ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΜΥ

ΠΑΡΑΛΛΗΛΟΙ ΚΑΙ ΔΙΚΤΥΑΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ

ΣΕΤ ΑΣΚΗΣΕΩΝ 2ο

ΑΛΕΞΑΝΔΡΙΔΗΣ ΓΙΩΡΓΟΣ

ΓΡΕΤΣΙΣΤΑΣ ΒΑΣΙΛΗΣ

ΜΑΡΕΛΑΣ ΓΙΩΡΓΟΣ

ΑΣΚΗΣΗ 1: Serial FFT Algorithm

Πριν προχωρήσουμε στην σειριακή υλοποίηση και μετέπειτα στην παράλληλη υλοποιήση του αλγορίθμου, πρέπει να επεξηγηθούν ορισμένες έννοιες της θεωρίας των FFT αλγορίθμων. Έστω ότι έχουμε είσοδο X = < X[0], X[1], ..., X[n-1] > μεγέθους n. Ο DFT της X είναι μια ακολουθία Y = < Y[0], Y[1], ..., Y[n-1] > (X[i], Y[i] μιγαδικοί), όπου

$$Y[i] = \sum_{k=0}^{n-1} X[k]\omega^{ki}, 0 \le \iota < n \ , \qquad \omega = e^{2\pi\sqrt{-1}/n}$$

(Οι δυνάμεις του ω που χρησιμοποιούνται στον FFT ονομάζονται *twiddle factors*.)

Υποθέτουμε πως n είναι δύναμη του 2. Ένας n-point FFT μπορεί να χωριστεί σε δύο (n/2) DFTs.Υποακολουθίες μονών και ζυγών indexes:
*even →2k *odd→2k+1 k=0,1,…,n/2-1

$$Y[i] = \sum_{k=0}^{(n/2)-1} X[2k]\omega^{2ki} + \sum_{k=0}^{(n/2)-1} X[2k+1]\omega^{(2k+1)i}$$

$$= \sum_{k=0}^{(n/2)-1} X[2k]e^{2(2\pi\sqrt{-1}/n)ki} + \sum_{k=0}^{(n/2)-1} X[2k+1]\omega^{i}e^{2(2\pi\sqrt{-1}/n)ki}$$

$$= \sum_{k=0}^{(n/2)-1} X[2k]e^{2\pi\sqrt{-1}ki/(n/2)} + \omega^{i}\sum_{k=0}^{(n/2)-1} X[2k+1]e^{2\pi\sqrt{-1}ki/(n/2)},$$

και θέτοντας $\varpi = e^{2\pi\sqrt{-1}/(n/2)} = \omega^2$ έχουμε :

$$Y[i] = \sum_{k=0}^{(n/2)-1} X[2k]\varpi^{ki} + \omega^{i}\sum_{k=0}^{(n/2)-1} X[2k+1]\varpi^{ki}$$

(Για τον υπολογισμό των ω χρησιμοποιούμε τις ιδιότητες της συμμετρικότητας ω^(κ+ν/2) = -ω^κ και περιοδικότητας ω^(κ+ν) = ω^κ)

One dimensional implementation, radix-2 FFT

```
1.    procedure ITERATIVE_FFT( X, Y, n )
2.    begin
3.          r := logn;
4.          for i:= 0 to n-1 do R[i]:=X[i];
5.          for m:= r - 1 to 0 do   /* Εξωτερικός βρόχος */
6.          begin
      /* η παρακάτω αντιγραφή μπορεί να αποφευχθεί με
       * κατάλληλη χρήση των ενδιάμεσων πινάκων */
7.                for i:= 0 to n-1 do S[i]:=R[i];
8.                for i:= 0 to n-1 do /*Εσωτερικός βρόχος */
9.                begin
      /*(b_{r-1} b_{r-2}… b_0) είναι η δυαδική αναπαράσταση του i */
10.                     j:= (b_{r-1}… b_{m+1}0 b_{m-1} … b_0);
11.                     k:=(b_{r-1}… b_{m+1}1 b_{m-1} … b_0);
12.                     R[i]:=S[j] + S[k] x ω^(b_m b_{m-1}…b_0 0…0)
13.               endfor;          /* Εσωτερικός βρόχος */
14.         endfor;          /* Εξωτερικός βρόχος */
15.         for i:= 0 to n-1 do Y[i]:=R[i];
16.   end ITERATIVE_FFT
```

Σχ.1)Στη γραμμή 12 η τιμή του R[i] ενημερώνεται με τη χρήση των S[j], S[k].Οι δείκτες j, k προκύπτουν από το i. Έστω $n = 2^r$,με $0 \le i < n$, η δυαδική αναπαράσταση του i αποτελείται από r bits. Έστω ότι $(b_{r-1}b_{r-2}\ldots b_0)$ είναι η δυαδική αναπαράσταση του i. Τότε στην επανάληψη m του εξωτερικού βρόχου ($0 \le m < r$), παίρνουμε τον δείκτη j αν θέσουμε το m-οστό ψηφίο του i (δηλαδή το b_m) μηδέν. Αντίστοιχα παίρνουμε τον δείκτη k αν θέσουμε το b_m ένα. Δηλαδή οι δυαδικές αναπαραστάσεις των j, k διαφέρουν μόνο στο ψηφίο b_m ($k - j = 2^m$).

Σχ.2)Η δύναμη του ω ουσιαστικά προκύπτει ως εξής:

i. Γράφουμε τον δείκτη i σε δυαδική μορφή με $r=\log_2 n$ bitsà$(b_{r-1} b_{r-2}\ldots b_0)$
ii. Κάνουμε δεξιά ολίσθηση κατά m bits (δες το m στον αλγόριθμο) γεμίζοντας με μηδενικά τα νέα bits.
iii. Αντιστρέφουμε την ακολουθία των bits.

Σχ.3) Στην έξοδο δεν έχουμε τη μορφή Y = < Y[0], Y[1], ..., Y[n-1] >, αλλά μία αναδιαταγμένη σειρά των δεδομένων.Γίνεται με την αντιστροφή των bit (bit reversal process). Για να εχουμε στην έξοδο τη σωστή διάταξη,αν τα στοιχεία της εξόδου είναι Y[i], απλά γράφουμε το i σε δυαδική μορφή και στη συνέχεια αντιστρέφουμε τα bit της δυαδικής παράστασής του. Τέλος ανταλλάσσουμε το Y[i] με το Y[k] όπου k είναι ο αριθμός του δυαδικά αντεστραμένου i.

0 : Y[0000] -> Y[0000]
1:Y[0001] -> Y[1000]
2: Y[0010] -> Y[0100]
3: Y[0011] -> Y[1100]
4 : Y[0100] -> Y[0010]
κ.ο.κ

ΚΟΣΤΟΣ ΣΕΙΡΙΑΚΟΥ ΑΛΓΟΡΙΘΜΟΥ

Για 2^κ στοιχεία,υπολογίζουμε πόσες πράξεις εκτελούνται σε κάθε υποδιπλασιασμό των στοιχείων

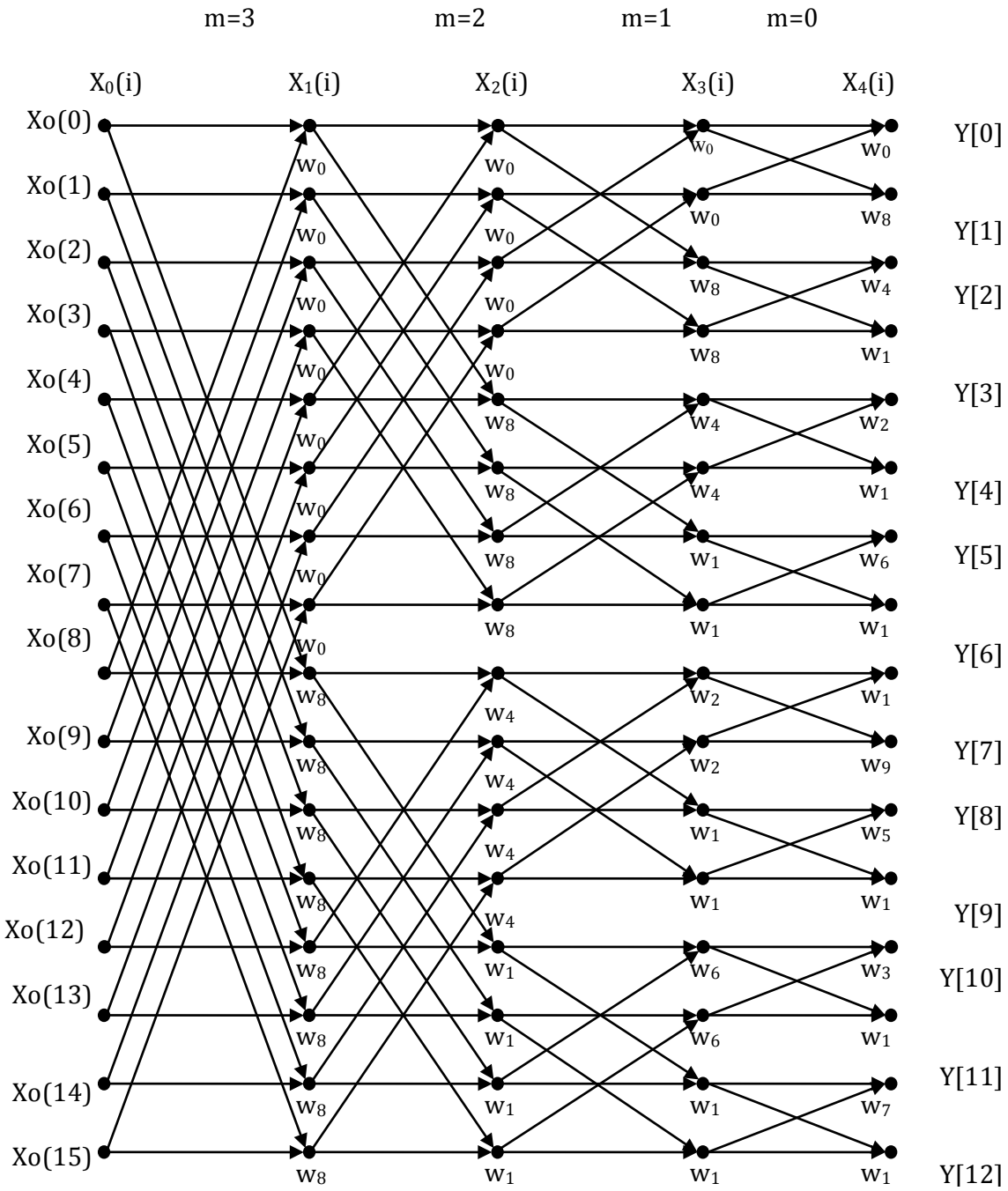
1: N/2 -> 2(N/2)^2 + N = N^2/2 + N

2: N/4 -> 2(2(N/2)^2 + N/2) +N = N^2/4 + 2N

3: N/8 -> 2[2(2(N^2/8)+N/4)+N/2]+N = N^2/8 +3N

κ : N/2^κ = 1 -> N^2/2^κ + κN = N^2/N +Nlog2N

Άρα **O(Nlog2N)** είναι το συνολικό κόστος.

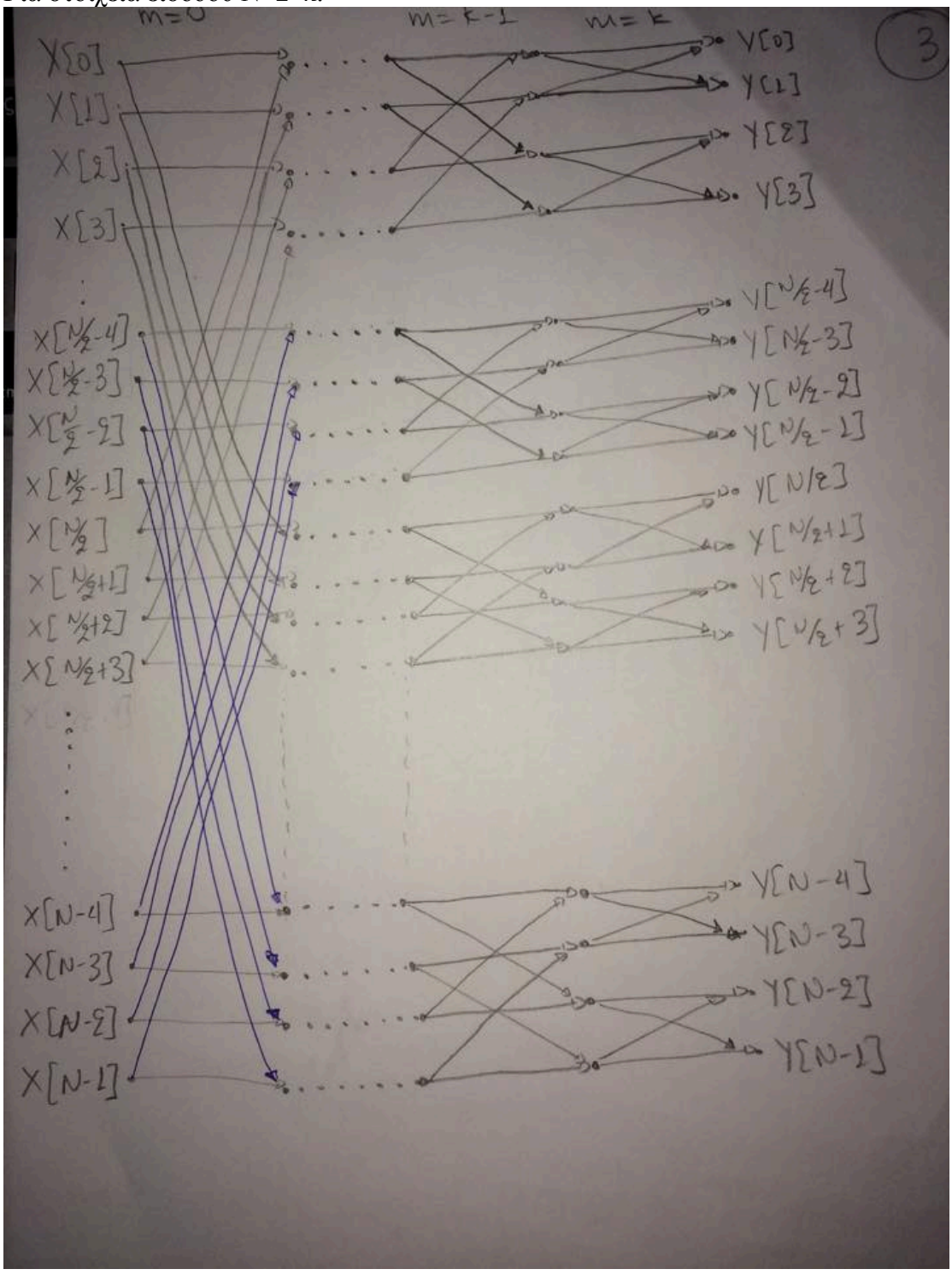


(figure 1)

*Στην έξοδο το Y[ ] έχει 16 δεδομένα εξόδου.(0-15)

Σχ.4) Κάθε κόμβος συμβολίζει ένα δεδομένο. Ένας πίνακας δεδομένων αποτελεί μία στήλη κόμβων. Κάθε κόμβος έχει δύο εισόδους, τις δύο ακμές όπου καταλήγουν σε αυτόν. Μία ακμή στον γράφο μεταφέρει το δεδομένο που βρίσκεται στο ένα άκρο της, πολλαπλασιασμένο με τον παράγοντα w^p , που βρίσκεται στο άλλο άκρο της. Όπου δεν υπάρχει παράγοντας w^p συνεπάγεται ότι $w^p = 1$. Οι δύο είσοδοι σε έναν κόμβο αθροίζονται.

Για στοιχεία εισόδου N=2^κ:



(figure 2)

ΑΣΚΗΣΗ 2: PARALLEL FFT ALGORITHM

Binary exchange. Το υπολογιστικό κομμάτι του FFT είναι ίδιο με αυτό του σειριακού που περιγράψαμε πιο πάνω.Η διαφορά είναι στην παράλληλη φύση του αλγορίθμου.Έχουμε επικοινωνία μεταξύ των διεργασιών. Στον αλγόριθμο έχουμε 2 φάσεις,2 κομμάτια κώδικα. Το ένα αφορά το τμήμα που έχουμε επικοινωνία μεταξύ των διεργασιών και το άλλο αφορά το κομμάτι που οι διεργασίες λειτουργούν τελείως ανεξάρτητα.

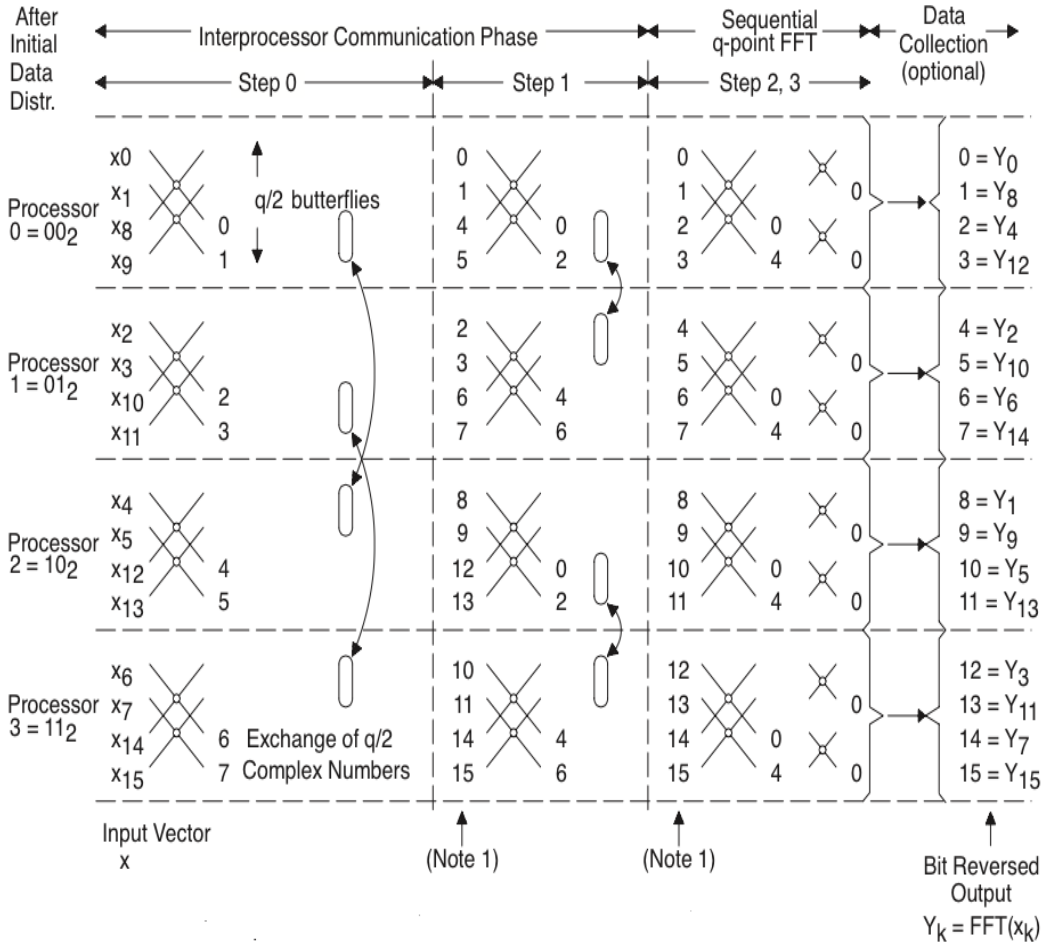
Παρακάτω φαίνεται ο διαχωρισμός των δεδομένων και η ανάθεση τους σε διεργασίες. Έχουμε πολλαπλά δεδομένα ανά διεργασία.Θεωρούμε ότι τα n δεδομένα ενος FFT n σημείων,μοιράζονται σε p διεργασίες, όπου n>p. Το n και p είναι δυνάμεις του 2, δηλαδή n=2^r, p=2^d. Χωρίζουμε την είσοδο σε μπλόκ με n/p συνεχόμενα δεομένα και αναθέτουμε κάθε μπλολ σε μία διεργασία. Σε κάθε στάδιο επικοινωνίας μεταφέρουμε n/p δεδομένα.Έχουμε n=pq, όπου n είναι το μέγεθος των εισερχόμενων δεδομένων και p ο αριθμός των επεξεργαστών που χρησιμοποιούμε.(2-d dimensional hypercube). Τα εισερχόμενα δεδομένα είναι σε κανονική σειρά ενώ η έξοδος της FFT σε διάταξη bit-reversed.

Ο **χρόνος εκτέλεσηςτου παράλληλου αλγορίθμου** είναι ο εξής:

$$T_p = t_c \frac{n}{p}\log(n) + t_s \log(p) + t_w(\frac{n}{p})\log(p)$$

όπου το 1ο μέρος του αθροίσματος αφορα τον χρόνο που χρειαζόμαστε για τους υπολογισμούς,το 2ο μέρος είναι ο χρόνος εκκίνησης που απαιτείται(start up time) και στο 3^o μέρος έχουμε το transfer time.

Για n=16 και p=4 έχουμε :



(figure 3)

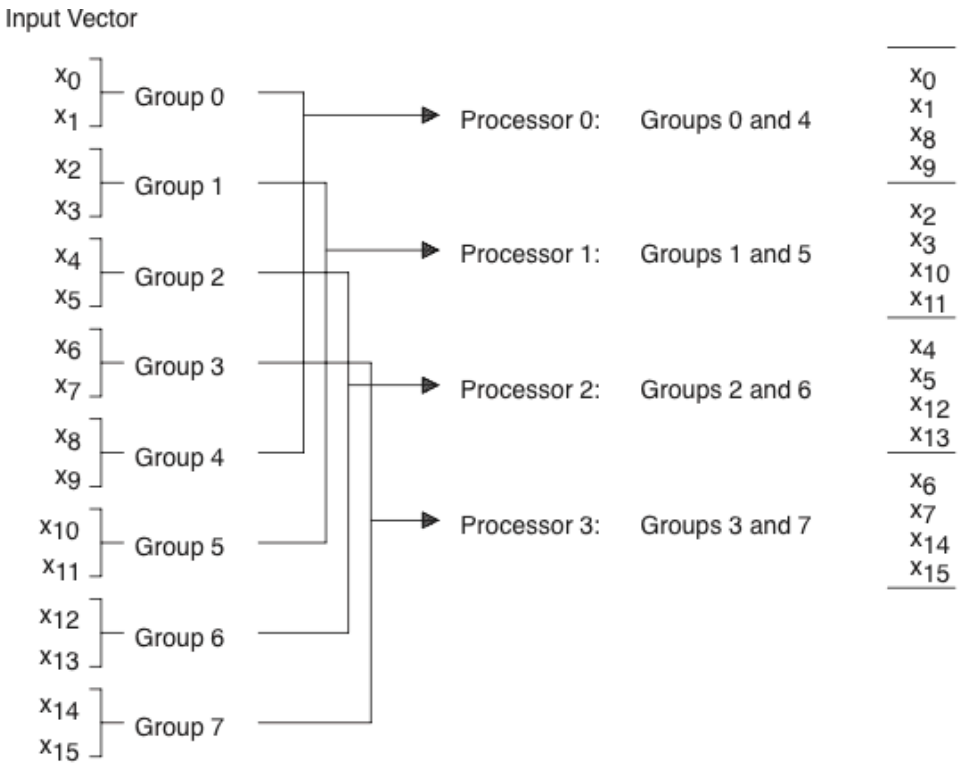
Note 1: Η αρίθμηση αφορά τα εισερχόμενα δεδομένα με την κανονική σειρά.

Επεξήγηση των φάσεων του figure 3:

Phase 1: Data Distribution phase

Το διάνυσμα εισόδου x κατανέμεται διαδοχικά σε 2*p ομάδες των q/2 complex numbers ο καθένας, και δίνονται στον επεξεργαστή i, γκρούπ των των i και i+p. Στο τέλος τησ κατανομής, ο επεξεργαστής i περιέχει τα εξής (figure 4):

(i*q/2)+j και (i*q/2)+n/2+j , όπου 0 < i < p , 0 < j < q/2



(figure 4) example FFT with 16 input elements

Phase 2 : Interprocessor communication phase

Απαιτειται επικοινωνια μεταξυ των επεξεργαστων γιατι οι απαιτουμενοι υπολογισμοί σε εναν επεξεργαστη εξαρτώνται απο τα ενδιάμεσα αποτελέσματα των άλλων επεξεργαστων. Με αυτο το σύστημα mapping d ταυτόχρονα βήματα επικοινωνίας και ανταλλαγής κατα τα d πρώτα στάδια όπου d = P είναι η διάσταση του υπερκύβου σε κάθε βήμα
1) υπολογίζουμε μια πράξη butterfly σε καθένα απο τα ζεύγη πεταλούδας που του αναλογούν και μετά
2) στέλνει τα μισά από τα υπολογισμένα αποτελέσματα (q/2) συνεχόμενα complex numbers) στον κόμβο που το χρειάζεται για το επόμενο βήμα υπολογισμού, ενώ περιμένει για τις πληροφορίες από το ίδιου μήκους μήνυμα (υπολογισμών) που έρχεται απο άλλο κόμβο για τη συνέχιση των υπολογισμών.

Η επιλογή του επεξεργαστή προορισμού καθώς και των δεδομένων προς αποστολή στον συγκεκριμένο κόμβο κατανέμονται στον κάθε επεξεργαστή ως εξής:
-εάν το j bit απο το ID του κόμβου ειναι 0 τότε στέλνει q/2 συνεχόμενων complex numbers (κάτω μισα) στον επεξεργαστή dnode (ID κομβου με j αλλαγμένο)
-αλλιώς στέλνει q/2 συνεχόμενων complex number(άνω μισά) στον επεξεργαστή dnode(ID κόμβου με j αλλαγμένο)

Η μεταβλητή j αρχικά επισημαίνεται στο (logp) -1bit, το πιο σημαντικό bit από το ID του κόμβου και μετατοπίζεται δεξιά μετά απο κάθε βήμα ενδιάμεσης επικοινωνίας . Εξαιτίας αυτής της αλλαγής του j (swaped) η επικοινωνία γίνεται μεταξύ γειτονικών επεξεργαστών σύμφωνα με τον υπερκύβο.

Phase 3: Sequentioal Execution Phase

Στα υπολοιπόενα (n-d) βήματα,δεν πραγματοποιείται επικοινωνία μεταξύ επεξεργαστών. Μία ακολουθιακή FFT (μεγέθους q) πραγματοποιείται σε κάθε επεξεργαστή.

Phase 4: Data Collection

Στο τέλος του FFT, ο επεξεργαστής i περιέχει q complex elements με ανεστραμένη σειρά bit διανυσμάτων i*q+j (0 <= j <= q και 0<= i <= p). Για να συλλέγονται τα δεδομένα, συνεπάγεται, η γραμμική μεταφορά q*2 συννεχόμενων θέσεων μνήμης. Τα συλλεγόμενα αποτελέσματα είναι σε σειρά με ανεστραμένα τα bits τους. Μπορεί στον κεντρικό επεξεργαστή να υλοποιηθεί μία bit αναστροφή σε διανύσματα μεγέθους n.

Με το συγκεκριμένο σύστημα παραλληλισμού μειώνεται η καθυστέρηση επικοινωνίας μεταξύ των επεξεργαστών και εξισσοροπείται ο φόρτος των υπολογισμών. Κάθε επεξεργαστής εκτελεί ίσο αριθμό υπολογισμών butterfly θεωρλωντας το q = n/2 ως ένα ακέραιο αριθμό.

ΑΣΚΗΣΗ 3: ΥΠΟΛΟΓΙΣΜΟΣ ΑΠΟΔΟΣΗΣ ΚΑΙ ΧΡΟΝΟΒΕΛΤΙΩΣΗΣ

Για τον υπολογισμό της απόδοσης και της χρονοβελτίωσης έχουμε τα εξής:

*Ο χρόνος εκτέλεσης του σειριακού αλγορίθμου FFT είναι,

$$\mathrm{T}_s = n\log(n)t_c$$

*Ο χρόνος εκτέλεσης του παράλληλου αλγορίθμου είναι,

$$\mathrm{T}_p = t_c\frac{n}{p}\log(n) + t_s\log(p) + t_w(\frac{n}{p})\log(p)$$

Υπολογίζοντας την απόδοση έχουμε :

$$E_p = \frac{T_s}{pT_p}, \ldots, E_p \to 1$$

$$= \frac{n\log(n)t_c}{\log(\frac{n}{p})t_c + log(p)t_s + \frac{n}{p}\log(p)t_w}$$

Η απόδοση βελτιώνεται καθώς το n και ο χρόνος υπολογισμού αυξάνονται, ενώ αντίστοιχα η απόδοση επιρεάζεται αρνητικά καθώς αυξάνονται τα p, Ts και Tw. Σε συστήματα με διαμοιραζόμενη μνήμη ο χρόνος setup και transfer είναι 0 και συνεπώς δεν υπολογίζονται στην απόδοση.

Υπολογίζοντας την χρονοβελτίωση έχουμε :

$$S_p = \frac{T_s}{T_p} = pE_p,\ldots,S_p \rightarrow p$$

$$= \frac{n\log(n)t_c}{\log(\frac{n}{p})t_c + log(p)t_s + \frac{n}{p}\log(p)t_w}$$

Η χρονοβελτίωση βελτιώνεται όσο αυξάνεται το p. Όπως και στην απόδοση, έτσι και στην χρονοβελτίωση ισχύει το ίδιο (για το Ts και Tw)σε περίπτωση συστήματος με διαμοιραζόμενη μνήμη.

ΑΣΚΗΣΗ 4 : ΥΠΟΛΟΓΙΣΜΟΣ n,p ΠΟΥ ΒΕΛΤΙΣΤΟΠΟΙΕΙ ΤΟΝ ΠΑΡΑΛΛΗΛΟ ΧΡΟΝΟ

Ο χρόνος της παράλληλης εκτέλεσης υπολογίζεται από τον τύπο:

$$\mathrm{T}_p = t_c\frac{n}{p}\log(n) + t_s\log(p) + t_w(\frac{n}{p})\log(p)$$

Όπως γνωρίζουμε για να έχει ο χρόνος εκτέλεσης ακρότατο,συγκεκριμένα μέγιστο,εντός του διαστήματος στο οποίο ορίζεται θα πρέπει να ισχύει dT/dn(a,b) = 0 και dT/dp(a,b) = 0, όπου το (a,b) το σημείο ακροτάτου.