1

ΕΤΕΡΟΓΕΝΗ ΥΠΟΛΟΓΙΣΤΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΕΠΙΤΑΧΥΝΣΗ ΜΕ ΧΡΗΣΗ FPGA

Επικ.Καθηγητής Μιχάλης Ψαράκης

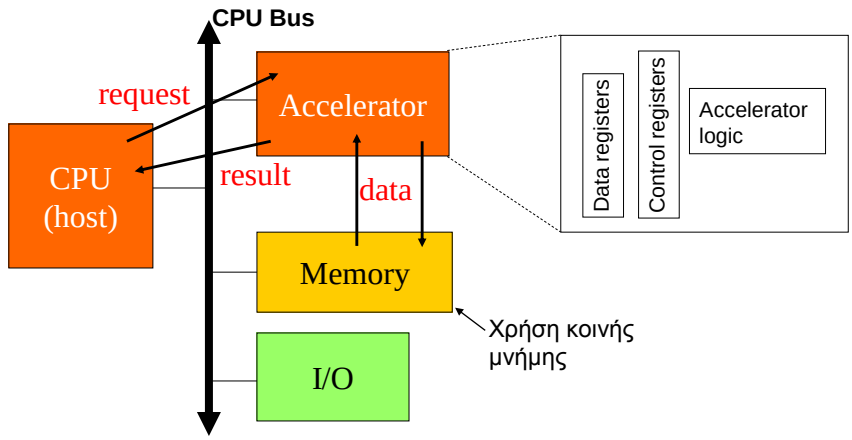
Ενότητα 1

- Επιτάχυνση με χρήση υλικού (hardware-based acceleration)
  - Πότε είναι χρήσιμη;
  - Πώς σχεδιάζουμε επιταχυντές (accelerator);

Επιταχυντές

- Χρήση πρόσθετων υπολογιστικών μονάδων που προορίζονται για την εκτέλεση συγκεκριμένων κρίσιμων λειτουργιών
  - υλικό (hardware)
  - επιπλέον επεξεργαστής
- Συ-σχεδίαση υλικού/λογισμικού (hardware/software co-design)
  - Ταυτόχρονη χρήση υλικού & λογισμικού για την επίτευξη των προδιαγραφών του συστήματος

Αρχιτεκτονική συστήματος με επιταχυντή



Επιταχυντής vs. συν-επεξεργαστής

- Ένας συν-επεξεργαστής εκτελεί εντολές
  - οι εντολές διεκπεραιώνονται από τον επεξεργαστή
- Ένας επιταχυντής λειτουργεί παρόμοια με μια περιφερειακή συσκευή αν και συνήθως δεν υλοποιεί λειτουργίες εισόδου/εξόδου
  - ο επιταχυντής δεν εκτελεί εντολές
  - ο επιταχυντής ελέγχεται με την χρήση καταχωρητών
    - Data & Control καταχωρητές

Υλοποιήσεις επιταχυντή

- Ολοκληρωμένα Κυκλώματα Ειδικής Εφαρμογής (Application-Specific Integrated Circuits, ASICs)
  - Κοστίζει σημαντικά αλλά συμφέρει για προϊόντα που παράγονται σε μεγάλο αριθμό (high-volume products)
- Προγραμματιζόμενες διατάξεις λογικής (Field-Programmable Gate Arrays, FPGAs)
  - Παρέχουν ευελιξία
  - Χρησιμοποιούνται τόσο σε ενσωματωμένες εφαρμογές όσο και σε συστήματα υψηλής απόδοσης

Διαδικασία σχεδίασης συστήματος με επιταχυντή

- Σχεδίαση μίας ετερογενούς πολυ-επεξεργαστικής αρχιτεκτονικής
  - επεξεργαστικά στοιχεία (processing elements, PEs):
    - CPUs
    - Επιταχυντές (ASICs, FPGAs, etc)
- Προγραμματισμός του συστήματος

Γιατί επιταχυντές;

- Η σχεδίαση ενός επιταχυντή απαιτεί επιπλέον προσπάθεια και διαφορετικές επιδεξιότητες από την σχεδίαση ενσωματωμένου λογισμικού
- Για λοιπόν να χρησιμοποιήσει κάποιος επιταχυντές;
  - Το κέρδος θα πρέπει να είναι σημαντικό

Γιατί επιταχυντές;

- Καλύτερος λόγος κόστους/απόδοσης
  - καλύτερη απόδοση και λειτουργικότητα ανά $
  - ο επιταχυντής μπορεί να εκτελέσει γρηγορότερα κάποιες λειτουργίες από ότι ένας επεξεργαστής ίδιου κόστους
  - Το κόστος του επεξεργαστή είναι μη-γραμμική συνάρτηση της απόδοσης

Γιατί επιταχυντές;

- Καλύτερος λόγος κόστους/απόδοσης (συν.)
  - Διαμερισμός της εφαρμογής σε περισσότερα επεξεργαστικά στοιχεία (PE) χαμηλότερης απόδοσης και κόστους
    - Τα αποδοτικά επεξεργαστικά στοιχεία κοστίζουν ακριβά
    - Ακόμα και εάν ληφθεί υπ'όψιν το κόστος συναρμολόγησης, το κόστος σε επίπεδο συστήματος είναι χαμηλότερο
    - Πρέπει να ληφθεί υπ'όψιν το engineering cost
- Καλύτερη απόδοση πραγματικού χρόνου
  - οι χρονικά κρίσιμες λειτουργίες ανατίθενται σε λιγότερο επιβαρυμένα επεξεργαστικά στοιχεία

Γιατί επιταχυντές;

- Καλύτερη επεξεργασία εισόδων/εξόδων σε πραγματικό χρόνο
  - π.χ. engine control
- Καλύτερη απόδοση σε εφαρμογές συνεχούς ροής δεδομένων (streaming data)
  - π.χ. Wireless, multimedia, κλπ.
- Χαμηλότερη κατανάλωση ενέργειας
- Τελικά, ακόμα και ο ταχύτερος επεξεργαστής μπορεί να μην ικανοποιεί τις προδιαγραφές

Σχεδίαση συστήματος με επιταχυντή

- Επιβεβαίωση ανάγκης χρήσης επιταχυντή στο σύστημα
- Σχεδίαση του επιταχυντή
- Σχεδίαση της διεπαφής επεξεργαστή - επιταχυντή

Επιβεβαίωση ανάγκης χρήσης επιταχυντή

- Η λειτουργία που θέλουμε να επιταχύνουμε εκτελείται γρηγορότερα στον επιταχυντή μας από την εκτέλεση ως λογισμικό σε μια CPU;
  - Σίγουρο όφελος εάν η CPU είναι ένας απλός μικροελεγκτής
  - Τι συμβαίνει όμως εάν συγκρίνουμε με μια ταχύτατη CPU?
- Είναι η λειτουργία που θέλουμε να επιταχύνουμε το πραγματικό bottleneck ή κάποια άλλη λειτουργία;
- Μήπως η επιβάρυνση από την μεταφορά δεδομένων από/προς τον επιταχυντή είναι τόσο μεγάλη που καταργεί όλα τα πλεονεκτήματα;

Σχεδίαση του επιταχυντή

- Απαιτείται καλή κατανόηση του αλγορίθμου που θέλουμε να επιταχύνουμε
  - Συνήθως είναι στην μορφή κώδικα σε γλώσσα υψηλού επιπέδου
  - Μετάφραση του αλγορίθμου στο υλικό
    - Ιδιαίτερα δύσκολη διαδικασία
    - Σήμερα παρέχεται αυτοματοποίηση με χρήση high-level synthesis αλλά δεν εγγυάται ότι θα οδηγήσει σε βέλτιστες λύσεις
- Σχεδίαση της διασύνδεσης μεταξύ του επιταχυντή και του διαύλου της CPU
  - Προσδιορισμός απαραίτητων καταχωρητών για την επικοινωνία με το λογισμικό που θα τρέχει στην CPU
  - Λειτουργίες συγχρονισμού για την χρήση κοινής μνήμης
  - Ειδική λογική για διάβασμα και γράψιμο από την μνήμη

Σχεδίαση της διεπαφής επεξεργαστή-επιταχυντή

- Σχεδίαση της διεπαφής από την πλευρά του επεξεργαστή
- Το λογισμικό της εφαρμογής θα πρέπει να επικοινωνεί με τον επιταχυντή και να παρέχει τα απαραίτητα:
  - δεδομένα
  - σήματα ελέγχου για τον χειρισμό των δεδομένων
- Απαιτείται συγχρονισμός της λειτουργίας του επιταχυντή με το λογισμικό ώστε να βεβαιωθεί πως:
  - ο επιταχυντής έχει τα απαραίτητα δεδομένα
  - ο επεξεργαστής λαμβάνει τα επιθυμητά αποτελέσματα

Ανάλυση της απόδοσης

- Όπως σε οποιοδήποτε ενσωματωμένο σύστημα, σε ένα σύστημα με επιταχυντή, η κατανάλωση ισχύος και το κόστος κατασκευής είναι σημαντικές παράμετροι
- Επιτάχυνση: Η πιο κρίσιμη παράμετρος
  - Πόσο πιο γρήγορο είναι το σύστημα με την ενσωμάτωση του επιταχυντή από ότι χωρίς αυτόν?
- Πρέπει να λάβουμε υπόψιν:
  - τον χρόνο εκτέλεσης του επιταχυντή
  - τον χρόνο μεταφοράς δεδομένων
  - τον συγχρονισμό με την κύρια CPU

Χρόνος εκτέλεσης του επιταχυντή

- Ένας επιταχυντής θα διαβάσει δεδομένα, θα τα επεξεργαστεί και θα γράψει πίσω τα αποτελέσματα
- Ο χρόνος εκτέλεσης του επιταχυντή δεν εξαρτάται μόνο από τον χρόνο εκτέλεσης της κύριας λειτουργίας που θέλουμε να επιταχύνουμε
- Πρέπει να ληφθεί υπ'όψιν ο χρόνος που απαιτείται για την μεταφορά των δεδομένων εισόδου/εξόδου
- Συνολικός χρόνος εκτέλεσης:

$$t_{accel} = t_{in} + t_x + t_{out}$$

Χρόνος για μεταφορά δεδομένων εισόδου/εξόδου

- Οι τιμές t_{in} και t_{out} που αφορούν στον χρόνο για την μεταφορά δεδομένων εισόδου (διάβασμα) και εξόδου (γράψιμο) αντίστοιχα, αντικατοπτρίζουν τον χρόνο που απαιτείται για λειτουργίες του διαύλου συμπεριλαμβανομένων των ακολούθων:
  - Τον χρόνο που απαιτείται για το flushing μιας τιμής καταχωρητή ή cache στην κύρια μνήμη
  - Τον χρόνο που απαιτείται για την μεταφορά του ελέγχου μεταξύ της CPU και του επιταχυντή
  - Την επιβάρυνση της μεταφοράς δεδομένων από πακέτα, handshaking, κλπ.

Επιτάχυνση

- Το κύριο βάρος της ανάλυσης της απόδοσης αφορά στην επιτάχυνση που προκύπτει από την υλοποίηση της κρίσιμης λειτουργίας από τον επιταχυντή αντί της υλοποίησης με λογισμικό
- Η επιτάχυνση (speedup) σε σύγκριση με το σύστημα χωρίς επιταχυντή δίνεται από την ακόλουθη σχέση:

$$S = n\,(t_{CPU} - t_{accel}) = n\,[t_{CPU} - (t_{in} + t_x + t_{out})]$$

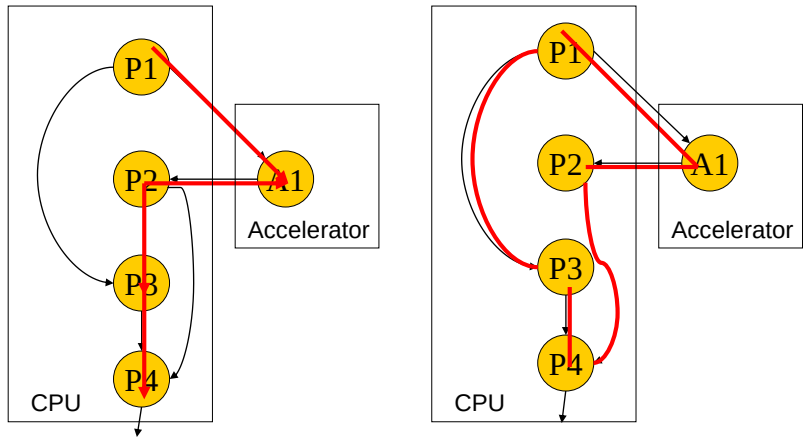
  - n: ο αριθμός των επαναλήψεων που εκτελείται η κρίσιμη λειτουργία του επιταχυντή
  - t_{CPU} :χρόνος εκτέλεσης της κρίσιμης λειτουργίας με λογισμικό

Παραλληλία εκτέλεσης

- Σημαντικός παράγοντας στην επιτάχυνση είναι η δυνατότητα παράλληλης εκτέλεσης
  - Blocking σύστημα: ο επεξεργαστής περιμένει (κατάσταση idle) τον επιταχυντή
  - Non-blocking σύστημα: ο επεξεργαστής συνεχίζει την εκτέλεση των διεργασιών παράλληλα με τον επιταχυντή
- Η δυνατότητα παραλληλίας στην εκτέλεση προϋποθέτει ότι το λειτουργικό σύστημα του επεξεργαστή υποστηρίζει παράλληλες διεργασίες

Παραλληλία και χρόνος εκτέλεσης

- Blocking
- Non-blocking

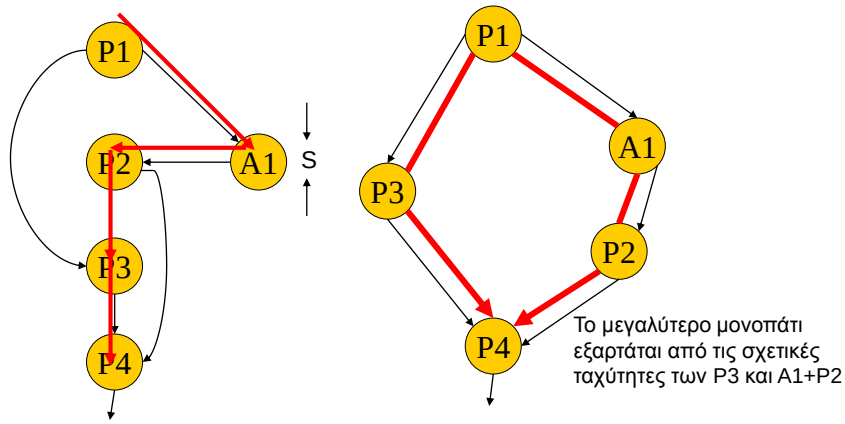


Ανάλυση συνολικού χρόνου εκτέλεσης

- Δεν μας ενδιαφέρει τόσο η επιτάχυνση που παρέχει ο επιταχυντής, όσο η επιτάχυνση ολόκληρου του συστήματος
  - Πόσο πιο γρήγορα εκτελείται ολόκληρη η εφαρμογή
- Σε σύστημα χωρίς παραλληλία στην εκτέλεση (blocking), ο συνολικός χρόνος εκτέλεσης μειώνεται κατά ***S***
  - Ένα μοναδικό μονοπάτι εκτέλεσης
- Σε σύστημα με παραλληλία στην εκτέλεση (non-blocking), ο συνολικός χρόνος εκτέλεσης εξαρτάται από το μεγαλύτερο μονοπάτι
  - π.χ. (P1, P3, P4) or (P1, A1, P2, P4)

Ανάλυση συνολικού χρόνου εκτέλεσης

- Blocking
- Non-blocking



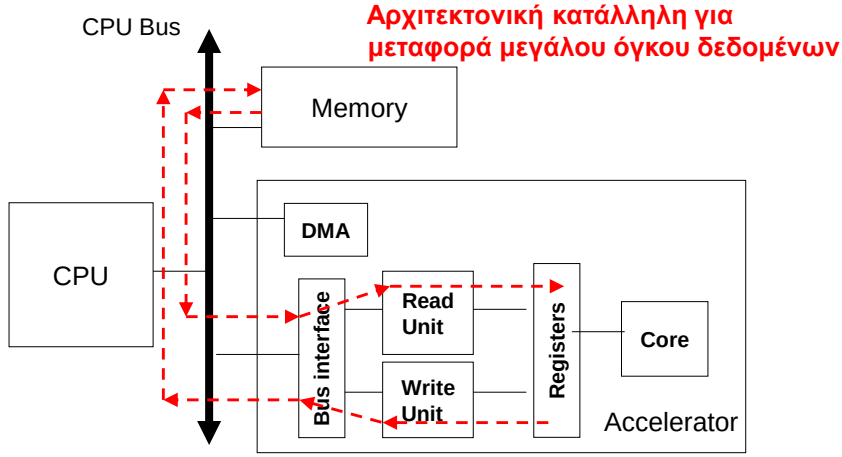
Πηγές παραλληλίας

- Επικάλυψη I/O και λειτουργίας του επιταχυντή
  - εκτέλεση λειτουργιών σε δέσμη δεδομένων (batches)
  - ανάγνωση της δεύτερης δέσμης ενώ ο επιταχυντής επεξεργάζεται την πρώτη δέσμη
- Ο επεξεργαστής εκτελεί άλλες λειτουργίες
  - επαναδρομολόγηση λειτουργιών ώστε κάποιες να εκτελεστούν αφού ενεργοποιηθεί ο επιταχυντής

Αρχιτεκτονική συστήματος επεξεργαστή-επιταχυντή

- Ο επιταχυντής παρέχει καταχωρητές για έλεγχο και παρακολούθηση της κατάστασης από τον επεξεργαστή (control/status registers)
  - Βασικές λειτουργίες ελέγχου start, stop, reset
- Μεταφορά μικρού αριθμού δεδομένων
  - χρήση καταχωρητών δεδομένων
- Μεταφορά μεγάλου αριθμού δεδομένων
  - χρήση αρχείου καταχωρητών
  - χρήση ειδικής λογικής ανάγνωσης/εγγραφής
    - π.χ. DMA

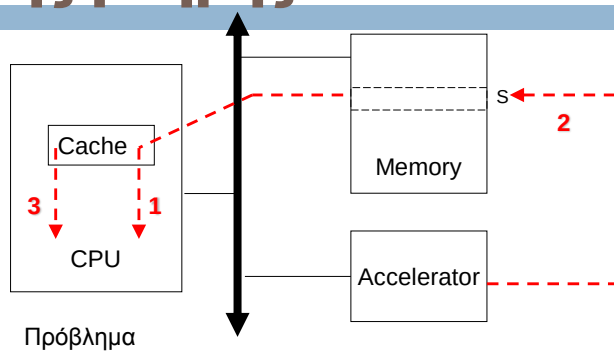
Αρχιτεκτονική συστήματος επεξεργαστή-επιταχυντή



Προβλήματα με την χρήση της μνήμης cache

- Η χρήση της μνήμης cache μπορεί να δημιουργήσει προβλήματα σε εφαρμογές ενσωματωμένων συστημάτων με επιταχυντή
- Η κύρια μνήμη παρέχει τον κύριο μηχανισμό μεταφοράς δεδομένων από/προς τον επιταχυντή
- Ας θεωρήσουμε το εξής σενάριο:
  1. Η CPU διαβάζει από την διεύθυνση S
  2. Ο επιταχυντής γράφει στην διεύθυνση S
  3. Η CPU διαβάζει ξανά από την διεύθυνση S

Προβλήματα με την χρήση της μνήμης cache



- Πρόβλημα
  - Εάν η CPU φέρει την διεύθυνση S στην cache, το πρόγραμμα δεν θα δει την τιμή στην S που έγραψε ο επιταχυντής αλλά θα διαβάσει την παλαιά τιμή
- Λύσεις
  - Ενημέρωση της cache για invalid entries
  - Χρήση κατάλληλων CPU που υποστηρίζουν multiprocessing
    - Υποστήριξη μηχανισμών που ενημερώνουν για αλλαγές στην cache

Συστήματα με επιταχυντή

- Πολλά προϊόντα είναι διαθέσιμα για επιτάχυνση στο PC
  - PCI cards
  - βασίζονται σε FPGAs
  - διασύνδεση με τον επεξεργαστή μέσω «γρήγορης» διασύνδεσης, π.χ. PCIe bus

Ενότητα 2

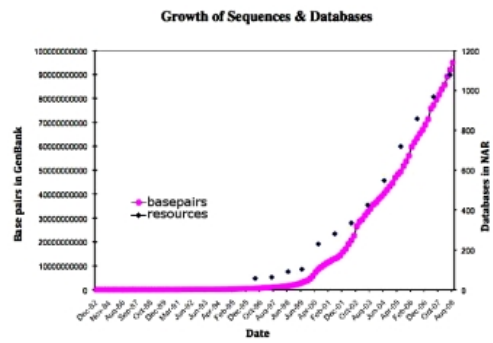
- Παραδείγματα επιταχυντών με χρήση FPGA (FPGA-based accelerator)
  - Επιταχυντής αλγορίθμου Smith-Waterman για ευθυγράμμιση μοριακών ακολουθιών
  - Επιταχυντής βίντεο (video accelerator)

Ευθυγράμμιση μοριακών ακολουθιών

- Base-paired DNA sequences: ATCGATTGAGCTCTAGCG
  - Βάσεις (σύμβολα ακολουθίας): A T C G
- Ευθυγράμμιση ακολουθιών DNA (sequence alignment): σύγκριση ομοιότητας ακολουθιών
  - Εύρεση μεταλλάξεων (mutation)
    - Διαφορετικοί τύποι μεταλλάξεων: deletion, insertion, substitution
- *Deletion*: αφαίρεση μιας ή περισσοτέρων συνεχόμενων βάσεων (substring)
- …TT**GA**TCA… => …TTTCA…
- *Insertion*: προσθήκη υπο-ακολουθίας
- …GGCTAG… => …GG**TCAAC**TAG…
- *Point mutation*: αντικατάσταση μιας βάσης
- …ACG**G**CT… => …ACG**C**CT…

Υπολογιστική πολυπλοκότητα

- Οι βάσεις δεδομένων γενετικών ακολουθιών μεγαλώνουν ταχύτατα
- Απαιτείται σύγκριση ακολουθιών DNA με βάσεις δεδομένων
- Η πολυπλοκότητα της σύγκρισης είναι ανάλογη του γινομένου του μεγέθους της ακολουθίας υπό αναζήτηση και της βάσης δεδομένων



⇒ ***Η σύγκριση είναι πολύ αργή σε συμβατικούς υπολογιστές***

Αλγόριθμος Smith-Waterman

- Βέλτιστη τοπική ευθυγράμμιση (local alignment) δύο ακολουθιών
- Εκτελεί εξαντλητικό έλεγχο
  - Πολυπλοκότητα $O(n \times m)$ για ακολουθίες μήκους n και m
- Βασίζεται σε έναν αλγόριθμο δυναμικού προγραμματισμού (dynamic programming)
  - Υπολογίζει μια μήτρα nxm με βάση μια συνάρτηση κόστους ευθυγράμμισης
  - Βρίσκει την μέγιστη τιμή (score) στην μήτρα
  - Με αναδρομή (trace back) από την μέγιστη τιμή βρίσκει τα όρια που ταιριάζουν οι 2 ακολουθίες

Αλγόριθμος Smith-Waterman

- Ευθυγράμμιση των ακολουθιών *S1*, *S2* με μήκος *l1*, *l2* με συνάρτηση ανάδρασης:

$$H(i,j) = \max \begin{cases} 0 \\ E(i,j) \\ F(i,j) \\ H(i-1,j-1) + Sbt(S1_i, S2_j) \end{cases} \quad ,1 \le i \le l1, 1 \le j \le l2$$

$$H(i,0) = E(i,0) = 0 \qquad H(0,j) = F(0,j) = 0$$

$$E(i,j) = \max \begin{cases} H(i,j-1) - \alpha \\ E(i,j-1) - \beta \end{cases} \quad , \quad F(i,j) = \max \begin{cases} H(i-1,j) - \alpha \\ F(i-1,j) - \beta \end{cases}$$

- Τρεις πιθανοί τρόπου για να επεκταθεί μια ευθυγράμμιση
  - Κατά ένα σύμβολο σε κάθε ακολουθία
  - Κατά ένα σύμβολο στην S1 και ευθυγράμμιση με ένα κενό στην S2
  - Κατά ένα σύμβολο στην S2 και ευθυγράμμιση με ένα κενό στην S1

Αλγόριθμος Smith-Waterman

S1=*ATCTCGTATGATG* S2=*GTCTATCAC*

$$Sbt(x,y)=\begin{cases}2 \text{ if } (x=y)\\ -1 \text{ else}\end{cases}$$

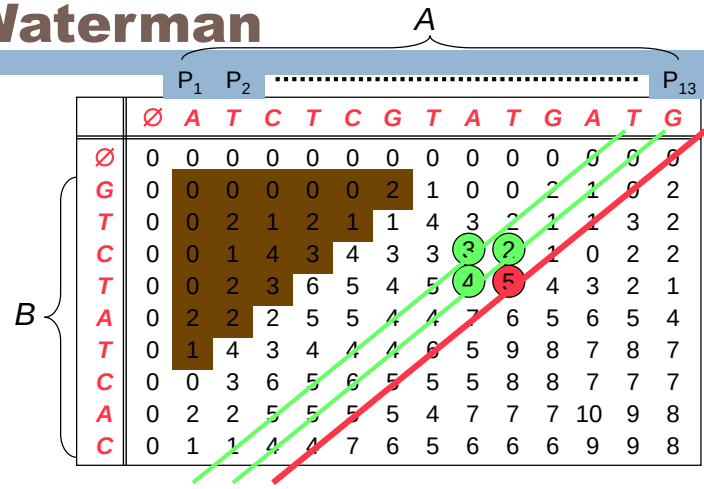
α=1, β=1

	∅	A	T	C	T	C	G	T	A	T	G	A	T	G
∅	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	2	1	0	0	2	1	0	2
T	0	0	2	1	2	1	1	4	3	2	1	1	3	2
C	0	0	1	4	3	4	3	3	3	2	1	0	2	2
T	0	0	2	3	6	5	4	5	4	5	4	3	2	1
A	0	2	2	2	5	5	4	4	7	6	5	6	5	4
T	0	1	4	3	4	4	4	6	5	9	8	7	8	7
C	0	0	3	6	5	6	5	5	5	8	8	7	7	7
A	0	2	2	5	5	5	5	4	7	7	7	10	9	8
C	0	1	1	4	4	7	6	5	6	6	6	9	9	8

$$H(i,j)=\max\begin{cases}0\\ H(i-1,j)-1\\ H(i,j-1)-1\\ H(i-1,j-1)+Sbt(S1_i,S2_j)\end{cases}$$

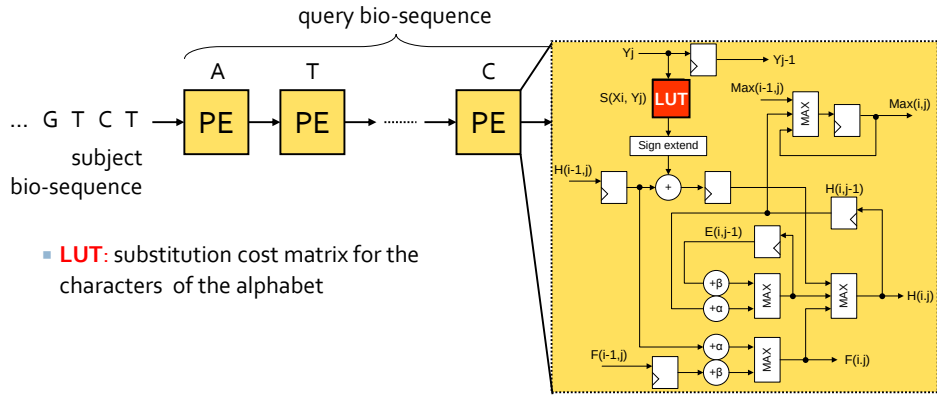
A T C T C G T A T G A T G
G T C – T A T C A C

Parallelization of Smith-Waterman



- Τα κελιά της μήτρας μιας διαγωνίου μπορούν να υπολογιστούν παράλληλα
- Ο υπολογισμός του πίνακα μπορεί να εκτελεστεί σε A+B–1 βήματα σε A υπολογιστικές μονάδες

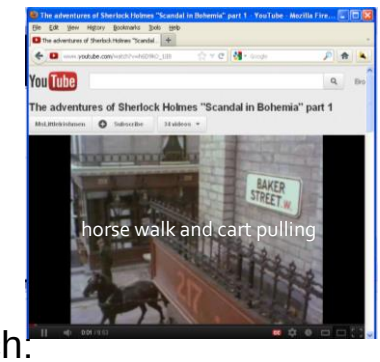
Systolic array



- **LUT**: substitution cost matrix for the characters of the alphabet

Application of SW in audio

- Automated annotation of digital movies
  - tracking audio events in the audio streams
  - audio scene description

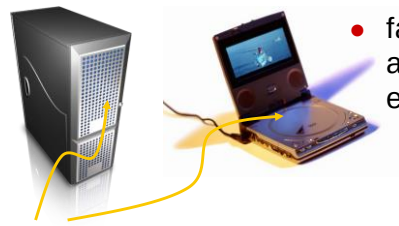


- Query-by-example approach:
  - Sound effects from commercial databases
    - accompanied by textual description
    - similar effects are grouped to form semantic clusters

Goal

Future goal:

- Web-servers for rapid annotation
  - metadata for indexing and retrieval operations
- Next-generation DVD players
  - fast browsing, automated subtitling, etc.



audio tracking module: core technology

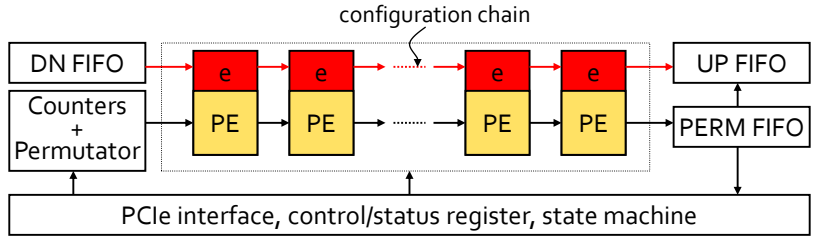
target ➡

FPGA-based acceleration of time consuming parts

Problem formulation

- Inspired by the problem of **molecular sequence alignment** from bioinformatics
- Tracking an audio effect in a movie:
  - transform audio streams into sequences of feature vectors
  - calculate similarity matrices based on the feature vectors
  - perform sequence alignment based on a variant of the **Smith-Waterman algorithm**

Proposed FPGA accelerator



- e-PE: enhanced processing element
  - original PE + fixed-point arithmetic, configuration chain, freeze/reset
- Permutator:
  - variant of the Fisher-Yates shuffle algorithm
- FIFOs:
  - DN FIFO: similarity matrix, UP FIFO: alignment score matrix, PERM FIFO: results of the random permutations

Alignment process

Host PC | **FPGA accelerator**

Calculation of the similarity matrix

Write the similarity matrix to the FPGA accelerator

Shift-in the similarity matrix to the systolic array via the configuration chain

Execution of the Smith-Waterman algorithm

Shift-out the alignment scores from the systolic array via the configuration chain

Similarity threshold

Read the alignment scores from the FPGA accelerator

Execution of the Smith-Waterman algorithm for random permutations of the audio effect

Wilcoxon test

Read the permutation results from the FPGA accelerator

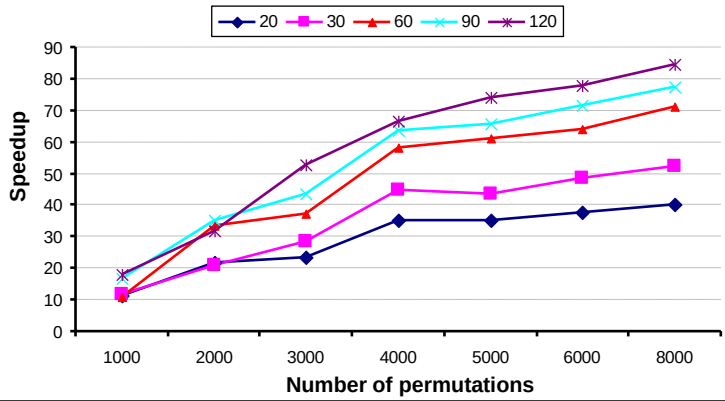
Backtracking

FPGA platform

- Xilinx ML505 development board
  - Virtex-5 XC5VLX50T device
  - PCIe interface
- Accommodates 100 PEs with 128-entries LUTs
  - movie segment: max. 100 frames (i.e. 4.64sec)
    - Frame: 46.4 ms, 1024 samples at 22.05KHz sampling rate
  - audio effect: max. 128 frames (i.e. 5.94sec)
- Comparison with the typical PE (from biosequence SW)
  - 25% area overhead
  - no performance degradation

Speedup

- Experimental setup:
  - short movie window: 13min -> 168 segments x 100 frames
  - five audio effects: 20, 30, 60, 90 and 120 frames
  - number of permutations: from 1000 up to 8000



- Compared with the CPU execution of the method (C, gcc with -O3)
- Intel Core2Duo E8400, @ 3.6GHz

Execution times

- Average execution time of a single audio effect tracking in a movie segment:
  - for different audio effects (20 to 120 frames)
  - for different number of permutations

K	FPGA (ms)	SW (ms)	Speedup
1000	6.7	174.2	26
2000	7.0	343.7	49
3000	8.5	551.5	65
4000	8.8	744.3	85
5000	11.1	942.2	85
6000	13.1	1127.8	86
8000	17.2	1501.0	87
Average	**10.3**	**764.3**	**74**

- Execution time of an audio effect tracking in an 1-hour movie
  - FPGA: ≈ 8 sec, SW ≈ 9.9 min

Βασικές αρχές συμπίεσης βίντεο

- Πλεονάζουσα πληροφορία στο πεδίο του χώρου
  - Ενδοπλαισιακή (Intra-frame) συμπίεση
    - Χρήση τεχνικών συμπίεσης που εφαρμόζονται στην πληροφορία μόνο μέσα στο ίδιο το πλαίσιο (frame)
      - Συμπίεση (lossy ή lossless) πλαισίου παρόμοια με JPEG, π.χ quantization (κβάντιση συντελεστών)
- Πλεονάζουσα πληροφορία στο πεδίο του χρόνου
  - Διαπλαισιακή (Inter-frame) συμπίεση
    - Χρήση τεχνικών συμπίεσης που εκμεταλλεύονται τον τρόπο που μεταβάλλεται η πληροφορία μεταξύ διαδοχικών πλαισίων

Διαπλαισιακή (Inter-frame) συμπίεση

- Οι εικόνες σε ένα βίντεο (συνήθως) δεν μεταβάλλονται σημαντικά σε μικρά χρονικά διαστήματα
  - Μικρή διαφοροποίηση λόγω κίνησης (κάμερας, αντικειμένου)
    - πχ. το φόντο σε ένα video conference μένει σταθερό. Κινούνται μόνο τα χείλη των ανθρώπων
- Η κωδικοποίηση κάθε επόμενου πλαισίου μπορεί να βασιστεί στην πληροφορία του προηγούμενου
- Δεν χρειάζεται να αποθηκευθεί ολόκληρη η πληροφορία του επόμενου πλαισίου παρά μόνο το τμήμα που διαφέρει από το προηγούμενο πλαίσιο

Εφαρμογή επιταχυντή στην συμπίεση βίντεο

- Χρήση επιταχυντή για την εκτίμηση της κίνησης μπλοκ (block motion estimation)
- Εκτελεί δισδιάστατη συσχέτιση (two-dimensional correlation)
  - Εύρεση του καλύτερου ταιριάσματος μεταξύ περιοχών από δύο γειτονικά πλαίσια

Εκτίμηση κίνησης του μπλοκ

- Διαίρεση του πλαισίου (frame) σε macroblocks
  - Σύνηθες μέγεθος ενός macroblock = 16 x 16 pixels
- Για κάθε macroblock (MB), εύρεση της περιοχής του προηγούμενου πλαισίου (πλαίσιο αναφοράς) με την μεγαλύτερη ομοιότητα
  - Η έρευνα σε ολόκληρο το πλαίσιο δεν είναι αποδοτική από πλευράς κόστους
    - Περιορισμός της έρευνας σε μια περιοχή γύρω από το MB
    - Η περιοχή της έρευνας (search area) είναι μεγαλύτερη από το MB

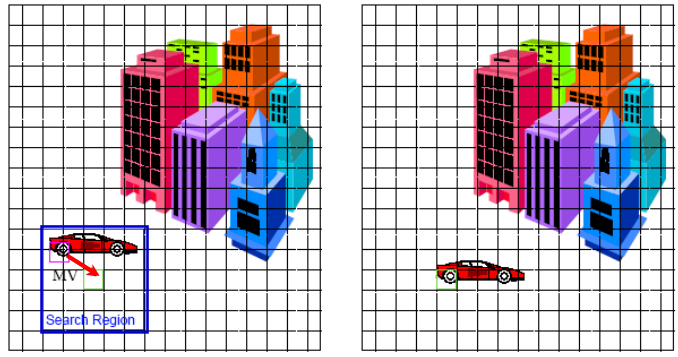
Εκτίμηση κίνησης του μπλοκ

- Ο υπολογισμός της ομοιότητας μπορεί να γίνει με την χρήση αθροίσματος απόλυτων διαφορών (Sum-of-Absolute-Differences, SAD)

$$\sum_{1\le i,j\le n} \left| M(i,j) - S(i-o_x, j-o_y) \right|$$

- *n:* το μέγεθος του macroblock
- *M(i,j):* η ένταση του Macroblock στο pixel i,j
- *S(i,j):* η ένταση της περιοχής έρευνας (Search area) στο pixel i,j
- $<o_x,o_y>$: η μετατόπιση μεταξύ του macroblock και της περιοχής έρευνας
- Η ένταση συνήθως υπολογίζεται ως η 8-bit φωτεινότητα (luminance)

Ιδέα εκτίμησης κίνησης (motion estimation)



Frame *t-1*
(Reference Frame)

Frame *t*
(Predicted frame)

Καλύτερο ταίριασμα

- Επιλέγουμε ως καλύτερο ταίριασμα την θέση στην περιοχή έρευνας του πλαισίου αναφοράς, που ελαχιστοποιεί το μέτρο του αθροίσματος απόλυτων διαφορών (SAD)
- Το καλύτερο ταίριασμα παράγει ένα διάνυσμα κίνησης (motion vector)
  - Το διάνυσμα κίνησης ορίζεται από το κέντρο της περιοχής έρευνας (πλαίσιο αναφοράς) προς το κέντρο του macroblock (πλαίσιο προς κωδικοποίηση)

Καλύτερο ταίριασμα

- Για λόγους απλότητας στην περιγραφή, ο αλγόριθμος που θα υλοποιήσουμε αφορά σε full search
  - Σύγκριση του macroblock και της περιοχής έρευνας σε κάθε σημείο
  - Δεν είναι ο καλύτερος από πλευράς απόδοσης
  - Υπολογιστικά δαπανηρές απαιτήσεις

Αλγόριθμος

```
bestx = 0; besty = 0; bestsad = MAXSAD;
for (ox = - SEARCHSIZE; ox <= SEARCHSIZE; ox++) {
    for (oy = -SEARCHSIZE; oy <= SEARCHSIZE; oy++) {
            int result = 0;
            for (i=0; i<MBSIZE; i++) {
                    for (j=0; j<MBSIZE; j++) {
                            result += abs(mb[i][j] – search[i-ox] [j-oy]);
                    }
            }
            if (result <= bestsad) {
                    bestsad = result; bestx = ox; besty = oy; }
    }
}
```

Υπολογιστικές απαιτήσεις

- MBSIZE = 16, SEARCHSIZE = 8
- Περιοχή έρευνας: 8 + 1 + 8 σε κάθε διάσταση
- Για κάθε macroblock πρέπει να εκτελεστούν:
  - n_{ops} =(16 x 16) x (17 x 17) = 73,984 υπολογισμοί SAD
- Για ένα βίντεο με σχετικά χαμηλή ανάλυση (Common Intermediate Format, CIF) το μέγεθος του πλαισίου είναι 352 x 288 pixels
  - 352 x 288 pixels -> πίνακας 22 x 18 macroblocks
  - Απαιτούνται 29,297,664 υπολογισμοί SAD για κάθε πλαίσιο
- Δεν γίνεται εκτίμηση κίνησης σε κάθε πλαίσιο ενός βίντεο

Επιτάχυνση με FPGA

- Υλοποίηση σε FPGA σε κάρτα PCI
- Απαιτείται μεγάλο μέγεθος μνήμης για τα δεδομένα:
  - το macroblock έχει 16 x 16 = 256 pixels
  - η περιοχή έρευνας έχει $(8+8+1+8+8)^2$=1,089 pixels
    - 8-bit φωτεινότητα (luminance)
- Χρήση εξωτερικής μνήμης στην κάρτα PCI
  - ιδιαίτερα εάν αποθηκεύονται πολλαπλά macroblocks και περιοχές έρευνας

Οργάνωση του επιταχυντή