40      /*  Implement application code */
41
42      // cleanup
43      stnoc_profile_free(old);
44      stnoc_profile_free(new);
45  }
```

6.2.3 Το εργαλείο stnoctl

Μαζί με την *libstnoc* περιλαμβάνεται και το εργαλείο *stnoctl*, το οποίο υλοποιεί ένα φιλικό command line interface για την λειτουργικότητα της *libstnoc*, χρήσιμο για την ανάπτυξη και το debugging ενός *Spidergon STNoC* συστήματος. Το *stnoctl* διαθέτει ένα κέλυφος (σχήμα 6.3), από το οποίο μπορεί να δεχτεί μια σειρά από εντολές για την διαχείριση των παραμέτρων του Network on Chip.



Σχήμα 6.3: Το interactive κέλυφος του εργαλείου stnoctl

Οι εντολές που δέχεται το κέλυφος του *stnoctl* είναι case insensitive.

list Δίνει μια λίστα με τα διαθέσιμα Network Interface στο Network on Chip, με έναν αύξων αριθμό στο καθένα. Στις εντολές που παίρνουν ένα Network Interface ως παράμετρο, μπορούμε να χρησιμοποιήσουμε το όνομα του Network Interface ή αυτόν τον αύξων αριθμό.

qos **`source`** Δίνει τις QoS τιμές για το Network Interface `source`.

qos **`source destination`** Για ένα *Multi QoS* σύστημα, δίνει τις QoS τιμές από το Network Interface `source` προς το `destination`.

qos **`source [destination] fba priority`** Ορίζει τις QoS τιμές για το Network Interface `source`. Για ένα *Multi QoS* σύστημα απαιτούμε και το `destination` για το Network Interface προορισμού που ορίζουμε τις QoS παραμέτρους.

routing **`source [destination]`** Δίνει τις τρέχουσες παραμέτρους για το routing από το Network Interface `source` προς το `destination`, ή για όλους τους πιθανούς προορισμούς.

routing **`source destination dir1 dir2 destid`** Για ένα σύστημα *Spidergon STNoC* που χρησιμοποιεί *Spidergon Routing*, ορίζουμε τις παραμέτρους που ορίζουν το routing μεταξύ των Network Interface `source` και `destination`.

routing **`source destination path`** Για ένα *Spidergon STNoC* σύστημα που χρησιμοποιεί *Source Routing*, ορίζουμε το `path` που ορίζει το routing μεταξύ των Network Interface `source` και `destination`.

routing **`source destination`** **select** Για ένα *Spidergon STNoC* σύστημα που χρησιμοποιεί *Source Routing*, λαμβάνουμε μια λίστα έγκυρων paths για το routing μεταξύ των Network Interface `source` και `destination`, το καθένα με έναν αύξων αριθμό.

routing **`source destination`** **select** **`id`** Για ένα *Spidergon STNoC* σύστημα που χρησιμοποιεί *Source Routing*, επιλέγουμε ένα από τα έγκυρα paths για το routing μεταξύ των Network Interface `source` και `destination`, με βάση τον αύξων αριθμό του `id`.

dump **`[node]`** Τυπώνει στην έξοδο όλες τις διαθέσιμες πληροφορίες για το Network on Chip, ή για κάποιο συγκεκριμένο Network Interface που δίνεται από την παράμετρο `node`.

diag Ένα απλό διαγνωστικό που ελέγχει τους προγραμματιζόμενους registers του *Spidergon STNoC*, αν μπορούν να αναγνωστούν και να γραφτούν κανονικά (σχήμα 6.4).

varlist Μια λίστα των προσωρινών μεταβλητών που κρατάει το *stnoctl* για λογαριασμό του χρήστη· αυτές οι μεταβλητές μπορούν να περιλαμβάνουν *profiles* με παραμέτρους QoS, ή λίστες περιορισμών *constraints* από τις οποίες μπορούμε να κάνουμε generate ένα *profile*.

del var Διαγραφή μίας των προσωρινών μεταβλητών που κρατάει το *stnoctl* για λογαριασμό του χρήστη.

con constraints source target bwratio Δημιουργεί ή προσθέτει στην λίστα περιορισμών (constraints) `constraints`, την απαίτηση για εγγυημένο bandwidth, από το Network Interface `source` πρός το `target`. Το bandwidth που θα δεσμευτεί αντιστοιχεί με `bwratio`/20000% του διαθέσιμου throughput στο Network Interface προορισμού.

con constraints pksize node pksize Ενημερώνει την λίστα προορισμών με την πληροφορία ότι το IP core πίσω από το Network Interface `node` χρησιμοποιεί για την κίνηση του στο interconnect μέγεθος πακέτου των `pksize` bytes. Αυτό επιτρέπει πιο ακριβής λειτουργία της δυνατότητας υπολογισμού ρυθμίσεων QoS.

profile gen constraints profile Από μία λίστα περιορισμών που έχουμε ορίσει `constraints`, ζητάμε να δημιουργηθεί το προφίλ ρυθμίσεων QoS `profile` που ικανοποιεί αυτούς του περιορισμούς, εάν αυτό υπάρχει.

profile apply profile Εφαρμόζει τις παραμέτρους QoS που έχουν αποθηκευτεί στο προφίλ `profile`.

profile copy profile Αποθηκεύει τις τρέχουσες παραμέτρους του Network on Chip για το QoS στην μεταβλητή `profile`, για επαναφορά αργότερα.

clear Καθαρισμός της οθόνης.

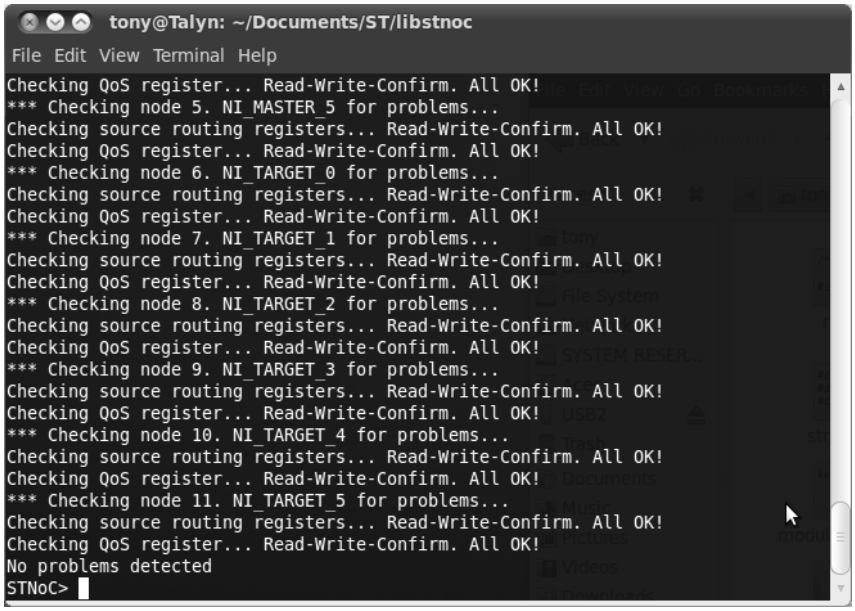
help Εκτύπωση των διαθέσιμων εντολών του *stnoctl*.

exit Έξοδος από το *stnoctl*.

6.3 Χρήση του stack σε Android

Για να χρησιμοποιηθεί το λογισμικό σε Android, η διαδικασία δεν διαφέρει πάρα πολύ, καθώς το Android είναι Linux-based. Εντούτοις, το Android περιλαμβάνει διαφορετικό user space και χρησιμοποιεί δικό του build system. Όσον αφορά τον πυρήνα πάντως, η διαδικασία είναι ακριβώς η ίδια.

Για τα υψηλότερου επιπέδου components του λογισμικού, χρησιμοποιούμε το build system του Android. Αρκεί να τοποθετήσουμε ένα αντίγραφο του πηγαίου κώδικα των απαραίτητων components υπό τον κατάλογο `vendor` του πηγαίου κώδικα του Android



Σχήμα 6.4: Η διαγνωστική εντόλή `diag` του stnoctl

ή σε υποκατάλογο του. Αυτά περιλαμβάνουν την *libstnoc* στον υποκατάλογο `libstnoc`, τον JNI wrapper της *libstnoc* στον υποκατάλογο `libstnoc_jni`, και τέλος η υλοποίηση του ContentProvider μας στον υποκατάλογο `StNocContentProvider`.

Όταν κάνουμε rebuild το image του Android, αυτό θα περιλαμβάνει πλέον το λογισμικό μας, και το μόνο που μένει είναι να κάνουμε boot αυτό το image με έναν πυρήνα στον οποίο έχουμε συμπεριλάβει και τον driver μας. Ανοίγοντας ένα κέλυφος μάλιστα στην συσκευή ή emulator Android, μπορούμε να χρησιμοποιήσουμε και το εργαλείο stnoctl όπως και σε ένα vanilla Linux σύστημα (σχήμα 6.5).

6.3.1 Ο STNoC content provider

Σε ένα image όπου περιλαμβάνεται το λογισμικό μας, κάποια εφαρμογή Android μπορεί να διαχειριστεί τις προγραμματιζόμενες παραμέτρους του *Spidergon STNoC* μέσω του Content Provider interface του Android. Σε αυτό το τμήμα θα τεκμηριώσουμε τα URLs που χρησιμοποιούνται και τις στήλες των πινάκων που επιστρέφουν τα δεδομένα σε μια Network on Chip aware εφαρμογή. Οι πίνακες μπορούν να ενημερωθούν με νέες παραμέτρους, με τα συνηθισμένα ContentProvider interfaces του Android.

content://com.st.stnoc/parameters

Γενικές παραμέτρους με τις οποίες έχει ρυθμιστεί το Network on Chip. Αυτός ο πίνακας επιστρέφει μονάχα μία γραμμή.

_id Πάντα μηδέν.

Σχήμα 6.5: Το εργαλείο stnoctl σε περιβάλλον Android

routing Το είδος του routing, *Spidergon Routing* ή *Source Routing*.

content://com.st.stnoc/nis[/#id]

Ο πίνακας που επιστρέφει από αυτό το URL, περιλαμβάνει ένα Network Interface ανά γραμμή (ή μόνο το Network Interface με αύξων αριθμό `#id`), με τις εξής παραμέτρους του κάθε Network Interface.

_id Αύξων αριθμός του Network Interface.

routing Εάν το Network Interface έχει ενεργοποιημένο το service για προγραμματιζόμενο routing.

qos Εάν το Network Interface έχει ενεργοποιημένο το QoS service, και εάν αυτό είναι *Multi QoS* ή *Single QoS*.

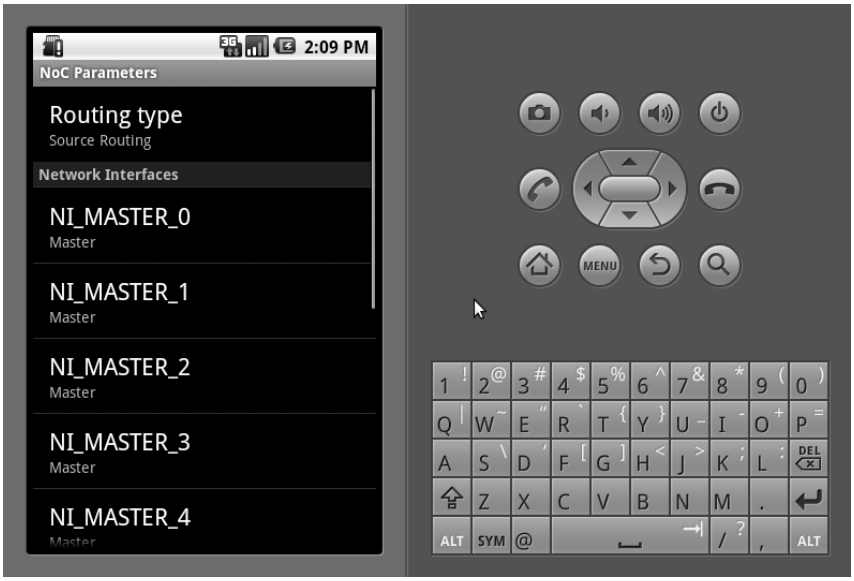
type Εάν το Network Interface εξυπηρετεί ένα IP core που είναι *Master* ή *Slave*.

content://com.st.stnoc/nis/#id/routing

Για το Network Interface με αύξων αριθμό `#id`, οι παράμετροι του προγραμματιζόμενου routing, με ένα προσβάσιμο προορισμό ανά γραμμή. Για *Spidergon Routing*:

_id Ο αύξων αριθμός του προορισμού.

dir1 Η *Spidergon Routing* κατεύθυνση που έχει ρυθμιστεί για τον προορισμό.



Σχήμα 6.6: Η εφαρμογή NocDrod μέσα απο την οποία μπορούμε να έχουμε πρόσβαση και να επεξεργαστούμε τις τιμές routing και QoS μέσω ενός GUI

dir2 Η *Spidergon Routing* κατεύθυνση που έχει ρυθμιστεί για τον προορισμό μετά από ιεραρχικό σύνδεσμο.

destid Το *Spidergon Routing* destID που αντιστοιχεί στον προορισμό.

Για *Source Routing*:

_id Ο αύξων αριθμός του προορισμού.

path Το *Source Routing* path που έχει ρυθμιστεί για τον προορισμό.

content://com.st.stnoc/nis/#id/qos

Για το Network Interface με αύξων αριθμό `#id`, οι παραμέτροι του προγραμματιζόμενου service για QoS, με έναν προσβάσιμο προορισμό ανά γραμμή εάν πρόκειται για *Multi QoS*.

_id Ο αύξων αριθμός του προορισμού για *Multi QoS*, διαφορετικά πάντα μηδέν για *Single QoS*.

fba Η αντίστοιχη `FBA` τιμή.

prio Η αντίστοιχη `priority` τιμή.

6.3.2 User Interface

Διαθέτουμε επίσης και την εφαρμογή `NocDroid` που επιτρέπει εύκολα σε ένα σύστημα Android σε λειτουργία, που βασίζεται σε ένα System on Chip που περιλαμβάνει μια *Spidergon STNoC* υλοποίηση, να επεξεργαστούμε τις τιμές routing και QoS μέσω ενός application με GUI. Για να το συμπεριλάβουμε στο image μας, αρκεί να τοποθετήσουμε ένα αντίγραφο του πηγαίου κώδικα του, στον κατάλογο `vendor` του πηγαίου κώδικα του Android, και να κάνουμε build το image μας, όπως κάνουμε και με τα υπόλοιπα software componenets του λογισμικού μας. Το αποτέλεσμα είναι να ενσωματωθεί στο image μας, στο μενού των εφαρμογών, και η εφαρμογή *NocDroid* η οποία περιλαμβάνει μια λίστα από Network Interface τα οποία μπορούν να επιλεγούν από τον χρήστη για να αποκτήσει πρόσβαση στις routing και QoS τιμές, και να τις επεξεργαστεί.

7 Αποτελέσματα

Η πλήρης αξιοποίηση των δυνατοτήτων run time επαναπρογραμματισμού των σύγχρονων, Network on Chip based, λύσεων interconnect, όπως το *Spidergon STNoC* μπορούν να παρουσιάσουν διάφορες προκλήσεις για το integration, το portability, και την δυνατότητα συντήρησης (maintainability), ιδιαίτερα σε υψηλού επιπέδου, cross-platform κώδικα. Σε αυτή την εργασία επιδείξαμε τη χρήση του λογισμικού μας, το οποίο επιτρέπει στον σχεδιαστή του λογισμικού ενός συστήματος να αξιοποιήσει τα platform services του *Spidergon STNoC* χωρίς να φθείρεται η ικανότητα για τη δημιουργία αξιόπιστων cross platform λύσεων, εντός των ήδη στενών χρονικών περιορισμών της ανάπτυξης των σημερινών συστημάτων. Η software λύση μας βελτιώνει την δυναμική προσαρμογή πόρων του Network on Chip, διατηρώντας ταυτόχρονα τη συμβατότητα με τα υπάρχοντα πρότυπα και tool flows.

Συνοπτικά παρέχουμε στον σχεδιαστή ενός συστήματος *Spidergon STNoC*:

- Τα απαραίτητα εργαλεία για generate ενός Linux driver για ένα δεδομένο Network on Chip βασισμένο σε τεχνολογία *Spidergon STNoC*.
- Την βιβλιοθήκη *libstnoc*, για χρήση των δυνατοτήτων του Network on Chip που εκθέτονται από τον Linux driver, μέσω ενός απλού C API που επιτρέπει το portability σε μελλοντικές πλατφόρμες, και τον ίδιο κώδικα να επαναχρησιμοποιηθεί μεταξύ διαφορετικών Network on Chip.
- Εργαλεία διαχείρισης του Network on Chip, όπως το *stnoctl* για Linux επιτρέπουν την εύκολη διαχείριση ενός *Spidergon STNoC* συστήματος σε λειτουργία.
- Android *ContentProvider* based interface στις δυνατότητες του *Spidergon STNoC*, και επίσης GUI για Android που επιδεικνύει αυτήν την λειτουργικότητα.

7.1 Μελλοντική Εργασία και Επεκτάσεις

Το software μας επιτρέπει εύκολη εκμετάλλευση των σημερινών δυνατοτήτων του *Spidergon STNoC*, όμως καθώς το *Spidergon STNoC* εξελίσσεται, έτσι και το software που αναπτύξαμε μπορεί μελλοντικά να ακολουθήσει αυτές τις εξελίξεις και να εκθέσει περισσότερα services στον σχεδιαστή του software ενός System on Chip. Επιπλέον, το υπάρχον πλαίσιο από services θα μπορούσε να βελτιωθεί στο επίπεδο του interface που παρέχουμε. Στο μέλλον θα μπορούσαμε να δούμε υλοποιήσεις όπως:

- Υποστήριξη για μελλοντικές *Spidergon STNoC* υπηρεσίες, π.χ. διάγνωσης και υπηρεσίες παρακολούθησης της κυκλοφορίας του Network on Chip.
- Περισσότερο «informed» μέθοδοι για την παραγωγή προφίλ QoS από ένα σύνολο περιορισμών μπορούν να διερευνηθούν, με ευρύτερη υποστήριξη περιορισμών (π.χ. latency ή ενεργειακής απόδοσης).
- Συνδυασμός των primitives μιας υπηρεσίας παρακολούθησης της κυκλοφορίας και της υπηρεσίας QoS, για αυτόματη δυναμική προσαρμογή των παραμέτρων του Network on Chip με βάση τον συνωστισμό στα links του interconnect.
- Επιπλέον, στην τρέχουσα υλοποίηση εφαρμογές που απαιτούν να επωφεληθούν από services του *Spidergon STNoC* θα πρέπει να χαίρουν εμπιστοσύνης για πρόσβαση στις ρυθμίσεις του Network on Chip· μια υλοποίηση από security policies θα μπορούσε να επιτρέψει σε ένα unpriviledged process για προγραμματισμό του Network on Chip χωρίς να θέτει σε κίνδυνο την ασφάλεια και σταθερότητα του συστήματος.

Βιβλιογραφία

[1] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC.* CRC Press, 2008.

[2] L. Benini, "Application specific NoC design," in *Proceedings of the conference on Design, Automation and Test in Europe*, pp. 491–495, 2006.

[3] "From bus and crossbar to network-on-chip," tech. rep., Arteris S.A., 2009.

[4] *AMBA AXI Protocol Version: 2.0 Specification*, 2010.

[5] J. Corbet, A. Rubini, and G. Kroah-Hartman, *Linux Device Drivers.* O'Reilly, third edition ed., 2005.

[6] P. Mochel, "The sysfs filesystem," *Ottawa Linux Symposium*, 2005.

[7] U. Drepper, "How to write shared libraries," 2006.

[8] U. Drepper, "Good practices in library design, implementation, and maintenance," 2002.

[9] M. A. Trick, "A tutorial on integer programming." `http://mat.gsia.cmu.edu/orclass/integer/integer.html`.

[10] "Android developer's guide." `http://developer.android.com/guide/index.html`.

[11] "Android platform developer's guide." `http://source.android.com/porting/`.

Παραρτήματα

Α Παρουσίαση της πτυχιακής

Software για το Spidergon STNoC

- Μοτάκης Αντώνιος A.M. 899
- Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων, Τ.Ε.Ι. Κρήτης
- Επιβλέπων καθηγητής: Κορνάρος Γεώργιος

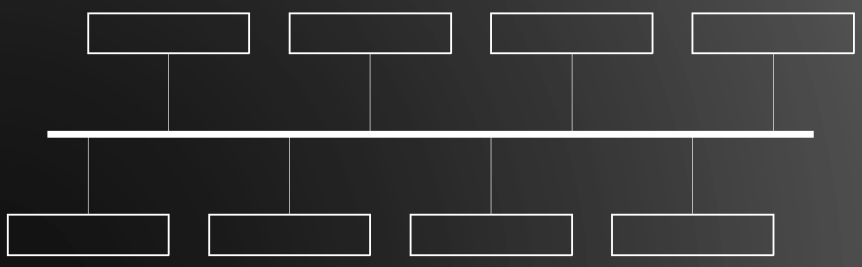
Linux driver generation

Portable βιβλιοθήκη libstnoc

Εργαλεία για Linux & Android

Εισαγωγή

- Σε ένα τυπικό System on Chip για τις ανάγκες επικοινωνίας μεταξύ των IP Cores χρησιμοποιούνται διασυνδέσεις τύπου bus.



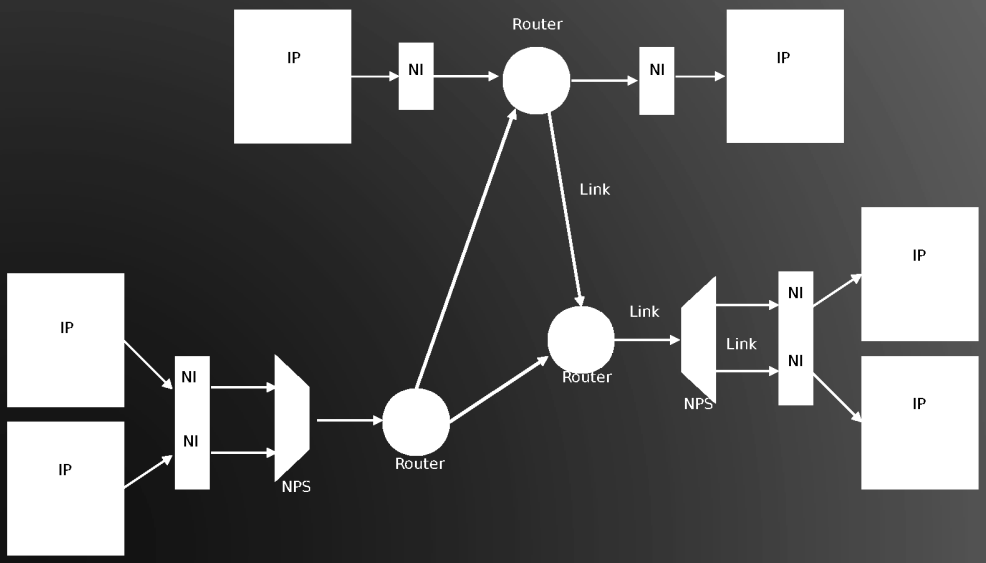
Εισαγωγή

- Στις διασυνδέσεις τύπου bus, όλοι οι χρήστες της διασύνδεσης μοιράζονται το bandwidth ενός link.
- Ιεραρχικά buses και διασυνδέσεις τύπου crossbar χρησιμοποιούνται καθώς οι ανάγκες σε bandwidth και latency αυξάνονται.
- Αυτό προκαλεί αυξημένη πολυπλοκότητα για τον σχεδιαστή του συστήματος, και από άποψης καλωδίων πάνω στο τσιπ.

Εισαγωγή

- Για τους λόγους αυτούς, η βιομηχανία κινείται προς λύσεις “Network on Chip” για τα SoC.
- Η κίνηση από ένα IP Core διασπάται σε πακέτα από ένα Network Interface.
- Γίνεται inject σε ένα δίκτυο από routers, όπου θα μετακινηθεί από router σε router.
- Όταν φτάσει στον προορισμό του, ξανά συναρμολογείται από το Network Interface εξόδου.

Εισαγωγή

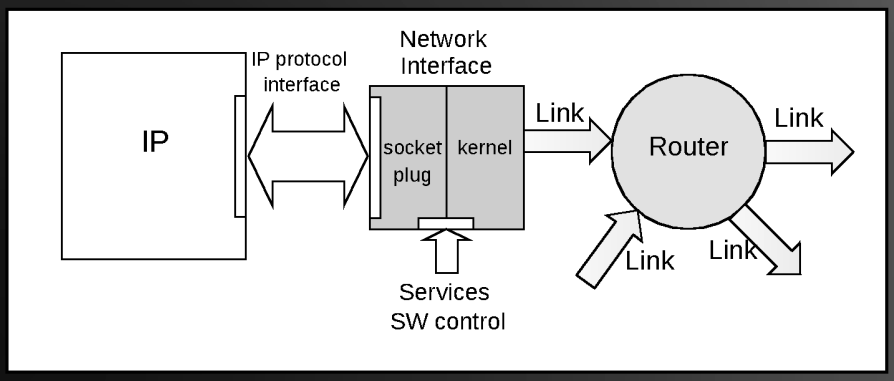


Εισαγωγή

- Το Spidergon STNoC είναι η τεχνολογία Network on Chip της STMicroelectronics.
- Αναπτύσεται στα εργαστήρια AST, στο τμήμα R&D της εταιρίας στην Grenoble της Γαλλίας.

Services του Spidergon STNoC

- Το Spidergon STNoC υλοποιεί προγραμματιζόμενα services στα άκρα του δικτύου· στο σημείο του Network Interface.



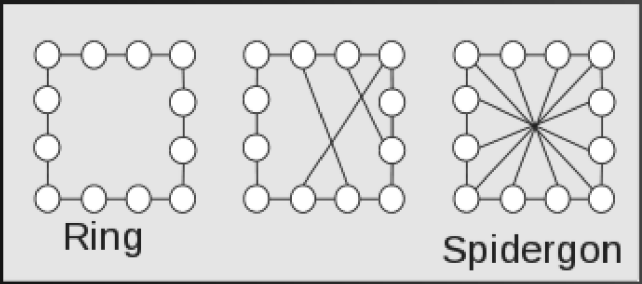
Services του Spidergon STNoC

- Προγραμματιζόμενο routing.
- Quality of Service.
- Security

- Κάθε service μπορεί να ενεργοποιηθεί και προγραμματίζεται σε κάθε Network Interface χωριστά.

Προγραμματιζόμενο routing

- Δύο routing αλγόριθμοι routing μπορούν να επιλεχθούν κατά την σχεδίαση του συστήματος: Spidergon & Source Routing.
- Με χρήση Spidergon Routing υποστηρίζεται η οικογένεια τοπολογιών Spidergon.



Προγραμματιζόμενο routing

- Οι τοπολογίες Spidergon, βασίζονται στην τοπολογία ring, με έναν επιπλέον σύνδεσμο σε κάθε node προς το node απέναντι του.
- Το routing ορίζεται κατά την εισαγωγή του πακέτου στο NoC ως:
  - Right
  - Right Across
  - Left
  - Left Across

Προγραμματιζόμενο routing

- Με χρήση Source Routing υποστηρίζεται οποιαδήποτε τοπολογία.
- Το routing ορίζεται κατά την εισαγωγή του πακέτου, ως η διαδρομή που θα ακολουθηθεί μέσα στο NoC σε κάθε ενδιάμεσο router.
- Το routing service επιτρέπει να ορίσουμε κατά την λειτουργία του συστήματος το routing για κάθε προορισμό χωριστά.

Quality of Service

- Το QoS service υλοποιεί τον αλγόριθμο “FBA”.
- Σε κάθε router κάθε εισερχόμενος σύνδεσμος εξυπηρετείται σε στυλ *round robin*.
- Η “FBA” τιμή ενός traffic stream ορίζει για πόσα bytes θα δεσμεύσει ένα router όταν είναι η σειρά του stream να μεταδώσει δεδομένα.
- Υπάρχει και μια τιμή 2 bits προτεραιότητας.
  - Streams με υψηλότερη προτεραιότητα μεταδίδουν νωρίτερα σε κάθε router.

Quality of Service

- Οι τιμές FBA και προτεραιότητα ορίζονται στο Network Interface κατά την εισαγωγή του traffic στο NoC.
- Ορίζονται καθολικά ή για κάθε προορισμό χωριστά.
  - Single QoS και Multi QoS.

Security

- Το Security service μας επιτρέπει να φιλτράρουμε την κίνηση σε ένα Network Interface σε στυλ firewall.
- Ορίζουμε περιοχές μνήμης.
- Μπορούμε να περιορίσουμε την πρόσβαση σε αυτές της περιοχές μνήμης με βάση την πηγή που μεταδίδει τα δεδομένα.
- Δικαιώματα read – write – execute.

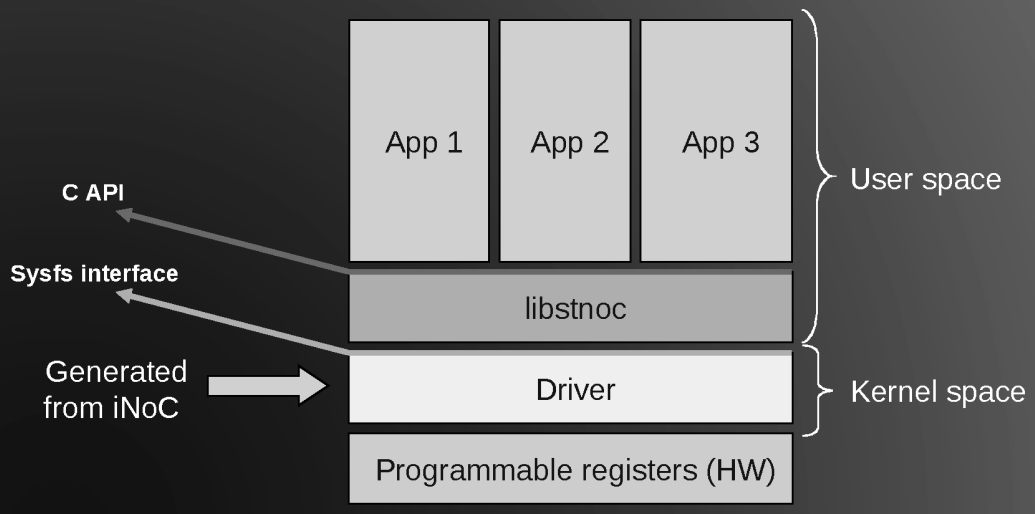
Σκοπός της εργασίας

- Υλοποίηση του λογισμικού που να επιτρέπει την εκμετάλλευση των δυνατοτήτων επανάπρογραμματισμού κατά τον χρόνο εκτέλεσης του Spidergon STNoC.
- Linux driver
  - Διαφορετικός για κάθε Spidergon STNoC εφαρμογή.
- Βιβλιοθήκη για χρήση από εφαρμογές.
  - Portable, decoupled από low level στοιχεία.

Σκοπός της εργασίας

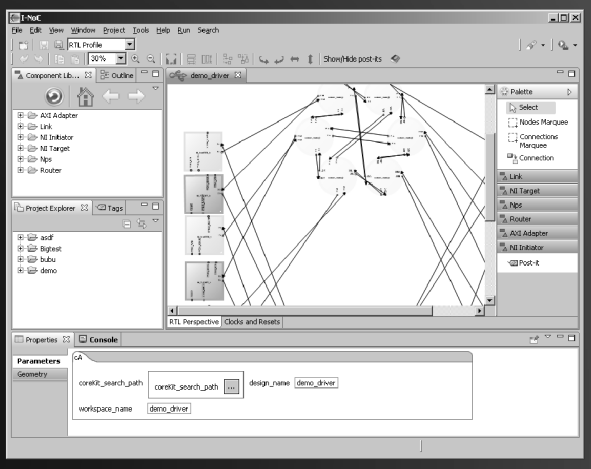
- Υλοποίηση σε περιβάλλον Linux εφαρμογής που χρησιμοποιεί τα παραπάνω στοιχεία.
  - Interface για διαχείριση και debugging ενός Spidergon STNoC συστήματος σε λειτουργία.
- Υλοποίηση εργαλείου για συστήματα Android.
  - Port των προηγουμένων σε Android.
  - GUI για διαχείριση ρυθμίσεων Routing και QoS.

Γενική αρχιτεκτονική



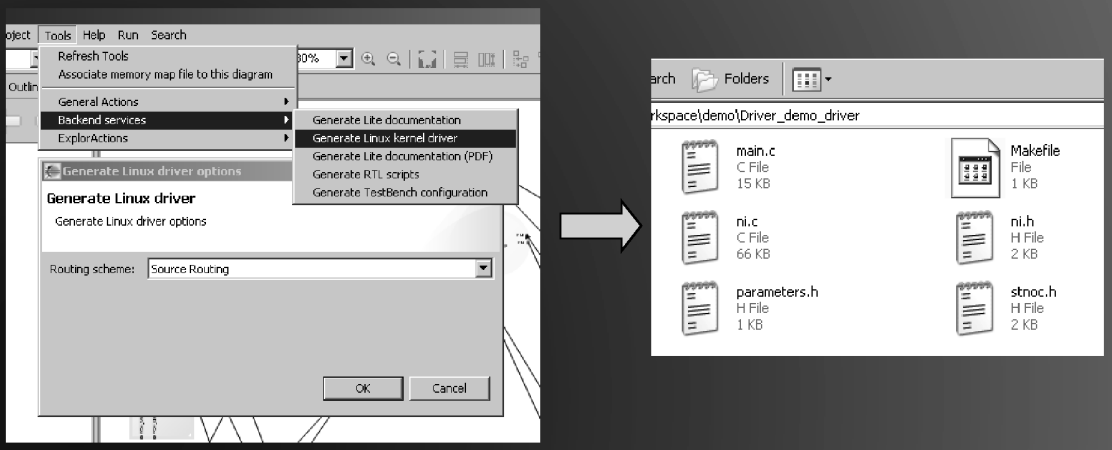
Linux Driver

- Ο σχεδιαστής έχει σχεδιάσει το NoC του χρησιμοποιώντας το εργαλείο iNoC.



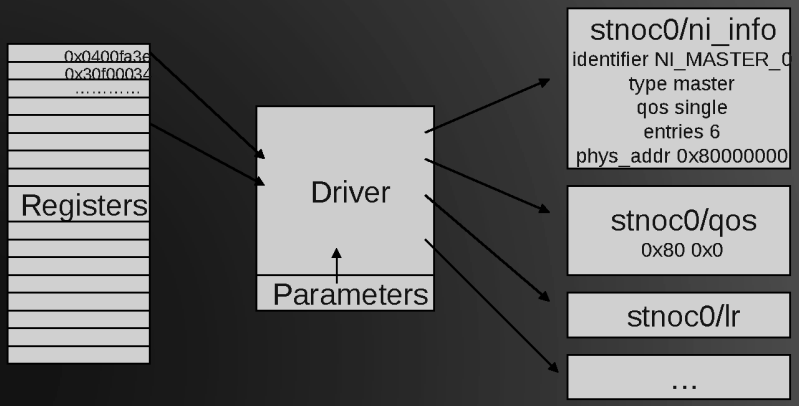
Linux Driver

- Έχουμε αναπτύξει το plugin για το iNoC που επιτρέπει να κάνουμε generate τον driver.



Linux Driver

- Ο driver διαχειρίζεται τις τιμές των προγραμματιζόμενων services και τις εκθέτει μέσω του sysfs filesystem.



Libstnoc

- Η libstnoc πατάει πάνω στο sysfs interface του driver, και παρέχει ένα portable C API.
- Get/Set για τις τιμές των προγραμματιζόμενων services.
- Έλεγχος για errors/valid τιμές. Δεν χρειάζεται χειροκίνητο error handling.
- Υψηλού επιπέδου λειτουργίες.
  - Προγραμματισμός του QoS βάση περιορισμών.

Libstnoc API

- Γενικά
  - stnoc_available()
  - stnoc_num_ni()
  - stnoc_routing_type()
  - stnoc_get_by_id(id)
  - stnoc_get_id(node, &id)
  - stnoc_reachable(node, dest)
  - stnoc_dump()
  - stnoc_dump_all()

Libstnoc API

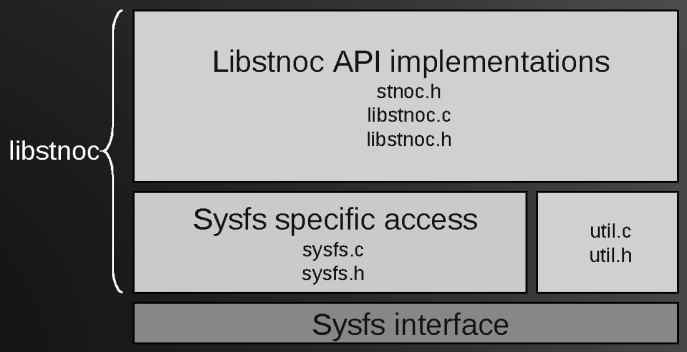
- Routing
  - Get/set τιμές για Spidergon και Source Routing.
  - stnoc_get_alt_paths(node, dest, &paths) για Source Routing.
- Quality of Service
  - Get/set τιμές για Single QoS και Multi QoS.
- Εναλλαγή profiles

Libstnoc API

- Δημιουργία profiles βάση περιορισμών.
  - new_stnoc_constraints()
  - stnoc_constraints_free(constraints)
  - stnoc_constraints_bw(constraints, source, dest, value)
  - stnoc_constraints_pksize(constraints, node, value)
  - stnoc_gen_profile(constraints, profile)

Αρχιτεκτονική της libstnoc

- Modular σχεδιασμός, διαχωρισμός του κώδικα που εξαρτάται από το λειτουργικό.



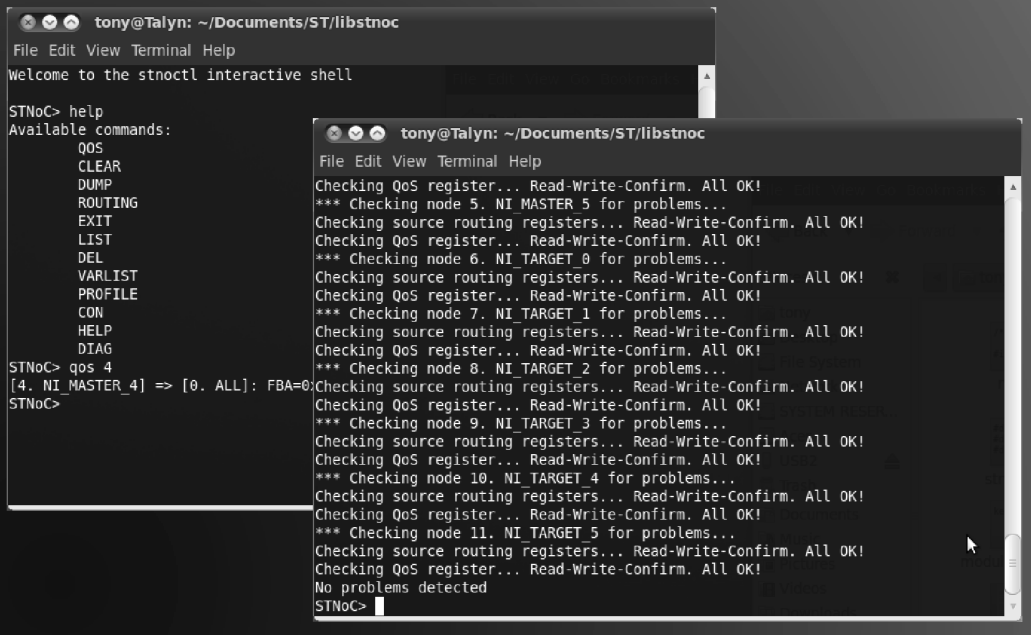
Libstnoc – Constraints

- Η libstnoc μας δίνει την δυνατότητα να ορίσουμε ανάγκες σε bandwidth μεταξύ δύο Network Interfaces.
- Κωδικοποιούμε αυτές τις ανάγκες σε μια σειρά γραμμικών ανισοτήτων.
  - Ισοδύναμες με τις ανάγκες σε bandwidth.
  - Οι μεταβλητές αντιστοιχούν στις FBA τιμές.

Libstnoc – Constraints

- Αυτές αντιστοιχούν σε ένα *Integer Programming* πρόβλημα.
- Μπορούμε να ελαχιστοποιήσουμε το σύνολο των μεταβλητών, για μικρότερο latency.
- Έχουμε υλοποιήσει έναν solver που μπορεί να βρίσκει τις τιμές FBA που χρειαζόμαστε.

Η εφαρμογή stnoctl



Η εφαρμογή stnoctl

- Command line interface για την διαχείριση των προγραμματιζόμενων services.
- Χρήση σε σύστημα σε λειτουργία.
- Υλοποίηση πάνω στην libstnoc, εκθέτει τις λειτουργίες της libstnoc.
- Μπορεί να χρησιμοποιηθεί σε διαφορετικά Spidergon STNoC συστήματα χωρίς αλλαγές στον κώδικα.

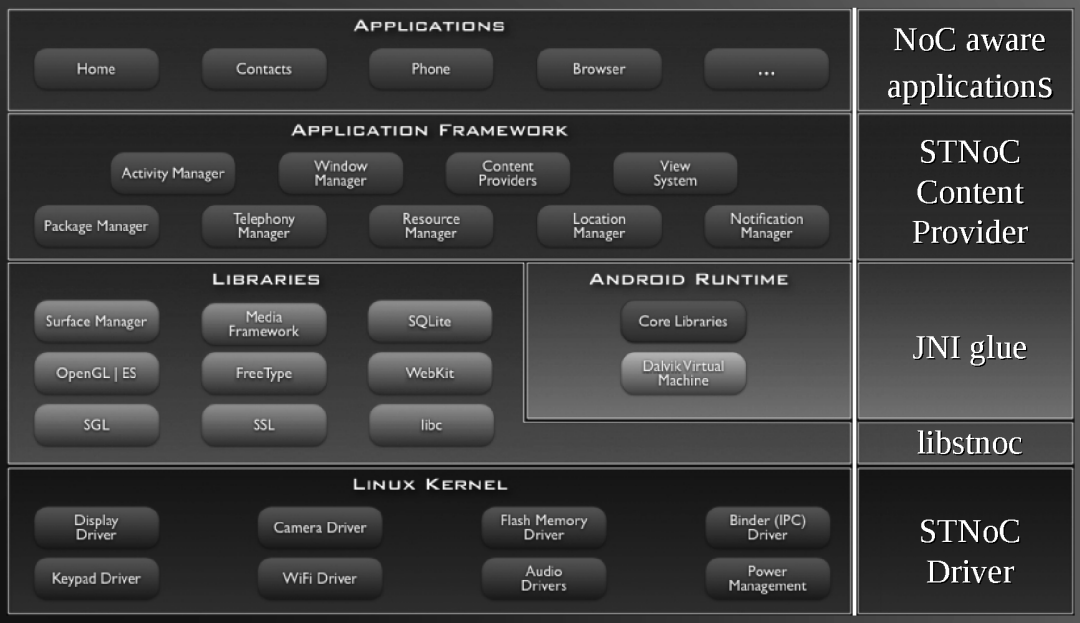
Υποστήριξη Android

- Το Android χρησιμοποιεί τον πυρήνα του Linux, οπότε χρησιμοποιεί τον ίδιο driver.
- Ωστόσο το userspace είναι διαφορετικό.
- Διαφορετική C standard library.
  - Bionic C library.
  - Port της libstnoc και του stnoctl στην Bionic.

Υποστήριξη Android

- Οι εφαρμογές Android τρέχουν πάνω σε ένα Java Virtual Machine, το Dalvik VM.
- Δεν έχουν άμεση πρόσβαση σε native κώδικα, όπως η libstnoc.

Υποστήριξη Android



Υποστήριξη Android

- Σε περιβάλλον Android οι εφαρμογές χρησιμοποιούν Content Providers για να χειριστούν δεδομένα.
- Χρησιμοποιούνται URLs, π.χ. content://com.st.stnoc/parameters
- Το Content Provider interface επιστρέφει έναν πίνακα με τα δεδομένα.
  - Για ανάγνωση, και επίσης για εγγραφή.

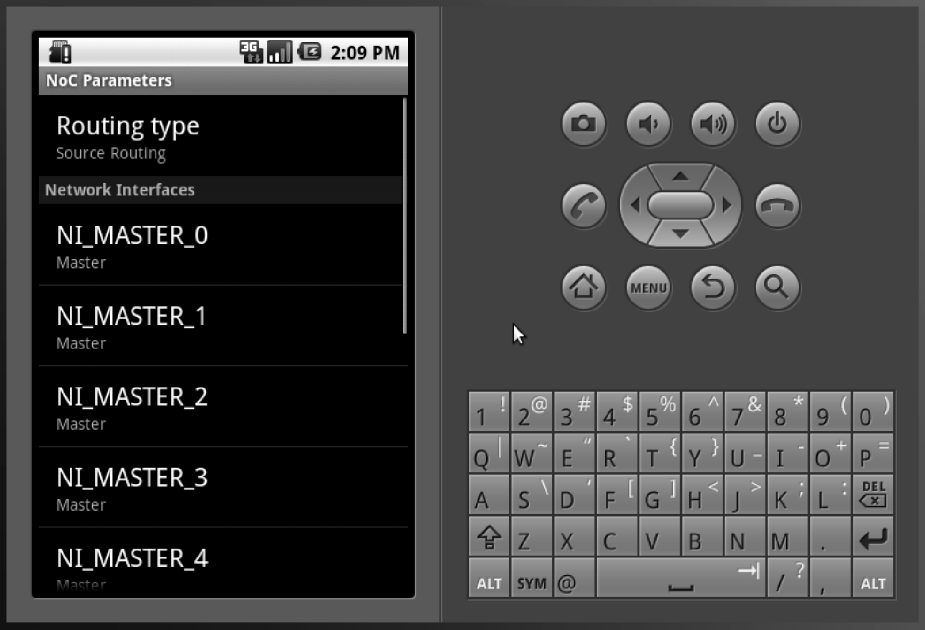
Υποστήριξη Android

- Έχουμε υλοποιήσει τον STNoC Content Provider για πρόσβαση στο NoC, π.χ.
  - content://com.st.stnoc/parameters
  - content://com.st.stnoc/nis
  - content://com.st.stnoc/nis/5
  - content://com.st.stnoc/nis/5/qos
  - content://com.st.stnoc/nis/5/routing

Υποστήριξη Android

- Η υλοποίηση χρησιμοποιεί την libstnoc μέσω του Java Native Interface (JNI).
- Έτσι κώδικας στην κορυφή του Android stack μπορεί να χειριστεί τις παραμέτρους του NoC
- Πιθανές εφαρμογές:
  - GUI για διαχείριση του NoC, debuggign σε συσκευή υπό ανάπτυξη.
  - Χρήση από το περιβάλλον Android για προσαρμογή σε γεγονότα, π.χ. τηλεφώνημα.

Υποστήριξη Android



Υποστήριξη Android

Συμπεράσματα

- Τα σύγχρονα NoC όπως το Spidergon STNoC δίνουν την δυνατότητα προσαρμογής της λειτουργίας τους κατά τον χρόνο εκτέλεσης.
- Ο χειροκίνητος προγραμματισμός αυτών των services είναι χρονοβόρος και μη portable.
- Παρέχουμε τα εργαλεία και APIs για εύκολη προσαρμογή των πόρων του NoC από υψηλού επιπέδου εφαρμογές.
  - Δεν θυσιάζουμε το portability.

Πιθανές μελλοντικές εξελίξεις

- Υλοποίηση μελλοντικών services του Spidergon που μπορεί να εμφανιστούν.
  - Παρακολούθηση δικτύου, διαγνωστικά.
- Περισσότερα constraints για QoS, π.χ. με βάση το latency ή την κατανάλωση ενέργειας.
- Αυτόματη προσαρμογή με βάση events.
- Security policies για προγραμματισμό του NoC από μη privileged εφαρμογές.