GPUs, Accelerators and Architectural Specialization

Μάιος 2023

Επιταχυντές και Αρχιτεκτονικές

- Τάσεις προς εξειδικευμένες αρχιτεκτονικές, πλεονεκτήματα και προκλήσεις
- Παραδείγματα – case studies
  - GPUs
  - Deep Learning
  - Graphs
- Ανάλυση και τάσεις
  - Εξειδικευμένες αρχιτεκτονικές και όρια; είναι πραγματικά απαραίτητες?
  - Υπολογισμοί των ερχόμενων 5-10 χρόνων

Τι είναι η Εξειδίκευση

- Αρχιτεκτονικές που σχεδιάζονται με στόχο μια εξειδικευμένη κλάση υπολογισμών
  - Βελτιστοποιήσεις που έχουν νόημα για συγκεκριμένες εφαρμογές
  - Μερικές φορές είναι "fixed-function" – δηλαδή εκτελούν μόνο τις συγκεκριμένες εφαρμογές – συνήθως όμως έχουν κάποια ελευθερία για configurability / programmability
  - Γίνεται ολοένα και πιο γκρι το τοπίο που να τραβήξει κάποιος μια διαχωριστική γραμμή

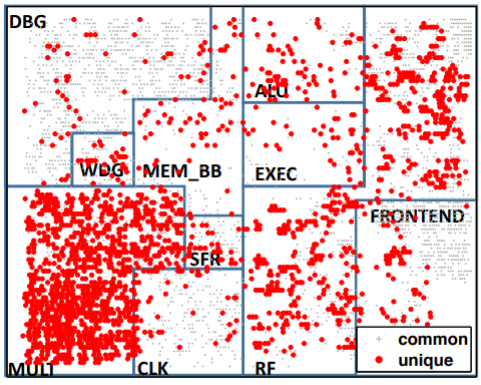
3

Μέτρια Εξειδίκευση

- [Cherupalli et al, ISCA'17]

- Profile applications για να ανακαλυφθούν οι πύλες που χρησιμοποιούνται
- **Εξάλειψη όλων των υπόλοιπων από τον επεξεργαστή**

- Εξοικονόμηση: 62% area, 50% power
- Πλήρως αυτοματοποιημένη διαδικασία
- Βασίζεται στην επαλήθευση και τα εργαλεία σχεδίασης ενός baseline επεξεργαστή



(b) binSearch

4

Υπερβολική Εξειδίκευση

- "Race logic" [Madhavan et al, ISCA'14]
- Υπολογισμός ελάχιστου μονοπατιού με χρήση ενός graph:
  - Nodes mapped onto PEs
  - PEs connected via on-chip network
  - PEs signal each other, adding delay according to edge weight between source and destination
  - The *delay* from source to destination gives the shortest path in the graph
  - PEs very simple ➔ lots of PEs & fast
- Βασικό υπολογιστικό κύκλωμα που μπορεί να λύσει αρκετά προβλήματα, όπως DNA alignment:

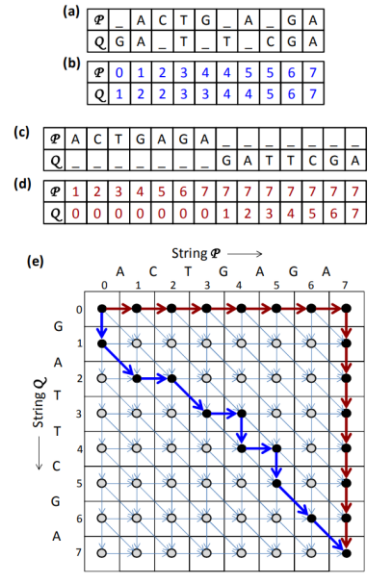


Figure 1. (a, c) Two possible alignments between strings $\mathcal{P}$ and Q and (b, d) their corresponding alignment matrixes and (e) edit graph.

5

Τι Προσφέρει η Εξειδίκευση

Πλεονεκτήματα

- Hard, έλεγχος με ελάχιστη κατανάλωση ισχύος
- Προσαρμοσμένες μονάδες υπολογισμών
- Προσαρμοσμένες και απευθείας επικοινωνίες (όχι καταχωρητές και cache)
- Προσαρμοσμένο σύστημα μνήμης
- Υπερβολικός παραλληλισμός με βάση γνώση από την εφαρμογή
- Η ενέργεια και ο χώρος ξοδεύεται μόνο για χρήσιμη δουλειά

Μειονεκτήματα – Προκλήσεις

- Εύρος εφαρμογών, πόσα διαφορετικά προγράμματα μπορεί να τρέξει το chip ?
- Αντικρούεται για αποδοτικότητα και ευελιξία
- Είναι καλύτερο να επιταχυνθεί το 1% των εφαρμογών σε ποσοστό 100% ή όλες οι εφαρμογές σε ποσοστό 1% ?
- Ολοκλήρωση σε επίπεδο συστήματος
  - Πως μπορούν οι χρήστες να χρησιμοποιήσουν έναν επιταχυντή?
  - Επικοινωνία μεταξύ επιταχυντών και πυρήνων ?

Κόστος ελέγχου και σχεδίασης...

6

Γιατί Είναι Σημαντική η Εξειδίκευση Σήμερα?

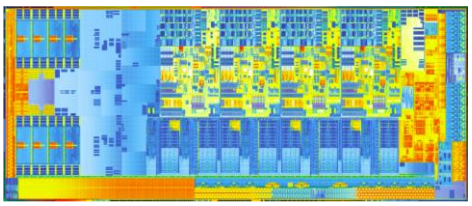
- Υπάρχουν όρια στον παραλληλισμό και στον Amdahl's Law
  - Η αποκόμιση καλών επιδόσεων από multicore αρχιτεκτονικές είναι δύσκολο
  - Η εξειδίκευση (specialization) δίνει 100× perf/energy for "free"
- Σήμερα υπάρχουν νέα σημαντικά workloads
  - Ειδικά σχετικά με deep learning!
  - Έρευνες δημοσιεύονται σε deep learning στα συνέδρια με εκθετικό ρυθμό

7

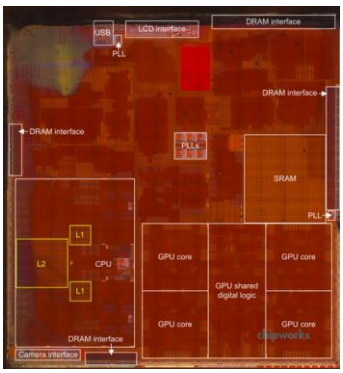
GPU σε Μοντέρνα Συστήματα

- Διακριτές GPUs
  - PCIe-based accelerator
  - Separate GPU memory
- Ολοκληρωμένες GPUs
  - CPU and GPU on same die
  - Shared main memory and last-level cache

Nvidia Kepler



Intel Ivy Bridge, 22nm 160mm2



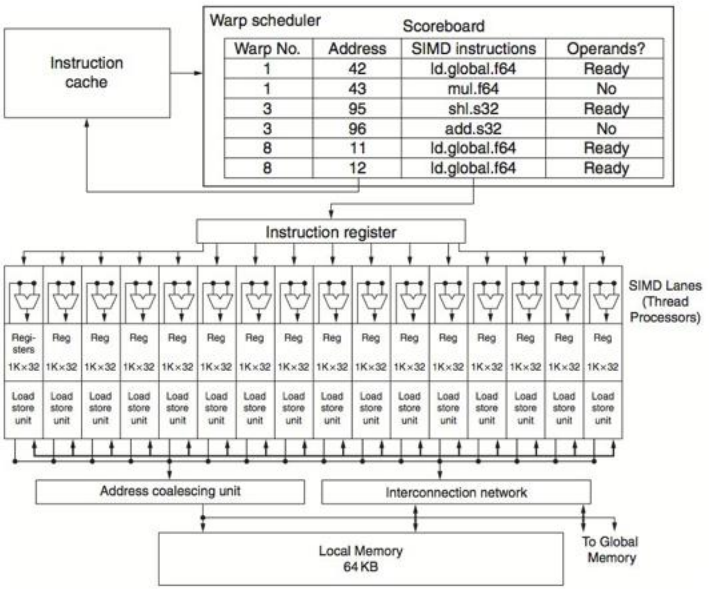
Apple A7, 28nm
TSMC, 102mm2

8

Streaming MP

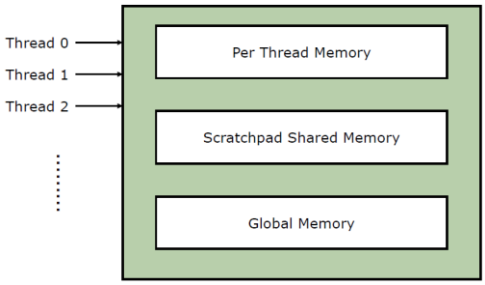
- Κάθε SM έχει warps, με 32 threads/warp
- Fetch 1 instr/cycle
- Issue 1 ready instr/cycle
- Instr broadcast to all lanes

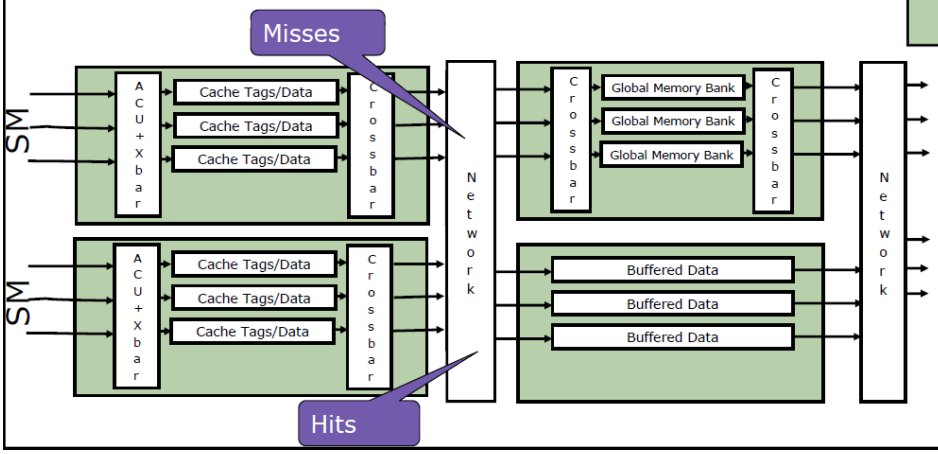
- Multithreading είναι ο βασικός μηχανισμός για να καλυφθεί η καθυστέρηση



9

Πολλά Είδη Μνήμης σε GPU



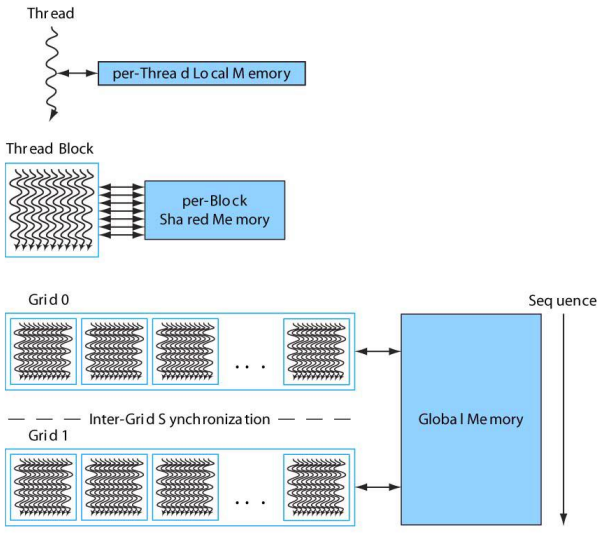


Ιεραρχία μνήμης με caches
– Cache για εξοικονόμηση bandwidth μνήμης
– Caches επιτρέπουν ακόμα την (από-)συμπίεση των δεδομένων

10

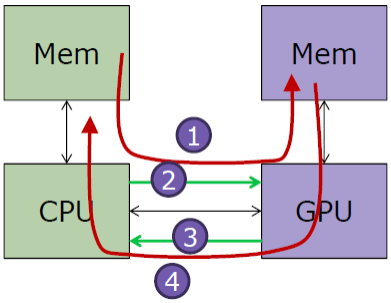
Μοντέλο CUDA GPU

- Μοντέλο ένα πρόγραμμα-πολλαπλά δεδομένα (SPMD)
- Ένα thread έχει καταχωρητές και τοπική μνήμη
- Παράλληλα threads ομαδοποιούνται σε Blocks
  - Τα blocks έχουν κοινόχρηστη μνήμη
  - Γίνεται συγχρονισμός στα threads μέσω barriers
- Τα Grid περιλαμβάνουν ανεξάρτητα blocks
  - Μπορούν να εκτελούνται ταυτόχρονα
  - Μοιράζονται την global μνήμη
  - Έχουν περιορισμένο συγχρονισμό



11

Εκτέλεση ενός GPU Kernel



1. Μεταφορά δεδομένων εισόδου από την μνήμη του CPU στην μνήμη του GPU
2. Ενεργοποίηση του kernel (grid)
3. Αναμονή για να ολοκληρωθεί η εκτέλεση του kernel (synchronous)
4. Μεταφορά των αποτελεσμάτων στην μνήμη του CPU

- Οι μεταφορές δεδομένων έχουν μεγάλο ρόλο στον χρόνο εκτέλεσης
- Νέες GPU είναι ολοκληρωμένες με τον επεξεργαστή και εκμεταλλεύονται τον ενοποιημένο χώρο διευθύνσεων
  - Δεν γίνονται μεταφορές δεδομένων, αλλά τώρα υπάρχει ανταγωνισμός για την πρόσβαση στη μνήμη

12

Παράδειγμα Κώδικα, C vs CUDA

```
// Invoke DAXPY
daxpy(n, 2.0, x, y);
// DAXPY in C
void daxpy(int n, double a, double *x, double *y)
{
        for (int i = 0; i < n; ++i)
                y[i] = a*x[i] + y[i];
}
```

```
// Invoke DAXPY with 256 threads per block
__host__
int nblocks = (n+ 255) / 256;
    daxpy<<<nblocks, 256>>>(n, 2.0, x, y);
// DAXPY in CUDA
__device__
void daxpy(int n, double a, double *x, double *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n) y[i] = a*x[i] + y[i];
}
```

- Ο κώδικας σε CUDA ξεκινά 256 threads ανά block

13

GPU ISA

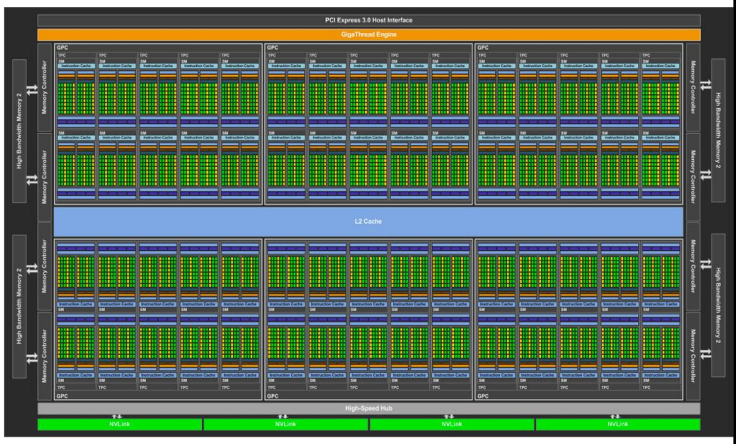
- Η αρχιτεκτονική της GPU και το σετ εντολών αλλάζουν συχνά
- Πρακτικά είναι δύσκολο ο κώδικας να είναι portable, συνήθως είναι προσαρμοσμένος σε μια συγκεκριμένη αρχιτεκτονική GPU

- Scheduling: κάθε kernel είναι non-preemptive
  - Η διαχείριση των πόρων συνήθως γίνεται από τον driver, δεν αποκαλύπτεται στο Λ.Σ.
  - Πρόσφατα παρέχεται υποστήριξη για βασικά χαρακτηριστικά εικονικής μνήμης

14

GPU

- Παράδειγμα: Nvidia Pascal GP100 (2016)
- 60 streaming multiprocessors (SMs)
  - 4MB Shared L2 cache
  - 8 memory controllers
  - 720 GB/s (HBM2)
- Fixed-function logic for graphics (texture units, raster ops, …)
- Scalability: change number of cores and memory channels
- Scheduling κυρίως ελέγχεται από το hardware



15

Πρώιμοι Επιταχυντές για DNN

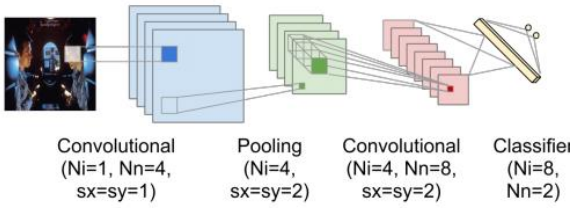
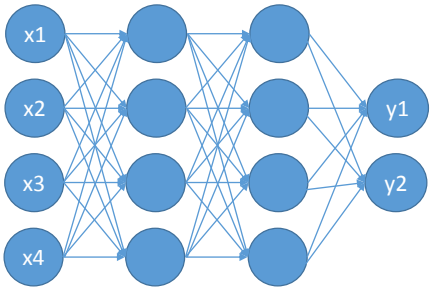


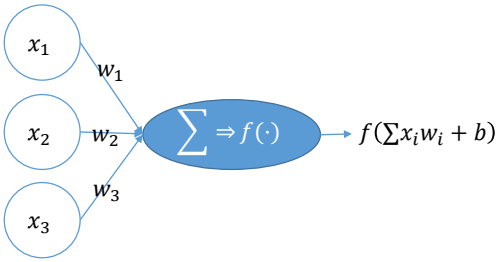
Figure 1. *Neural network hierarchy containing convolutional, pooling and classifier layers.*

- DNNs εξελίσσονται ραγδαία, γίνονται πιο σημαντικοί και **μεγάλοι**
  - Οι πρώτοι accelerators εστιάζανε στην επιτάχυνση των υπολογισμών
  - DianNao επικεντρώθηκε στις προκλήσεις της **memory**

16

Νευρωνικά Δίκτυα





- Τα ΝΔ εκπαιδεύονται για να προσδιοριστούν οι παράμετροι wi, b ώστε η συνάρτηση να έχει το μικρότερο δυνατό λάθος για ένα συγκεκριμένο σετ εισόδων

17

Επιταχυντές για DNN

- Απεικόνιση των «νευρώνων» σε υλικό
- Υπερ: απλότητα και επιδόσεις
- Κατά: μεγάλος χώρος
  - Η πολυπλεξία στο χρόνο γίνεται αλλά με μεγάλο κόστος
  - Μεγάλα DNN κοστίζουν πολύ...

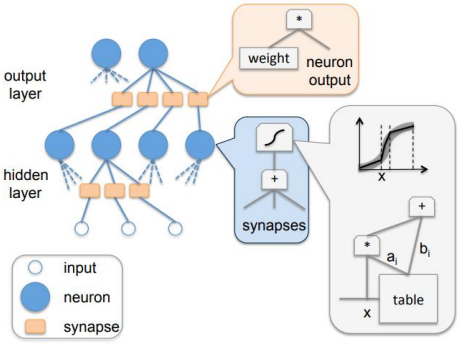


Figure 9. *Full hardware implementation of neural networks.*

18

Αρχιτεκτονική του DianNao

- Απλός μικρο-κώδικας για έλεγχο με πολύ εξειδικευμένες μικρο-εντολές
- Εξειδικευμένο datapath για multiply + add + sigmoid
- Εξειδικευμένες μνήμες scratchpads για inputs (NBin), outputs (NBout), and synapses (SB)
  - No associative lookups, no conflicts
  - Match line size to tile size for efficiency
  - DMA issued as needed to rotate values in/out

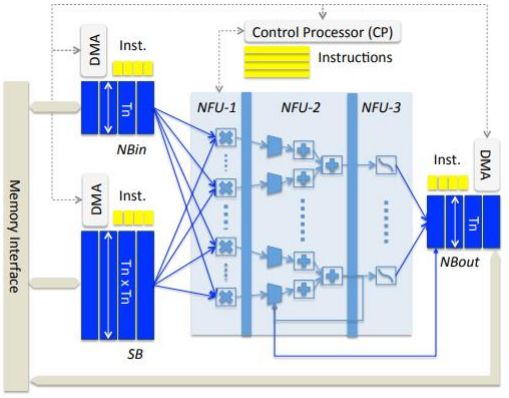


Figure 11. *Accelerator.*

DianNao Accelerator

- 110 × avg βελτίωση επιδόσεων vs SIMD
- 21 × avg βελτίωση σε ενέργεια vs SIMD
  - Much smaller improvement than other studies!
  - **Memory dominates energy**
- DaDianNao added large on-chip memories on multiple chips to improve energy by 150 ×

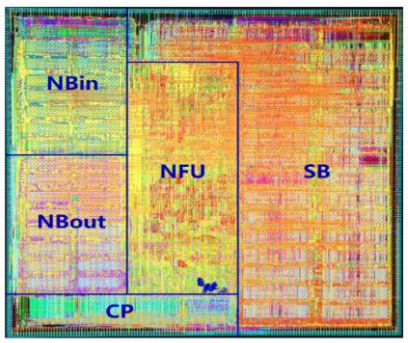
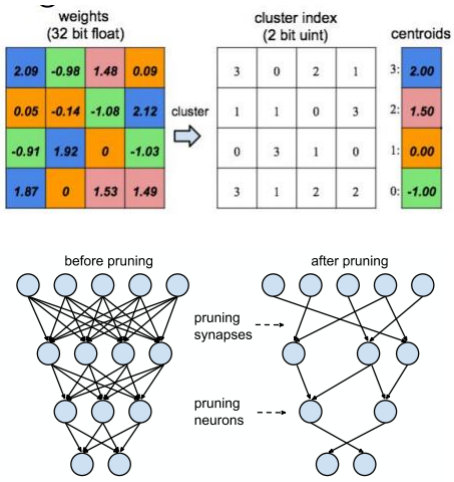


Figure 15. *Layout (65nm).*

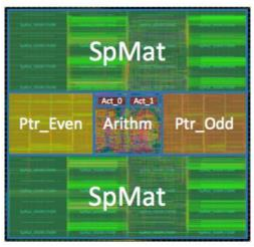
Sparse Neural Networks

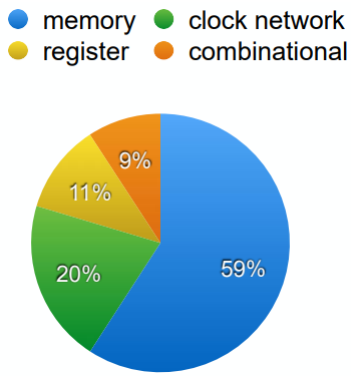
- [Han et al, ISCA'16]
- Παρατήρηση: Τα weights είναι συγκεντρωμένα μόνο σε μερικές τιμές
  - Χρήση μόνο **two bits** για αναπαράσταση των weight
  - Για χρήση μεγαλύτερης ακρίβειας (precision), τότε:
    ➔ use **codebook** to store $2^{2\ \text{bits}} = 4$ higher-precision values
- Παρατήρηση: τα περισσότερα weights είναι κοντά στο zero
  - **Prune** near-zero weights
    ➔ απατείται σημαντικά λιγότερη memory & compute !
  - αναπαράσταση NNs ως υπολογισμοί **sparse linear algebra**
    ➔ irregular control & memory references
- 10-49× μείωση στις απαιτήσεις μνήμης (memory footprint)



Αποτελέσματα

- 41mm^2 @ 45nm (much bigger than DianNao)
- Significantly less energy spent on memory accesses
- Claims $24{,}000\times$ improvement vs CPU and $3{,}400\times$ improvement vs GPU w/out compression





Χρειάζεται η Εξειδίκευση Αρχιτεκτονικής?

- Τα πέντε 'C' της εξειδίκευσης

1. Concurrency
2. Compute
3. Communication
4. Caching
5. Coordination / control

Το μεγαλύτερο ποσοστό βελτίωσης ≈ 100 × που προκύπτει από την εξειδίκευση δεν ισχύει αν χρησιμοποιηθεί ως βάση σύγκρισης μία βελτιστοποιημένη αρχιτεκτονική, προγραμματιζόμενη, που χρησιμοποιεί κάποιες από τις παραπάνω βελτιστοποιήσεις.

23

Πόση Εξειδίκευση Είναι Αρκετή ?

- …μόνο το ≈5% της ενέργειας καταναλώνεται από τα FUs ακόμα και σε απλούς πυρήνες! [Horowitz, ISSCC'14 Keynote]
- Case study: ενεργειακή αποδοτικότητα ενός H.264 encoder @ 720p [Hameed et al,ISCA'12]
  - General-purpose core – 1 ×
  - VLIW/Vector – 7 ×
  - Custom, "fused" FUs – 10 ×
  - "Magic" super-instructions – 180 ×
  - ASIC – 500 ×

Συμπέρασμα: *truly efficient designs will require application-specialized hardware."*

24

Αρχιτεκτονικές Αναδιατασσόμενες

- Μελλοντικές Τάσεις: "Agile" hardware και "Productive" HDLs
- Οι περιορισμοί σε αρχιτεκτονικές πολυ-πύρηνες δεν σημαίνει απαραίτητα άμεση ανάγκη για εξειδίκευση
  - Τα σημερινά SoCs είναι ήδη ετερογενή: CPUs + GPUs + DSPs
- Συστήματα σε chip με FPGA επανεμφανίζονται...
- Μία κατεύθυνση περιλαμβάνει **CGRAs** – coarse-grain reconfigurable arrays
  - Προγραμματιζόμενα όπως τα FPGAs, αλλά με hardened FUs / control / memories για αποδοτικότητα
  - Παραδείγματα: *Plasticine* [Prabhakar et al, ISCA'17] και *Stream-dataflow acceleration* [Nowatzki, ISCA'17]

25