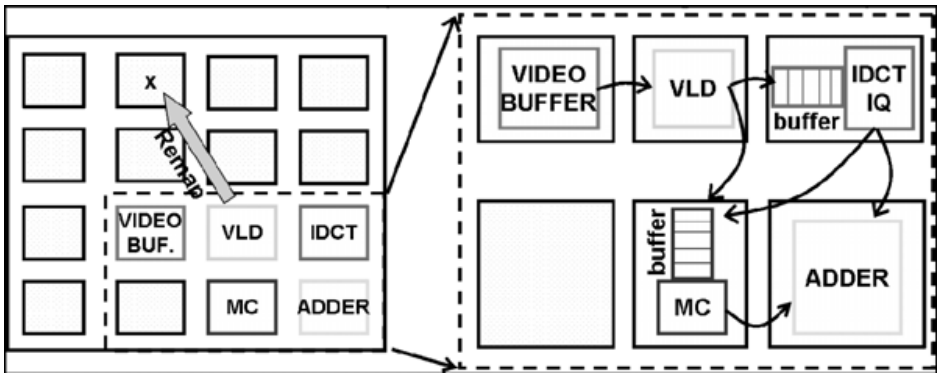
Το πρόβλημα της Αντιστοίχησης (The Mapping Problem)

Όταν είναι γνωστά επικοινωνιακα μοτίβα που χαρακτηρίζουν την εφαρμογή (communication patterns), ποιες είναι δηλαδή οι διαδικασίες επικοινωνίας που λαμβάνουν χώρα και με ποια συχνότητα, τότε η περιγραφή μπορεί να γίνει με τη χρήση ενός *κατευθηνόμενου γραφήματος* (directed graph), του οποίου οι κόμβοι αναπαριστούν στοιχεία της εφαρμογής ενώ οι συνδετικές ακμές, όπου αυτές υπάρχουν, συμβολίζουν την ύπαρξη επικοινωνίας μεταξύ των στοιχείων αυτών. Επιπλέον, οι γραμμές αυτές δύναται να παρέχουν επιπλέον πληροφορίες πέρα από τις εξαρτήσεις μεταξύ των στοιχείων της εφαρμογής. Μια τέτοια πληροφορία μπορεί να είναι το εύρος της επικοινωνίας (communication BW), δηλαδή οι επικοινωνιακές απαιτήσεις των στοιχείων. Συνεπώς το πρόβλημα στην περίπτωση αυτή είναι πως θα επέλθει μια αντιστοίχιση των κόμβων ούτως ώστε να ικανοποιούνται ορισμένες περιοριστικές συνθήκες. Επιπλέον, κατά παρόμοιο τρόπο μπορεί και η πλατφόρμα του NoC(Network-on-Chip) να περιγραφεί από ένα γράφημα, του οποίου οι κόμβοι αναπαριστούν τα στοιχεία του συστήματος που συνδέονται δικτυακά μέσω του NoC, δηλαδή υπολογιστικές μονάδες(IP tiles/cores), στοιχεία μνήμης κ.α., ενώ οι ακμές συμβολίζουν τις υπάρχουσες συνδέσεις που ορίζει η εκάστοτε τοπολογία του δικτύου. Ομοίως αυτές οι ακμές, όπως και οι ακμές στο γράφημα της εφαρμογής, συνοδεύονται από περαιτέρω πληροφορίες που μπορεί να σχετίζονται με το μέγιστο εύρος διαμεταγωγής που υποστηρίζουν, τη μέση καθυστέρηση στη μεταγωγή κ.α..

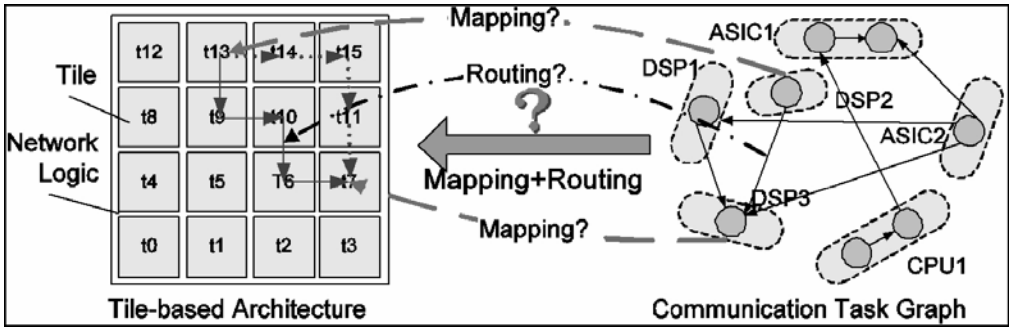
Με δεδομένες τις 2 περιγραφές, μένει μονάχα να επιτευχθεί μια καλή αντιστοίχιση που θα ικανοποιεί σε κάποιο βαθμό τους επιθυμητούς στόχους (απόδοση, κατανάλωση ισχύος κ.α.), υπακούοντας οπωσδήποτε στα εκάστοτε κριτήρια σχεδιασμού (επιφάνεια, εύρος επικοινωνίας, καθυστέρηση κ.α.). Μερικά παραδείγματα αντιστοίχησης φαίνονται στις εικόνες που ακολουθούν, όπου αποσαφηνίζονται με παραστατικό τρόπο οι περιγραφές που προηγήθηκαν, ενώ παράλληλα δίνεται και μια πιθανή αντιστοίχιση, όπως αυτή προκύπτει μέσα από έναν αλγόριθμο αντιστοίχισης.



Εικόνα 1: Αντιστοίχιση ενός αποκωδικοποιητή MPEG-2 σε ένα 4x4 NoC με στόχο την τοποθέτηση των στοιχείων που επικοινωνούν μεταξύ τους σε γειτονικές θέσεις[1].

Στο παράδειγμα που φαίνεται στην Εικόνα 1, κάτι που αξίζει να σημειωθεί είναι το γεγονός ότι η εφαρμογή την οποία καλούμαστε να αντιστοιχίσουμε πάνω στο NoC δεν καλύπτει πλήρως τη διαθέσιμη επιφάνεια. Είναι εμφανές ότι στην πλειοψηφία τους οι θέσεις της πλατφόρμας παραμένουν κενές, κάτι που υποδηλώνει τη δυνατότητα πολλαπλής αντιστοίχισης επιπλέον εφαρμογών στο ίδιο NoC, είτε αντιγράφων της ίδιας εφαρμογής για επεξεργασία περισσότερων δεδομένων είτε πολλών διαφορετικών εφαρμογών που στο σύνολό τους να εξυπηρετούν τη λειτουργία του chip.

Ένα άλλο παραστατικό παράδειγμα είναι αυτό της Εικόνας 2, όπου γίνεται σαφές ότι η αντιστοίχιση σίγουρα δεν είναι κάτι που θα έπρεπε να πραγματοποιηθεί εντελώς στην τύχη. Αντιθέτως, μεσολαβούν κάποιοι υπολογισμοί και επικρατεί μια λογική με βάση την οποία ανατίθεται κάποιο στοιχείο της εφαρμογής σε ένα κόμβο της πλατφόρμας. Οι υπολογισμοί ενδεχομένως να λαμβάνουν υπόψη τις διαφορετικές διαδρομές εντός του NoC που μπορεί να ακολουθήσουν τα δεδομένα προκειμένου να φτάσουν στον προορισμό τους, κάτι που εξαρτάται από την τοπολογία του δικτύου.



Εικόνα 2: Απεικόνιση της περιγραφής της εφαρμογής και της πλατφόρμας του NoC. Διαδικασία mapping λαμβάνοντας υπόψη διαφορετικές επιλογές routing.[1]

Προτού περάσουμε στη μαθηματική περιγραφή και τους ορισμούς των μοντέλων των γραφημάτων της εφαρμογής και της πλατφόρμας του NoC, είναι σημαντικό να ανακεφαλαιώσουμε δίνοντας έμφαση στην αναγκαιότητα μιας καλής αντιστοίχισης για το εκάστοτε SoC. Το κύριο πρόβλημα μεταφράζεται ως εξής: πως θα τοποθετήσω στη διαθέσιμη πλατφόρμα τα στοιχεία της εφαρμογής που μελετώ. Μια εύκολη λύση θα ήταν η εντελώς τυχαία τοποθέτηση, κάτι που όμως αποδεικνύεται παντελώς άστοχο. Είναι απολύτως λογικό κανείς να μην επιθυμούσε να αντιστοιχίσει σε αντιδιαμετρικούς κόμβους της πλατφόρμας δύο στοιχεία της εφαρμογής που, σύμφωνα με το γράφο περιγραφής, επικοινωνούν μεταξύ τους. Είναι απολύτως λογικό και φρόνιμο αυτά τα δύο στοιχεία να βρίσκονται σε κοντινούς κόμβους, κάτι που θα επιβαρύνει το δίκτυο στο ελάχιστο δυνατό και θα επιφέρει τη χαμηλότερη δυνατή κατανάλωση ισχύος συνολικά, μιας και θα οδηγείται ηλεκτρικά μονάχα ένα κανάλι επικοινωνίας, αλλά και θα αποφέρει τη μικρότερη δυνατή καθυστέρηση στην επικοινωνία.

Μάλιστα, ορισμένους από αυτούς θα έχουμε τη δυνατότητα να μελετήσουμε στη συνέχεια της παρούσης εργασίας. Η εκάστοτε διαδικασία σύμφωνα με κάποια λογική, αποφέρει μια ικανοποιητική αντιστοίχιση, ενδεχομένως διαφορετική από τις υπόλοιπες διαδικασίες. Σίγουρα όμως αυτές οι λύσεις δεν αποτελούν βέλτιστες λύσεις, παρόλη την προσπάθεια που γίνεται στο να δώσουν όσο πιο κοντινά αποτελέσματα είναι δυνατόν. Συνήθως, αφότου επέλθει μια αρχική αντιστοίχιση, στη συνέχεια εφαρμόζονται διάφορες τεχνικές προκειμένου να ερευνηθεί εάν υπάρχει μια καλύτερη λύση κοντινά στην δεδομένη κατάσταση. Μια τέτοια διαδικασία, είναι μια επαναληπτική διαδικασία αμοιβαίας εναλλαγής συγκεκριμένων κόμβων. Σε κάθε βήμα, δύο από τους κόμβους που έχουν αντιστοιχηθεί αλλάζουν θέσεις στο χώρο μεταξύ τους και δίνουν αφορμή για πραγματοποίηση εκ νέου ορισμένων υπολογισμών. Στην περίπτωση που η αλλαγή αυτή δώσει καλύτερα από τα μέχρι πρότινος τελικά αποτελέσματα, διατηρείται, και η διεργασία συνεχίζει ως έχει, ενώ σε αντίθετη περίπτωση η ανταλλαγή αναιρείται και πραγματοποιείται ακολούθως για κάποιο διαφορετικό ζευγάρι κόμβων. Όπως είναι λογικό, η διαδικασία αυτή σύντομα φτάνει στο τέλος της και όπως θα παρατηρήσουμε, στην πλειοψηφία των περιπτώσεων αποφέρει καλύτερο αποτέλεσμα. Μια άλλη τεχνική που βελτιώνει το αποτέλεσμα και οδηγεί σε λύσεις πολύ κοντά στη βέλτιστη περίπτωση, είναι η τεχνική της εξομοιωμένης ανόπτησης (Simulated Annealing) [2] [3], που όμως δε θα μας απασχολήσει διότι απαιτεί χρονοβόρους υπολογισμούς αλλά και ισχυρές μαθηματικές εξισώσεις που να καθοδηγούν την επαναληπτική διαδικασία.

Μαθηματική μοντελοποίηση

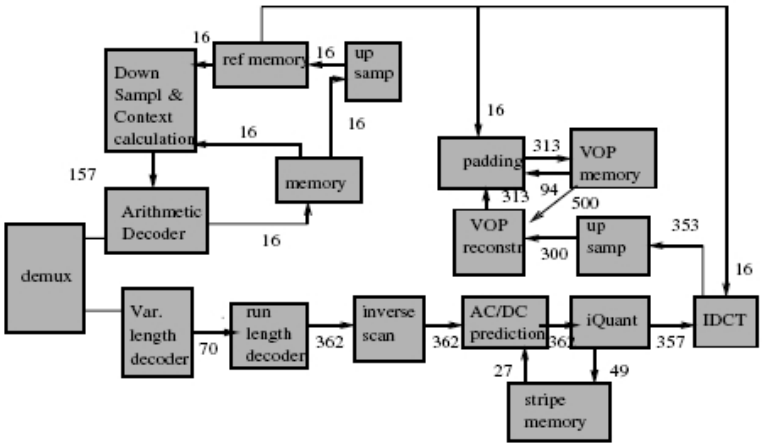
Η επικοινωνία μεταξύ των πυρήνων (Cores) ενός SoC αναπαρίσταται από ένα γράφο που θα ονομάσουμε core graph:

Ορισμός 1: Ο γράφος core graph είναι ένας κατευθυνόμενος (Directed) γράφος G(V,E) που καθένας από τους κόμβους του $u_i \in V$ αναπαριστά ένα πυρήνα και κάθε κατευθυνόμενη ακμή του (u_i,u_j), ορισμένη ως $e_{ij} \in E$, υποδηλώνει την υπάρχουσα επικοινωνία μεταξύ των κόμβων u_i και u_j. Το βάρος (Weight) της ακμής e_{ij}, δηλωμένο σαν $comm_{ij}$, υποδηλώνει το εύρος της επικοινωνίας (BW) από τον κόμβο u_i στον u_j.

Η συνδετικότητα (Connectivity) και το υποστηριζόμενο εύρος διαμεταγωγής των καναλιών επικοινωνίας του NoC αναπαρίστανται από επίσης ένα γράφο, τον οποίο ονομάζουμε NoC topology graph:

Ορισμός 2: Ο γράφος NoC topology graph είναι επίσης ένας κατευθυνόμενος γράφος P(U,F) όπου καθένας από τους κόμβους του $u_i \in U$ αναπαριστά ένα

κόμβο της τοπολογίας του δικτύου και η κατευθυνόμενη ακμή (u_i,u_j), δηλωμένη σαν f_{ij}, $\in F$, αναπαριστά την υπάρχουσα επικοινωνία με τη δεδομένη διεύθυνση των δεδομένων μεταξύ των κόμβων u_i και u_j. Το βάρος της ακμής f_{ij} , ορισμένο σαν bw_{ij}, υποδηλώνει το διαθέσιμο εύρος επικοινωνίας κατά μήκος της συγκεκριμένης ακμής f_{ij}.



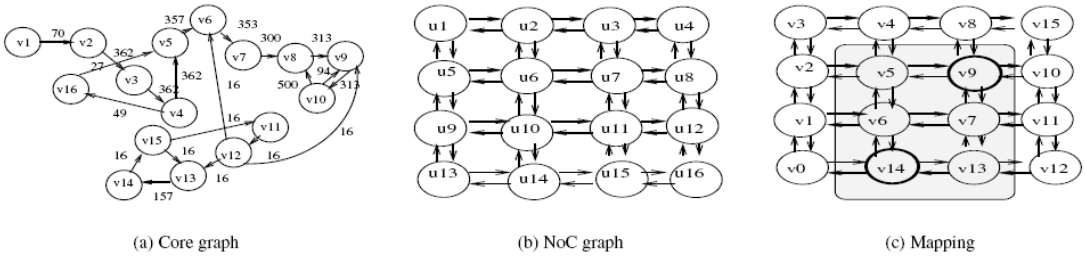
Εικόνα 3: Block διάγραμμα μιας εφαρμογής αποκωδικοποίησης βίντεο VOPD (Video Object Plane Decoder), με εμφανείς τις επικοινωνιακές απαιτήσεις των κόμβων σε MB/s.

Προτού συνεχίσουμε τη μαθηματική περιγραφή και μοντελοποίηση, παραθέτουμε μερικές εικόνες προκειμένου να αποσαφηνίσουμε τη διαδικασία του προβλήματος που μελετάμε. Η Εικόνα 3 παρουσιάζει το «block διάγραμμα» ενός αποκωδικοποιητή βίντεο, στο οποίο διακρίνονται τα εξαρτήματα που απαρτίζουν το σύστημα και οι μεταξύ τους επικοινωνιακές ανάγκες σε μονάδες MB/s [5], κάτι που επιπλέον τονίζει τον όγκο της πληροφορίας που χειρίζονται ορισμένες απαιτητικές εφαρμογές όπως η παρακάτω.

Έχοντας το παραπάνω διάγραμμα, το επόμενο βήμα είναι να εξάγει κανείς το γράφο της εφαρμογής, όπως περιγράψαμε προηγουμένως. Έτσι λοιπόν, ο γράφος core graph εμφανίζεται στην Εικόνα 4a και ο γράφος της πλατφόρμας του NoC, ενός κανονικού δικτυώματος 4x4 (mesh) παρουσιάζεται στην Εικόνα 4b. Η αντιστοίχιση του core graph G(V,E) στον platform graph P(U,F) πραγματοποιείται από την 1 προς 1 συνάρτηση αντιστοίχισης map:
map:

$$V \rightarrow U \text{ , } map(u_i) = u_j, \forall u_i \in V, \exists u_j \in U$$

Η αντιστοίχηση ορίζεται φυσικά μόνο εφόσον υπάρχουν αρκετές θέσης στην πλατφόρμα ούτως ώστε να επαρκούν για τα στοιχεία της εφαρμογής, δηλαδή να ισχύει: |V |≤|U |.
Ένα παράδειγμα αντιστοίχισης φαίνεται στην Εικόνα 4c.



(a) Core graph (b) NoC graph (c) Mapping

Εικόνα 4. Αντιστοίχιση του γράφου της εφαρμογής στο γράφο της πλατφόρμας του NoC.

Η επικοινωνία μεταξύ κάθε ζευγαριού πυρήνων (π.χ. κάθε ακμή $e_{ij} \in E$) μεταχειρίζεται σαν μια μονής διεύθυνσης ροή δεδομένων (Flow) που συμβολίζεται σαν d_k όπου k = 1,2,...|E|. Η τιμή της παραμέτρου d_k αναπαριστά το εύρος της επικοινωνίας κατά μήκος της ακμής και δηλώνεται σαν $vl(d_k)$. Το σύνολο των ροών D που χαρακτηρίζουν την επικοινωνία του συστήματος ορίζεται ακολούθως:

$$D = \left\{ \begin{array}{l} d^k : vl(d^k) = comm_{ij}, k = 1,2,...|E|, \forall e_{ij} \in E \\ source(d^k) = map(u_i), dest(d^k) = map(u_j) \end{array} \right\}$$

Πρόκειται δηλαδή για το σύνολο των ροών δεδομένων, για καθεμιά από τις οποίες ισχύει ότι η αριθμητική της τιμή ισούται με το ίδιο το βάρος της ακμής που συνδέει τους εκάστοτε δύο κόμβους, για τους οποίους ισχύει ότι ο κόμβος-αποστολέας και ο κόμβος- παραλήπτης της συγκεκριμένης ροής πάνω στην πλατφόρμα αποκαλύπτονται από τη συνάρτηση αντιστοίχησης.

Η ανισότητα που περιγράφει τον περιορισμό για το διαθέσιμο εύρος επικοινωνίας (Communication BW) που παρέχει κάθε κανάλι της πλατφόρμας του NoC είναι η εξής:

$$\sum_{k=1}^{|E|} x_{ij}^k \le bw_{ij}, \forall i, j \in 1,2,...,|U|$$

Υποδηλώνει δηλαδή ότι, σε κάθε κανάλι το άθροισμα των δεδομένων που κάθε στιγμή το διασχίζουν, ανεξάρτητα από ποια ροή προέρχονται, δεν πρέπει σε καμιά περίπτωση να υπερβαίνει το ανώτατο επιτρεπόμενο εύρος που υποστηρίζεται.

Για ένα μηχανισμό δρομολόγησης που εκμεταλλεύεται την ελάχιστη διαδρομή μεταξύ αποστολέα και παραλήπτη (Single minimum – path Routing), ο παράγοντας x_{kij} ορίζεται από τη σχέση, όπου ο όρος path(a,b) αναπαριστά το σύνολο των καναλιών που δομούν ακολουθιακά τη διαδρομή που θα ακολουθήσουν τα δεδομένα εντός του δικτύου έως ότου φτάσουν στον

προορισμό τους.

$$x_{ij}^{k} = \begin{Bmatrix} vl(d^{k}), & f_{ij} \in Path(source(d^{k}), dest(d^{k})) \\ 0, & otherwise \end{Bmatrix}$$

Ανακεφαλαιώνοντας, συνοψίζουμε ότι το πρόβλημα της αντιστοίχισης έγκειται στην αποδοτική τοποθέτηση των στοιχείων της εφαρμογής στις διαθέσιμες θέσεις του πλέγματος της εφαρμογής, όπου υποβόσκουσα επικοινωνιακή υποδομή είναι αυτή ενός NoC.

Αλγόριθμοι Αντιστοίχησης (Mapping Algorithms)

Αμφότερες οι λύσεις που θα μας απασχολήσουν υλοποιούν αλγορίθμους που λαμβάνουν υπόψη τους ένα δεδομένο περιορισμό για το διαθέσιμο εύρος που παρέχεται από κάθε κανάλι επικοινωνίας της πλατφόρμας του NoC. Πρόκειται για αλγορίθμους οι οποίοι χρησιμοποιούν ένα μηχανισμό δρομολόγησης με βάση την ελάχιστη διαδρομή μεταξύ των κόμβων σε μια αρχιτεκτονική απλού, κανονικού δικτυώματος (mesh). Η μεθοδολογία θεωρείται ευρετική (heuristic), διότι όπως εξηγήσαμε πρόκειται για ένα πρόβλημα για το οποίο δύσκολα κανείς φτάνει στη βέλτιστη λύση. Χωρίζεται σε τρείς φάσεις: *τη φάση αρχικοποίησης* (Initializing) που αποφέρει μια αρχική αντιστοίχιση των γράφων, ακολουθούμενη από τη δεύτερη φάση όπου πραγματοποιούνται οι υπολογισμοί για τον *εντοπισμό των ελάχιστων διαδρομών* που πρέπει να ακολουθήσουν οι ροές των δεδομένων, σύμφωνα με τον αλγόριθμο εύρεσης της βέλτιστης διαδρομής του Dijkstra [4]. Τέλος, η διεργασία κλείνει με μια *επαναληπτική διαδικασία* όπου η αρχική λύση βελτιώνεται (όταν και όσο αυτό είναι δυνατό να γίνει) έπειτα από συνεχείς αναζητήσεις και αμοιβαίες ανταλλαγές κόμβων που βρίσκονται στην αρχικά προτεινόμενη λύση που προκύπτει έπειτα από την αρχικοποίηση.

Στη συνάρτηση Initialize() που παρουσιάζεται παρακάτω, αρχικά ο κόμβος που εμφανίζει το μέγιστο όγκο στις επικοινωνιακές απαιτήσεις, που δεν είναι άλλος από το άθροισμα όλων των βαρών των ακμών που ξεκινούν ή καταλήγουν στον κόμβο αυτό, τοποθετείται σε κάποιο από τους κεντρικούς κόμβους του δικτυώματος της πλατφόρμας, όπου και έχουμε το μεγαλύτερο αριθμό ελεύθερων γειτονικών θέσεων. Φυσικά, δεδομένου ότι αρχικά το NoC είναι άδειο, οποιαδήποτε κεντρική θέση (που να συνορεύει με τέσσερις επιπλέον κόμβους της πλατφόρμας) είναι ικανοποιητική. Για παράδειγμα, σε ένα απλό κανονικό πλέγμα 5x5 κόμβων, επιλέγεται η θέση με συντεταγμένες (x,y)=(2,2), δηλαδή η κεντρική θέση. Σε περίπτωση άρτιου αριθμού πλευρικών διαστάσεων, όπως 4x4, επιλέγεται μια εκ των 4ων κεντρικών θέσεων.

Στη συνέχεια, για κάθε υποψήφιο κόμβο της εφαρμογής που έπεται να βρεθεί η αντίστοιχη θέση του στο NoC, πραγματοποιούνται κάποιοι υπολογισμοί και τελικώς επιλέγεται αυτός που επικοινωνεί περισσότερο με τους κόμβους που ήδη έχουν αντιστοιχηθεί στην πλατφόρμα. Έπειτα, αυτός ο κόμβος

τοποθετείται στην πρώτη ελεύθερη από τις υποψήφιες θέσεις του δικτυώματος, για τις οποίες ισχύει ότι το συνολικό κόστος επικοινωνίας του NoC (εννοείται για τους κόμβους που έχουν ήδη τοποθετηθεί) γίνεται ελάχιστο. Φυσικά η θέση αυτή προκύπτει από τον έλεγχο κάθε πιθανής θέσης στην πλατφόρμα.

Για παράδειγμα, στο σχήμα της Εικόνα 4c, φανταστείτε την περίπτωση όπου ο πρώτος κόμβος που τοποθετήθηκε είναι ο v9 και έπεται να τοποθετηθεί ο v5. Οι 4 υποψήφιες θέσεις είναι οι τέσσερις γειτονικές του v9 προς πάσα κατεύθυνση. Η διαδικασία αυτή επαναλαμβάνεται μέχρις ότου όλοι οι κόμβοι της εφαρμογής έχουν αντιστοιχηθεί στην πλατφόρμα. Η συνάρτηση ShortestPath() που επίσης παρατίθεται παρακάτω, υλοποιεί την εύρεση της ελάχιστης διαδρομής. Αρχικά όλες οι ροές επικοινωνίας ταξινομούνται σε φθίνουσα σειρά ανάλογα με το βάρος της επικοινωνίας. Για κάθε μια από αυτές δημιουργείται ένας γράφος με διαστάσεις μικρότερες ή το πολύ ίσες με τις διαστάσεις του αρχικού πλέγματος, όπου οι κόμβοι του αποστολέα και του παραλήπτη αποτελούν αντιδιαμετρικά άκρα, δεδομένου ότι η μικρότερη διαδρομή μεταξύ τους έγκειται οπωσδήποτε σε αυτό το σχηματισμένο βοηθητικό γράφο. Η σκιασμένη περιοχή της Εικόνα 4c αποτελεί ένα τέτοιο χαρακτηριστικό γράφο για τη ροή μεταξύ των v14 και v9.

Σημαντικό ρόλο εδώ παίζει φυσικά η φορά αρίθμησης των κόμβων και η φορά προσπέλασης τους από τον αλγόριθμο, διότι λίγο πριν μιλήσαμε για την πρώτη ελεύθερη θέση κατά την πλοήγηση στο πλέγμα.

Όσον αφορά την εύρεση της μικρότερης διαδρομής, το ρόλο αυτό αναλαμβάνει όπως προαναφέραμε ο αλγόριθμος εύρεσης βέλτιστης διαδρομής του Dijkstra (Dijkstra shortest path) [4] ο οποίος εφαρμόζεται πάνω στον προσωρινό βοηθητικό γράφο που υπολογίζεται για κάθε ροή δεδομένων.

Έπειτα, τα επιλεγμένα κανάλια της πλατφόρμας που απαρτίζουν τη διαδρομή μέχρι τον προορισμό επιφορτίζονται με το βάρος της επικοινωνίας, διατηρώντας την τιμή αυτή σε μια μεταβλητή η οποία αυξάνει όποτε ένα κανάλι χρησιμοποιείται εκ νέου από μια διαφορετική ροή, διαδικασία που επαναλαμβάνεται για όλες τις υπάρχουσες ροές δεδομένων. Στο τέλος, εφόσον έχουν δρομολογηθεί όλα τα πακέτα πληροφορίας, εξετάζεται αν το προσωρινό φορτίο σε κάποιο κανάλι ξεπερνά το διαθέσιμο εύρος που παρέχει, με αποτέλεσμα να παραβιάζεται ο κανόνας του περιορισμένου εύρους διαμεταγωγής (BW Constraint) που τέθηκε εξ αρχής. Τότε υπολογίζεται το συνολικό κόστος επικοινωνίας για την πλατφόρμα, σύμφωνα με την παρακάτω σχέση:

$$commCost = \sum_{k=1}^{|E|} vl(d^k) \cdot dist(source(d^k), dest(d^k))$$

Όπου dist(a,b) ορίζουμε τον ελάχιστο απαιτούμενο αριθμό των hops, δηλαδή των κόμβων που μεσολαβούν στην επικοινωνία, μεταξύ των κόμβων a και b. Η σχέση αυτή επομένως αποδίδει το συνολικό άθροισμα των γινομένων του βάρους της επικοινωνίας κάθε ροής επί τον αριθμό των καναλιών που διατρέχει η πληροφορία πάνω στην πλατφόρμα.

Η συνάρτηση MappingWithSinglePath() στοχεύει στο να βελτιώσει την αρχική αντιστοίχιση ψάχνοντας και κάνοντας συνεχείς εναλλαγές στις θέσεις ζευγών κόμβων στο δικτύωμα του NoC, καλώντας τη συνάρτηση ShortestPath() O(U 2) φορές. Σημειωτέον ότι η χειρότερη περίπτωση υπολογιστικής πολυπλοκότητας για τον αλγόριθμο είναι της τάξηςO(|U|3 Elog|F|).

Αναλύοντας τον παρακάτω αλγόριθμο στον Πίνακα 1, φτάνει κανείς στο συμπέρασμα ότι δίνεται προτεραιότητα στους κόμβους που εμφανίζουν τις μεγαλύτερες επικοινωνιακές απαιτήσεις. Από αυτούς κάθε φορά, επιλέγεται ο επόμενος της λίστας που έχει τις αμέσως μικρότερες σε όγκο απαιτήσεις και τοποθετείται στο πλέγμα στην πρώτη ελεύθερη θέση που συναντάται από όσες ελαχιστοποιούν το συνολικό κόστος επικοινωνίας.

initialize(*G(V,E)*, *P(U,F)*){
initialize *Placed(W,H)* to *Ø*;
assign the vertex with max communication requirements in *G(V,E)* to *maxs*;
assign the vertex with maximum neighbors in *P(U,F)* to *maxt*;
map(*maxs*) = *maxt*;
remove *maxt* from *P*, *maxs* from *G* and add *maxt* to *Placed(W,H)*;
While(|*V* | > 0){
assign the vertex in G with maximum comm with $\forall\ w_i \in W$, to *nexts*;
for $\forall\ u_j \in P(U,F)$ and $w_i \in Placed(W,H)$ {
$commCost(u_j) += comm_{nexts,map-1(wi)} \times [xdist(w_i,w_j) + ydist(w_i,w_j)]$;
}
assign u_j with minimum *cost* to *nextt*;
map(*nexts*) = *nextt*;
remove *nextt* from *P*, *nexts* from *G* and add *nexts* to *W*;
}
return(*map*, *Placed(W,H)*);
}

shortestPath(*Placed(W,H)*){
initialize edge weights of *Placed* with total comm BW available and *current BW* to 0;
sort commodities in *D* with decreasing *comm costs*;
for each $d^k \in D$ do{
make quadrant graph $Q(d^k)$ with $source(d^k)$ and $dest(d^k)$ as end vertices;
$Path(source(d^k), dest(d^k)) = minpath(Q(d^k))$;
increase edge current weights for edges in *Path* by $vl(d^k)$;
}
if BW constraints are satisfied
find the *comm cost* and store it in *cost*;
else
assign *maxvalue* to *cost*;
return(*cost*);
}

mappingWithSinglePath(*G(V, E), P(U,F)*){
S(A,B) = makeUndirected(*G(V,E)*);
initialize(*S(A,B), P(U,F)*);
bestcommcost = *shortestpath*(*Placed(W,H)*);
assign *Placed* to *Bestmapping*;
for i = 1 to |*U*| do{
for j = i+1 to |*U*| do{
assign *Placed(W,H)* to *Ptemp(tW, tH)*
swapping vertices w_i and w_j ;
commcost = *shortestpath*(*Ptemp(tW, tH)*);
if (*commcost* < *bestcommcost*){
assign *Ptemp* to *Bestmapping* and *commcost* to *bestcommcost*;
}
}
assign *Bestmapping* to *Placed*;
}
return(*bestcommcost*, *Bestmapping*);
}

Πίνακας 1: Οι τρεις συναρτήσεις – φάσεις του αρχικού αλγορίθμου αντιστοίχισης.

Βιβλιογραφία

[1] R. Marculescu, U. Y. Ogras, and N. H. Zamora. *Computation and communication refinement for multiprocessor SoC design: A system-level perspective*. In Proc. of DAC, pages 564–592. ACM Press, 2004.

[2] Z. Lu, L. Xia, and A. Jantsch. *Cluster-based simulated annealing for mapping* 9-3*cores onto 2D mesh networks on chip*. Pages 1- 6, April 2008.

[3] S. Kirkpatrick; C. D. Gelatt; M. P. Vecchi. *Optimization by Simulated Annealing*, Science Magazine, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680.

[4] Dijkstra's Shortest Path Algorithm: en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[5] S. Murali and G. D. Micheli. *Bandwidth-constrained mapping of cores onto NoC architectures*. In Proc. of DATE, page 20896. IEEE Computer Society, 2004.