

Εθνικό Μετσόβιο Πολυτεχνείο  
Σχολή Ηλεκτρολόγων Μηχ. και Μηχανικών Υπολογιστών

## ΣΥΣΤΗΜΑΤΑ ΠΑΡΑΛΛΗΛΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ

ΕΡΓΑΣΙΑ 2020 - Παρουσίαση

### Inference acceleration on FPGAs Focus on CNNs

Project Advisor: Παναγιώτης Μπάκος

#### Ομάδα 7

Αραπίδης Μανώλης  
Κουτρούμπας Αθανάσιος





# ΕΙΣΑΓΩΓΗ

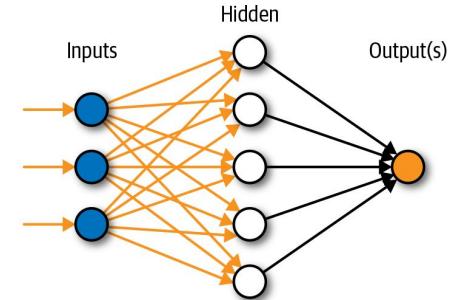
**Neural Networks**

# Neural Networks

---

## Ανάγκη Επιτάχυνσης

- Συνεχής και αυξανόμενη η χρήση των Νευρωνικών Δικτύων σε εφαρμογές Τεχνητής Νοημοσύνης.
- Τα CNN από τα πιο διαδεδομένα δίκτυα για εφαρμογές επεξεργασίας εικόνας, βίντεο και αναγνώριση φωνής.
- Έχουν μεγάλο υπολογιστικό κόστος.
- Υπάρχει ανάγκη για επιτάχυνση αυτών των εφαρμογών σε εξειδικευμένο hardware.
  - ◆ CPU 10-100 GOP/s
  - ◆ GPU up to 10 TOP/s
  - ◆ TPU - ASIC up to 92 TOP/s
- Τα τελευταία χρόνια επεκτείνεται και ερευνάται η χρήση των FPGAs για την επιτάχυνση αυτών των εφαρμογών.



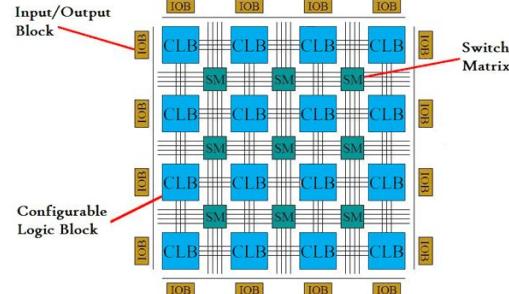


# ΕΙΣΑΓΩΓΗ

FPGA

# FPGA

## Τι είναι ένα FPGA;



- Τα FPGA (Field-Programmable Gate Array) είναι ολοκληρωμένα κυκλώματα, τα οποία μπορούν να προγραμματιστούν στο επίπεδο του υλικού από έναν σχεδιαστή, μετά την κατασκευή τους στο εργοστάσιο.
- Το FPGA αποτελείται από έναν πίνακα προγραμματιζόμενων λογικών μονάδων (Configurable Logic Block - **CLB**) που συνδέονται μεταξύ τους, οι οποίες υλοποιούν διάφορες λογικές συναρτήσεις.
- Αυτό επιτρέπει να κατασκευαστούν κυκλώματα υψηλής παραλληλίας, το οποία επιτυγχάνει επιδόσεις πολύ μεγαλύτερες από αυτές μιας κοινής επεξεργαστικής μονάδας.
- Προγραμματίζονται συνήθως από γλώσσες περιγραφής υλικού (**HDL**):
  - ◆ VHDL
  - ◆ Verilog
- Μπορούν και να προγραμματιστούν από γλώσσες υψηλού επιπέδου με High Level Synthesis (**HLS**):
  - ◆ C/C++
  - ◆ SystemC
  - ◆ MATLAB



# **ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ**

**Inference vs. Training**

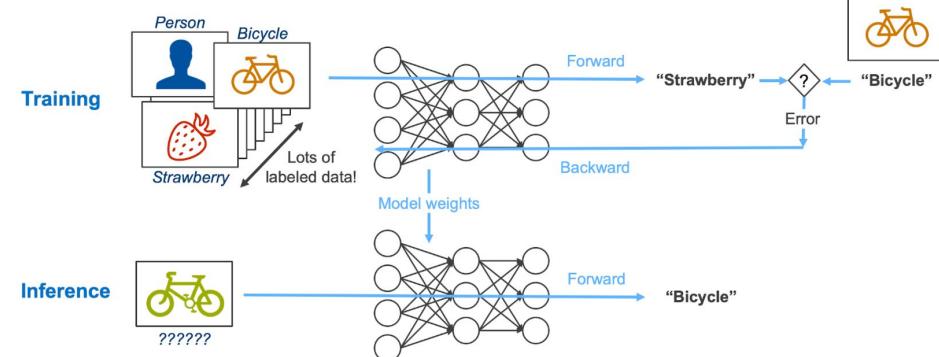
# Inference vs. Training

## Training Phase (Φάση Εκπαίδευσης)

- Έχουμε ένα σύνολο από δεδομένα που έχουμε κατηγοριοποιήσει και δίνουμε ως είσοδο.
- Δύο στάδια κατά το training:

### Forward Propagation

- Η είσοδος περνάει από τα διαδοχικά layers του δικτύου
- Εν τέλει παράγει μία έξοδο.

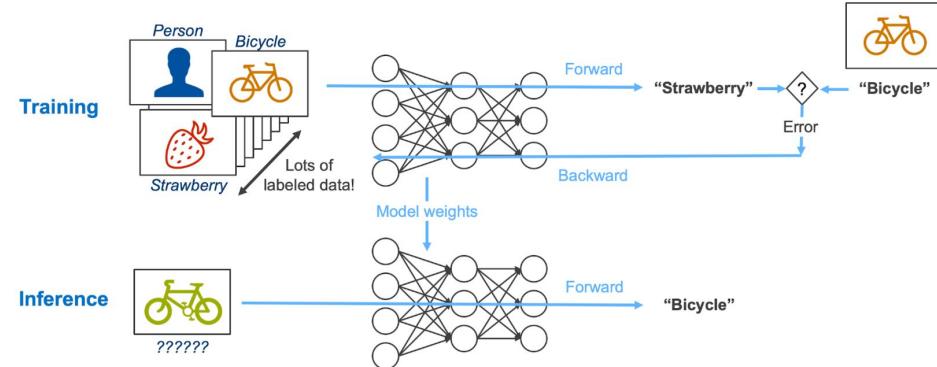


### Backward Propagation

- Η πρόβλεψη που δίνεται ως έξοδος συγκρίνεται με αυτή που είναι γνωστή.
- Η απόκλιση τους μεταδίδεται προς τα πίσω, διαδοχικά μεταξύ των layers.
- Χρησιμοποιείται για να ανανεωθούν τα βάρη στους νευρώνες του κάθε layer.

- Τα παραπάνω επαναλαμβάνονται διαδοχικά μέχρι να φτάσουμε στην επιθυμητή απόκλιση ή
- Να σταματήσουμε να παρατηρούμε βελτίωση στο accuracy που πετυχαίνουμε.

# Inference vs. Training



## Inference Phase (Φάση Εξαγωγής Συμπερασμάτων)

- Χρησιμοποιεί ένα ήδη εκπαιδευμένο νευρωνικό δίκτυο.
- Εκτελεί μόνο **Forward Propagation**.
- Δέχεται άγνωστες εισόδους και επιχειρεί να τις κατηγοριοποιήσει.

# Inference vs. Training

---

## Διαφορές

### Training

- Training γίνεται συνήθως μόνο μία φορά.
- To Training είναι πιο απαιτητική διαδικασία και μπορεί να διαρκέσει από ώρες μέχρι και μέρες.

### Inference

- Inference επαναλαμβάνεται διαρκώς.
- Αντίθετα το inference, ειδικά για real-life applications, είναι κρίσιμο να διαρκεί όσο το δυνατόν λιγότερο.



# **ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ**

## **GPU vs. FPGA**

# Πλεονεκτήματα FPGA

---

- Τα GPU χρησιμοποιούνται ευρέως για τα νευρωνικά και ειδικότερα για Training.
- Όμως πλέον η βελτίωση της επίδοσης των FPGA, τα κάνει όλο και πιο δημοφιλή για το Inference.

## Υπολογιστική Δύναμη

- Υψηλή δυνατότητα παραλληλίας.
- Μεγάλο πλήθος on-chip memory.
- Απευθείας υλοποίηση σε υλικό των datatypes (π.χ. Binary - 1 bit, 4 bit, Custom).

## Κατανάλωση Ενέργειας

- FPGA περισσότερο energy efficient.
  - ◆ Το Xilinx Virtex Ultrascale+ έχει κατά 4 φορές μικρότερη κατανάλωση από την Nvidia Tesla V100.

## Ευελιξία Αρχιτεκτονικής

- Το hardware προγραμματίζεται στις ανάγκες του νευρωνικού δικτύου.
  - Οι υπολογιστικές μονάδες και το datapath είναι πλήρως προγραμματίσιμο.
    - ◆ Το datapath ακόμα και κατά το runtime.
- 
- Ο προγραμματισμός των FPGA είναι απαιτητικός, ειδικά σε γλώσσες περιγραφής υλικού (**HDL**).
    - Όμως υπάρχουν HLS εργαλεία, όπως τα **Vivado HLS**, **Intel HLS Compiler** και **LegUp** που απλοποιούν αρκέτα.



# ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

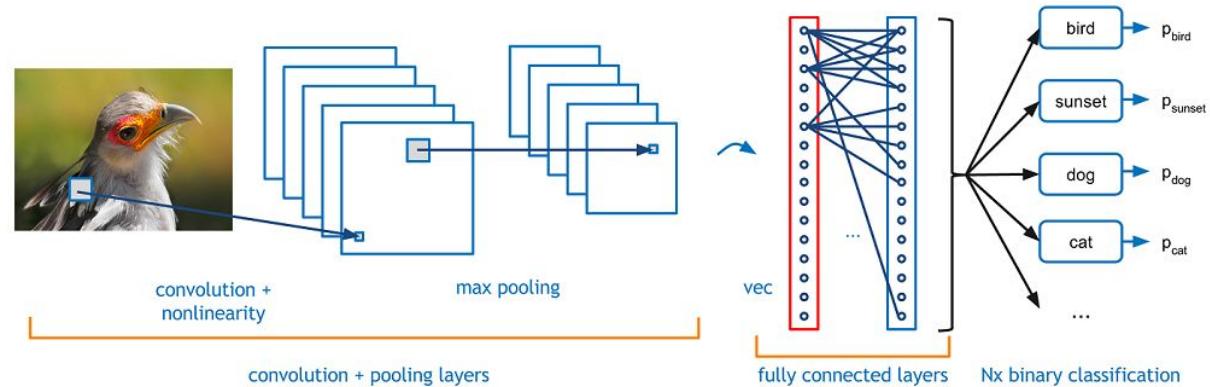
**Convolutional Neural Networks - CNN**

# Convolutional Neural Networks - CNN

## Περιγραφή

Τα CNN αποτελούνται από ένα ή παραπάνω από τα παρακάτω Layers:

- Convolutional layers
- Activation Layers
- Pooling layers
- Batch-Normalization Layers
- Fully Connected Layers

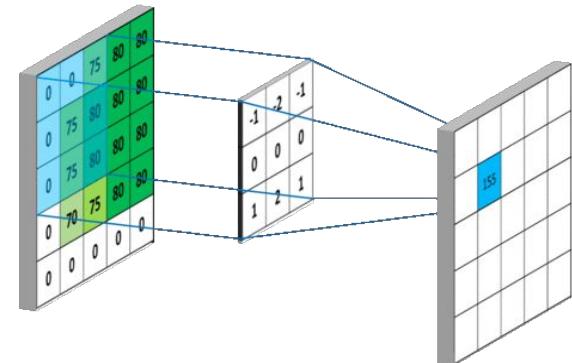


# Convolutional Neural Networks - CNN

## Convolution στα CNN - 2D Convolution

2D συνέλιξη:

- Element-wise multiplication τιμών φίλτρου (**kernel**) με τιμές **Input Feature Map (FM)** στις οποίες εφαρμόζεται το φίλτρο
- Άθροισμα γινομένων → Ένα κελί του **Output FM**
- Φίλτρο ολισθαίνει στο **Input FM** → Υπολογισμός όλων των κελιών **Output FM**

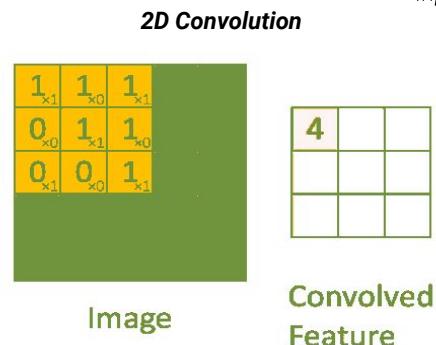


**2D Convolution**

2D Convolution				
Input Feature Map (FM)		Kernel	Output FM	
0	0	0	0	0
0	10	20	30	0
0	40	50	60	0
0	70	80	90	0
0	0	0	0	0

*		=	
1 2 1	2 4 2	1 2 1	630 1080 990
2 4 2	1 2 1	1 2 1	1560 2400 2040
1 2 1	1 2 1	1 2 1	1710 2520 2070



Input Feature Map (FM)      Kernel      Output FM

**2D Convolution**

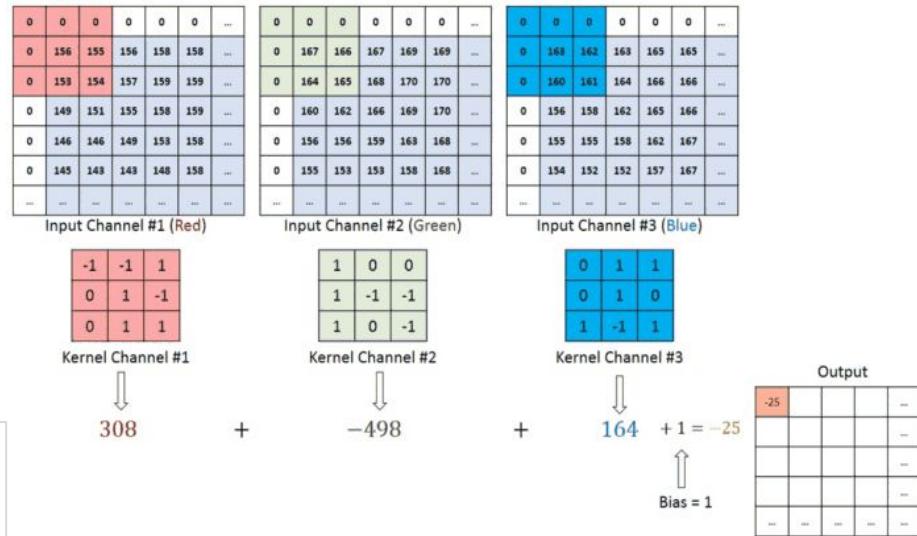
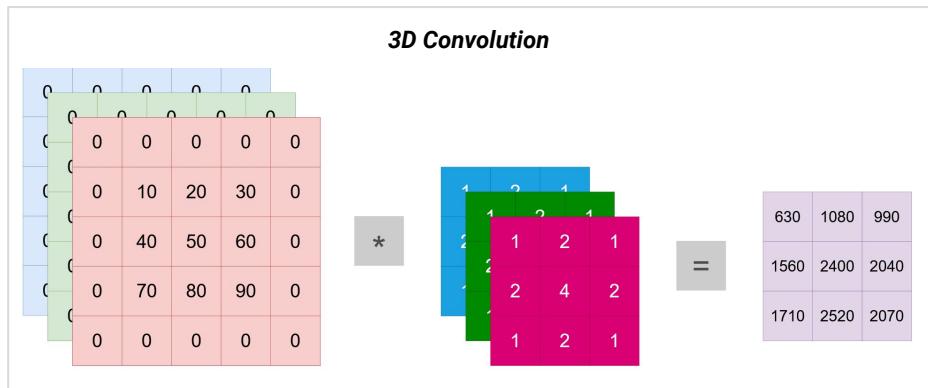
Input Feature Map (FM)      Kernel      Output FM

# Convolutional Neural Networks - CNN

## Convolution στα CNN - 3D Convolution

3D συνέλιξη:

- Κανάλια Input FM
- Κανάλια Output FM → Άθροισμα κελίων αποτελεσμάτων
- Φίλτρο ολισθαίνει στο Input FM → Υπολογισμός όλων των κελιών Output FM

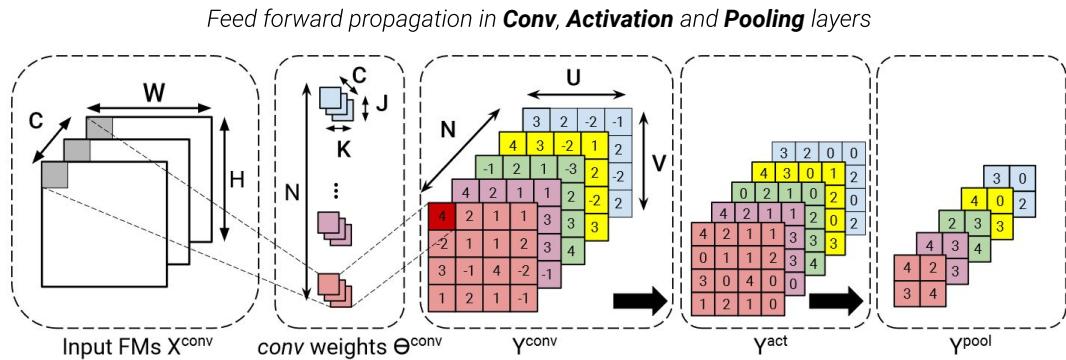


# Convolutional Neural Networks - CNN

## Inference στα CNN

### Inference Εικόνας στο δίκτυο:

- Εισαγωγή **Input FM** για επεξεργασία.
- **N** 3D συνελίξεις βάθους **C**, όσα και τα φίλτρα.
- Παραγωγή **N** Output FM.
- Οι τιμές των **N** Output FM, περνάνε από την Activation Function.
- Τα **N** Output FM, περνάνε από το Pooling Layer.



### Παράμετροι-Ορολογία:

- FM:** Feature Map  
**B:** Batch Size  
**C:** Βάθος Input FM - Φίλτρων  
**N:** Αριθμός φίλτρων  
**C,W,H:** Διαστάσεις Input FM  
**C,K,J:** Διαστάσεις φίλτρων  
**N,U,V:** Διαστάσεις Output FM

# Convolutional Neural Networks - CNN

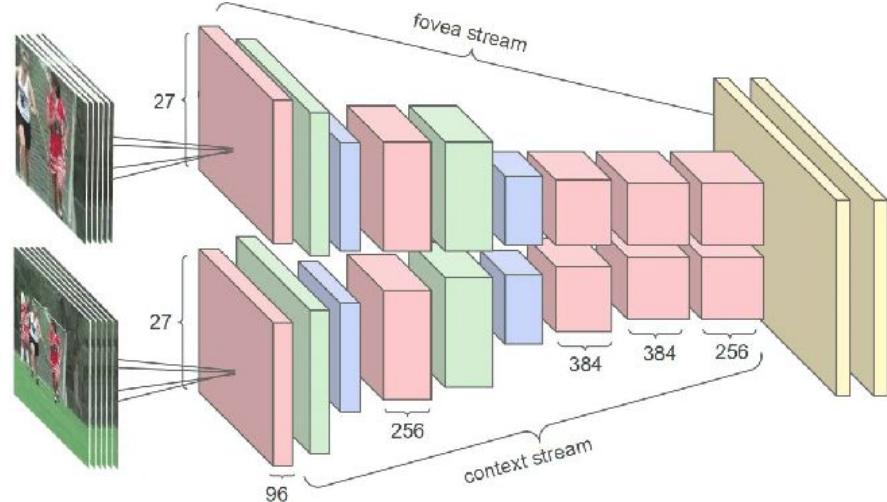
## Παραλληλίες στα CNN

### → Batch Parallelism

- ◆ Παράλληλη ταξινόμηση **B** εικόνων ταυτόχρονα.
- ◆ Επαναχρησιμοποίηση φίλτρων σε κάθε Layer.
- ◆ Ελαχιστοποίηση Accesses στην εξωτερική μνήμη.

### → Inter-Layer Parallelism

- ◆ Εξάρτηση μεταξύ δεδομένων μεταξύ των layers
- ◆ Επεξεργασία των layers γίνεται **Pipelined**.
- ◆ Η επεξεργασία του layer  $\ell$  ξεκινάει πριν τελειώσει η επεξεργασία του layer  $\ell-1$ .



### Παράμετροι-Ορολογία:

**FM:** Feature Map

**B:** Batch Size

**C:** Βάθος Input FM - Φίλτρων

**N:** Αριθμός φίλτρων

# Convolutional Neural Networks - CNN

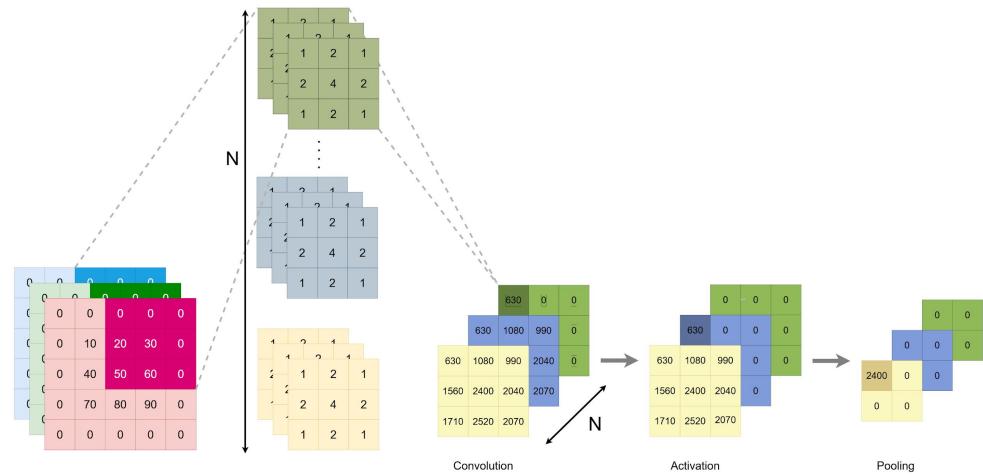
## Παραλληλίες στα CNN

### → Batch Parallelism

- ◆ Παράλληλη ταξινόμηση **B** εικόνων ταυτόχρονα.
- ◆ Επαναχρησιμοποίηση φίλτρων σε κάθε Layer.
- ◆ Ελαχιστοποίηση Accesses στην εξωτερική μνήμη.

### → Inter-Layer Parallelism

- ◆ Εξάρτηση μεταξύ δεδομένων μεταξύ των layers
- ◆ Επεξεργασία των layers γίνεται **Pipelined**.
- ◆ Η επεξεργασία του layer  $\ell$  ξεκινάει πριν τελειώσει η επεξεργασία του layer  $\ell-1$ .



### Παράμετροι-Ορολογία:

**FM:** Feature Map

**B:** Batch Size

**C:** Βάθος Input FM - Φίλτρων

**N:** Αριθμός φίλτρων

Inter-Layer Parallelism

# Convolutional Neural Networks - CNN

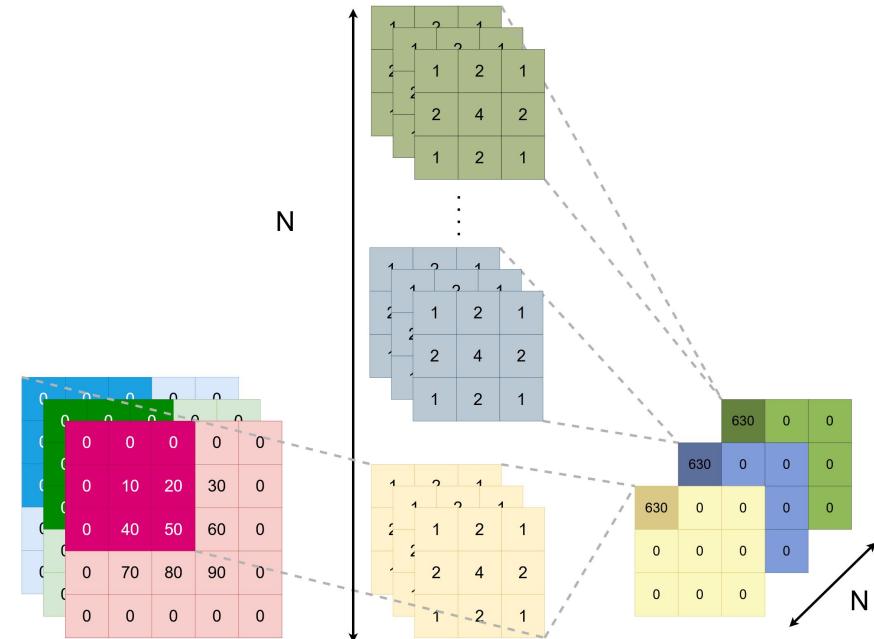
## Παραλληλίες στα CNN

### → Inter-FM Parallelism

- ◆ Κάθε Output FM από το Conv layer υπολογίζεται ανεξάρτητα από τα υπόλοιπα.
- ◆ Για  $N$  φίλτρα, έχουμε παράλληλη επεξεργασία  $N$  FM.

### → Intra-FM Parallelism

- ◆ Σε κάθε Output FM ο υπολογισμός του κάθε pixel, μπορεί να γίνει ταυτόχρονα με τα υπόλοιπα.
- ◆ Για διαστάσεις  $V \times U$  Output FM, έχουμε  $V \times U$  στοιχεία για παράλληλη επεξεργασία.



Inter-FM Parallelism

### Παράμετροι-Ορολογία:

**FM:** Feature Map

**B:** Batch Size

**C:** Βάθος Input FM - Φίλτρων

**N:** Αριθμός φίλτρων

**C,W,H:** Διαστάσεις Input FM

**C,K,J:** Διαστάσεις φίλτρων

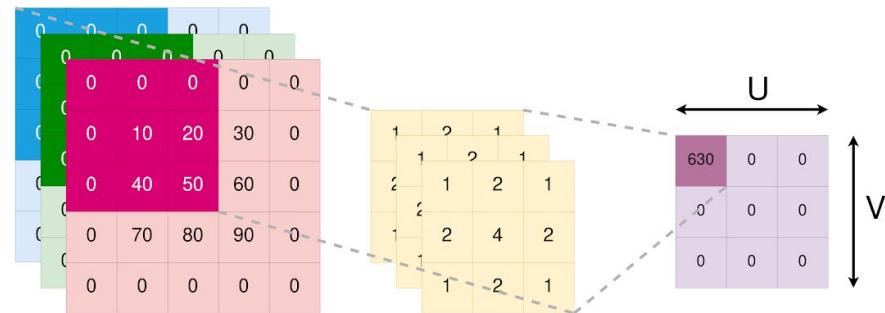
**N,U,V:** Διαστάσεις Output FM

# Convolutional Neural Networks - CNN

## Παραλληλίες στα CNN

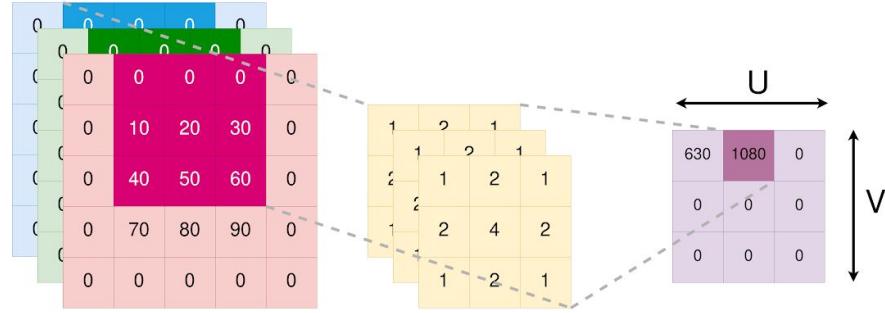
### → Inter-FM Parallelism

- ◆ Κάθε Output FM από το Conv layer υπολογίζεται ανεξάρτητα από τα υπόλοιπα.
- ◆ Για  $N$  φίλτρα, έχουμε παράλληλη επεξεργασία  $N$  FM.



### → Intra-FM Parallelism

- ◆ Σε κάθε Output FM ο υπολογισμός του κάθε pixel, μπορεί να γίνει ταυτόχρονα με τα υπόλοιπα.
- ◆ Για διαστάσεις  $V \times U$  Output FM, έχουμε  $V \times U$  στοιχεία για παράλληλη επεξεργασία.



### Παράμετροι-Ορολογία:

**FM:** Feature Map

**B:** Batch Size

**C:** Βάθος Input FM - Φίλτρων

**N:** Αριθμός φίλτρων

**C,W,H:** Διαστάσεις Input FM

**C,K,J:** Διαστάσεις φίλτρων

**N,U,V:** Διαστάσεις Output FM

Intra-FM Parallelism

# Convolutional Neural Networks - CNN

## Παραλληλίες στα CNN

### → Inter-Convolution Parallelism

- ◆ Σε κάθε Conv layer, γίνεται 3D συνελιξη **C** Input FM με **C** Φίλτρα.
- ◆ Οι 3D συνελίξεις εκφράζονται ως άθροισμα **C** 2D συνελίξεων του κάθε Input FM με το αντίστοιχο φίλτρο.
- ◆ Άρα παράλληλη επεξεργασία **C** 2D συνελίξεων.

### → Intra-Convolution Parallelism

- ◆ Η κάθε 2D συνέλιξη που συμβαίνει σε κάθε Conv layer, μπορεί να γίνει Pipelined.
- ◆ Αποτελείται από MAC operations.

#### Παράμετροι-Ορολογία:

**FM**: Feature Map

**B**: Batch Size

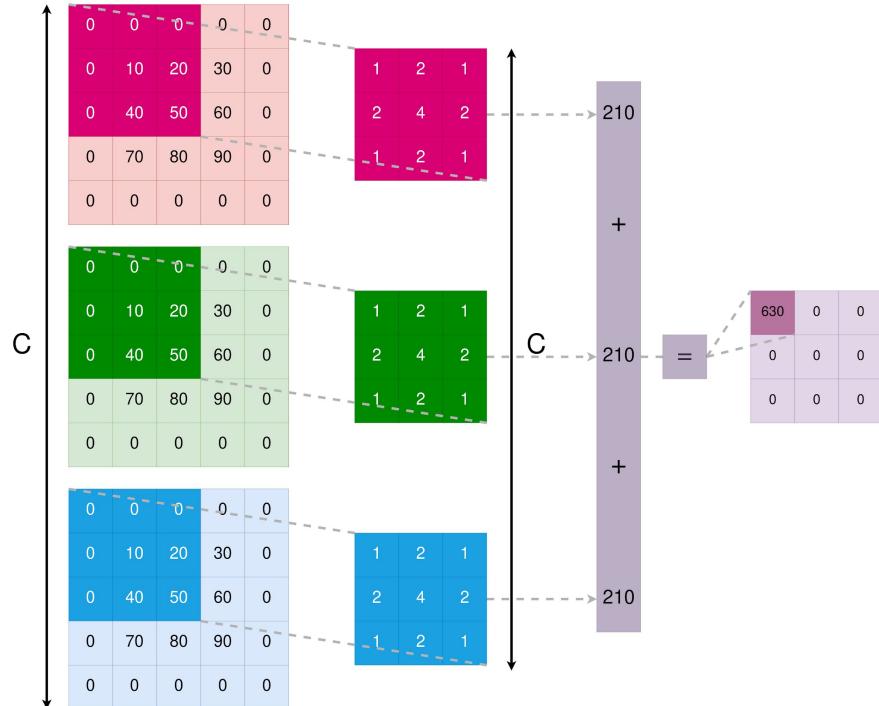
**C**: Βάθος Input FM - Φίλτρων

**N**: Αριθμός φίλτρων

**C,W,H**: Διαστάσεις Input FM

**C,K,J**: Διαστάσεις φίλτρων

**N,U,V**: Διαστάσεις Output FM



Inter-Convolution Parallelism



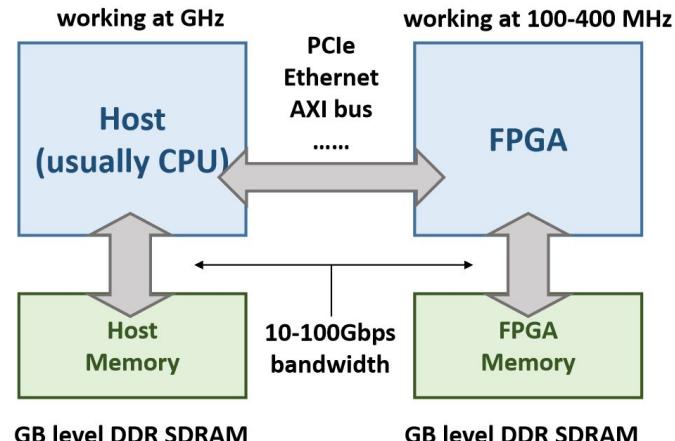
# FPGA

## Accelerating CNN Inference on FPGAs

# Accelerating CNN Inference on FPGAs

## Αρχιτεκτονική FPGA Επιταχυντή

- CPU Host
  - ◆ Συχνότητα Λειτουργίας ~ GHz
  - ◆ Ελέγχει και συντονίζει μέσω software FPGA
- FPGA Part
  - ◆ Συχνότητα Λειτουργίας ~100 MHz
  - ◆ Υλοποιεί τον CNN Accelerator
  - ◆ On-chip Μνήμη → SRAM (τάξης MB), Registers
- Επικοινωνία CPU - FPGA
  - ◆ PCIe Bus
  - ◆ AXI Bus
  - ◆ Ethernet
  - ◆ SoC
- Host και FPGA δικές τους εξωτερικές μνήμες
  - ◆ Off-chip Μνήμη → DDR (τάξης GB)
- Δυνατότητα πρόσβασης στη μνήμη του άλλου



Τυπική αρχιτεκτονική ενός FPGA-based CNN Accelerator

# Accelerating CNN Inference on FPGAs

## Περιορισμοί FPGA

- FPGA προσφέρονται για το Inference των CNN, λόγο των παραλληλιών τους.
- Όμως, υπάρχουν αρκετά προβλήματα.

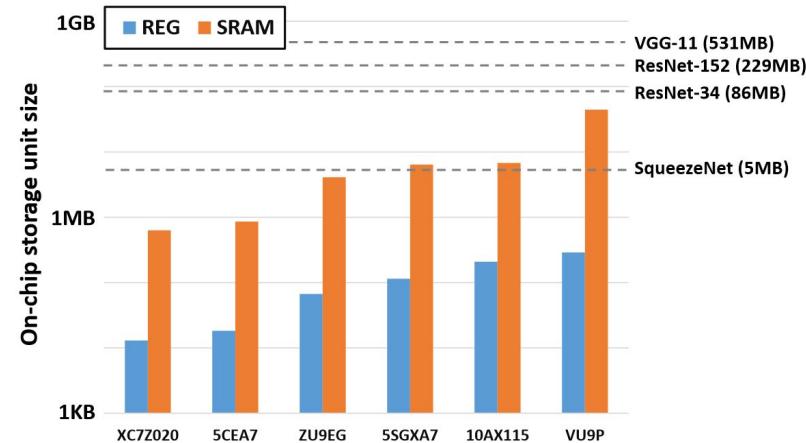
### Πρόβλημα (Περιορισμένοι Πόροι):

- CNN έχουν βαθιές τοπολογίες με πολλαπλά Layers.
- FPGA έχουν περιορισμένους πόρους για επεξεργασία.
  - ◆ Π.χ. Κάποιες φορές δεν μπορεί να γίνει πλήρως "Unrolled" ούτε ένα μόνο Conv layer.

### Πρόβλημα (Περιορισμένη on-chip Μνήμη):

- CNN απαιτούν μεγάλη μνήμη (π.χ. 100-500 MB).
- FPGA έχουν περιορισμένους on-chip μνήμη (π.χ. 50 MB).
- Οπότε χρειάζονται off-chip μνήμες.
  - ◆ Περιορίζουν επιδόσεις
  - ◆ Υψηλότερη ενεργειακή κατανάλωση

Η απόσταση που υπάρχει μεταξύ των μεγεθών των CNN μοντέλων και της on-chip μνήμης (Registers, SRAM) για διάφορα μοντέλα FPGA.



Η διακεκομένη γραμμή δείχνει το μέγεθος των παραμέτρων για διάφορα CNN μοντέλα με 32-bit floating point παραμέτρους.

# Accelerating CNN Inference on FPGAs

---

## Μέθοδοι Επιτάχυνσης

Για να αντιμετωπιστούν τα προβλήματα γίνονται βελτιστοποιήσεις στους παρακάτω τομείς:

- Αρχιτεκτονικές FPGA
  - ◆ Systolic Arrays
  - ◆ SIMD Accelerators
  - ◆ Loop Optimizations
  - ◆ CLB vs. DSP Designs
- Approximate Computing
  - ◆ Fixed Point Arithmetic (Fixed - Dynamic - Binary)
  - ◆ Weight Pruning
  - ◆ Low Rank Approximation
- Algorithmic Optimizations
  - ◆ GEMM Transformation
  - ◆ Winograd Transform
  - ◆ Fast Fourier Transform (FFT)



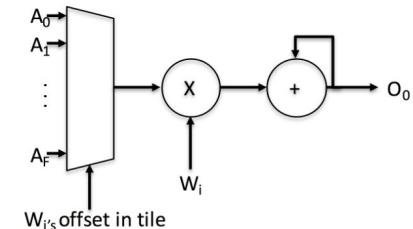
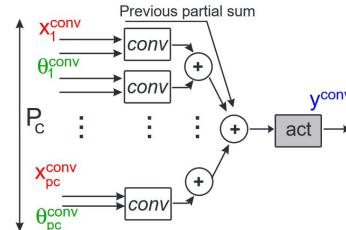
# FPGA

## FPGA Architectures - Optimizations

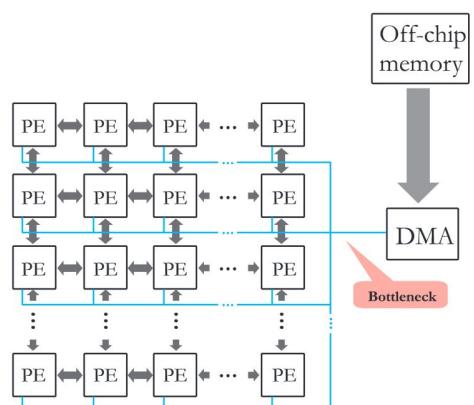
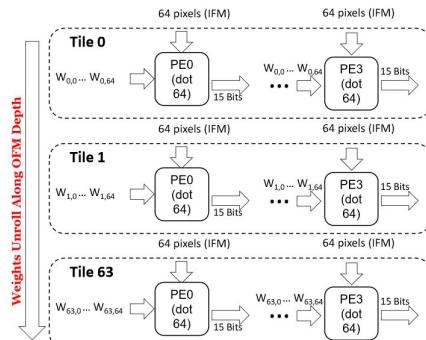
# FPGA Architectures - Optimizations

## Processing Elements (PE)

- Βασικές μονάδες επεξεργασίας:
  - ◆ Εκτελούν τις ίδιες πράξεις
  - ◆ Επαναχρησιμοποιούνται συνεχώς
  
- Σκοπός τους είναι:
  - ◆ Όσο το δυνατόν μικρότερα.
  - ◆ Να χρησιμοποιηθούν όσο το δυνατόν περισσότερα από αυτά.
  - ◆ Ωστε να επιτευχθεί High Peak Performance.
  - ◆ Και μεγαλύτερες συχνότητες.



Παραδείγματα δομής ενός PE

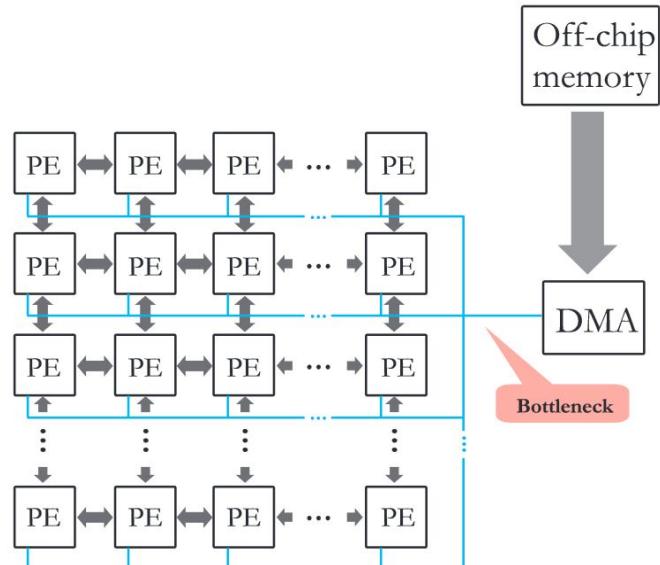


Παραδείγματα αρχιτεκτονικών που κάνουν χρήση των PE

# FPGA Architectures - Optimizations

## Systolic Arrays

- Από τις αρχικές αρχιτεκτονικές στα FPGA
  - ◆ Επιτάχυνση 2D συνελίξεων στα Conv Layers
- Αρχιτεκτονική:
  - ◆ Τοποθετούνται PE σε 2-διάστατη τοπολογία
  - ◆ Το κάθε PE ενώνεται με τα γειτονικά του
  - ◆ Δεδομένα περνάνε από το ένα PE στο άλλο
- Προβλήματα:
  - ◆ Υποστηρίζουν συνελίξεις με φίλτρα μέχρι ενός μεγέθους (π.χ.  $K = 7$ )
  - ◆ Δεν υλοποιούνται τεχνικές caching των δεδομένων
  - ◆ Παίρνουν τις εισόδους τους από κάποια off-chip μνήμη
  - ◆ Η απόδοση τους είναι αρκετά Memory Bound

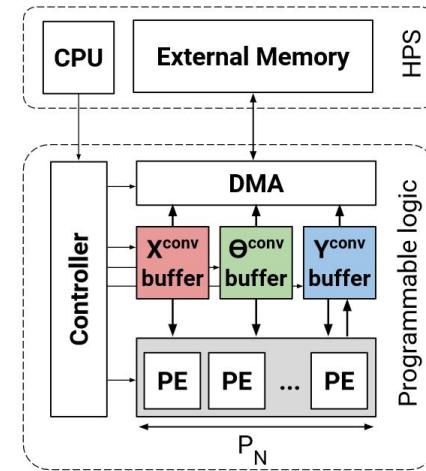


Παράδειγμα τοπολογίας PE σε Systolic Array

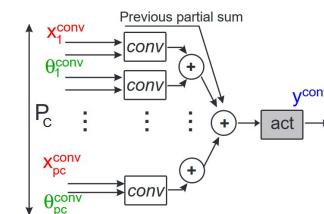
# FPGA Architectures - Optimizations

## SIMD Accelerators

1. **Input FMs (Xconv)** και **Weights (Θconv)** γίνονται fetched από DRAM στους on-chip buffers.
2. Τα δεδομένα περνάνε μετά στα PE.
3. Μετά την επεξεργασία τους στα PE, τα **αποτελέσματα (Yconv)** μεταφέρονται πίσω στους on-chip buffers.
4. Αν χρειαστεί μεταφέρονται στην εξωτερική μνήμη
  - ◆ Ήστε να χρησιμοποιηθούν από τα επόμενα layers.



Αρχιτεκτονική ενός γενικού SIMD επιταχυντή

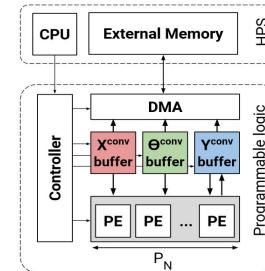


Δομή ενός PE

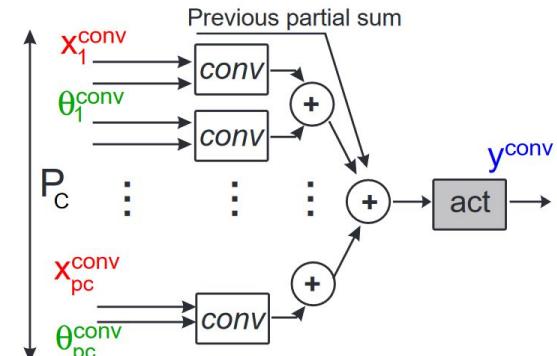
# FPGA Architectures - Optimizations

## SIMD Accelerators

- Το κάθε PE υλοποιεί:
  - ◆ Τις πράξεις για την 3D συνέλιξη ενός Input FM με ένα φίλτρο σε ένα Conv Layer
    - 2D Convolution
    - Άθροισμα 2D αποτελεσμάτων
    - Activation
  - ◆ Για τις  $N$  3D συνελίξεις με τα  $N$  φίλτρα που απαιτούνται σε ένα Conv Layer χρειάζονται  $N$  PE.
- Το να υλοποιηθεί ο επιταχυντής για ένα CNN δίκτυο είναι θέμα:
  - ◆ Εύρεσης της **βέλτιστης διαμόρφωσης των PE**
  - ◆ Βέλτιστου χρονοπρογραμματισμού της ροής των δεδομένων



Αρχιτεκτονική ενός γενικού SIMD επιταχυντή



Δομή ενός PE

# FPGA Architectures - Optimizations

---

## Loop Optimizations

- Η εύρεση του **βέλτιστου PE design**, μπορεί να αναχθεί σε ένα **Loop Optimization Problem**.
- Τεχνικές βελτιστοποίησεις των εμφωλευμένων loop:
  - ◆ Loop Unrolling              (Unroll Factor **P<sub>i</sub>**)
  - ◆ Loop Tiling                (Tiling Factor **T<sub>i</sub>**)

Table 4: Loop Optimization Parameters  $P_i$  and  $T_i$ 

Parallelism	Intra-layer	Inter-FM	Intra-FM		Inter-Convolution	Intra-Convolution	
Loop	$L_L$	$L_N$	$L_V$	$L_U$	$L_C$	$L_J$	$L_K$
Unroll factor	$P_L$	$P_N$	$P_V$	$P_U$	$P_C$	$P_J$	$P_K$
Tiling Factor	$T_L$	$T_N$	$T_U$	$T_U$	$T_C$	$T_J$	$T_K$

# FPGA Architectures - Optimizations

---

## Loop Optimizations - Loop Unroll

- Μπορεί να γίνει Loop Unrolling με factor  $P_i$  για οποιοδήποτε από τους παραλληλισμούς
 
$$P_i \leq i, i \in \{L, V, U, N, C, J, K\}$$
- To Unrolling Factor  $P_N$  καθορίζει τον αριθμό των PE που χρειαζόμαστε.
- Ta Unrolling Factor  $P_c, P_k, P_j$  καθορίζουν των αριθμό των:
  - ◆ Multipliers
  - ◆ Adders
  - ◆ Μέγεθος Registers σε κάθε PE.

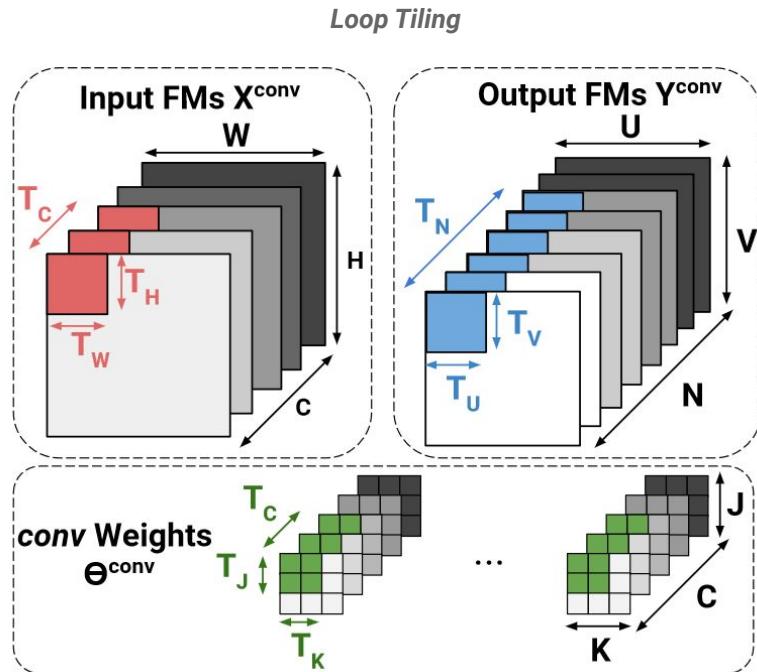
Table 4: Loop Optimization Parameters  $P_i$  and  $T_i$

Parallelism	Intra-layer	Inter-FM	Intra-FM		Inter-Convolution	Intra-Convolution	
Loop	$L_L$	$L_N$	$L_V$	$L_U$	$L_C$	$L_J$	$L_K$
Unroll factor	$P_L$	$P_N$	$P_V$	$P_U$	$P_C$	$P_J$	$P_K$
Tiling Factor	$T_L$	$T_N$	$T_U$	$T_U$	$T_C$	$T_J$	$T_K$

# FPGA Architectures - Optimizations

## Loop Optimizations - Loop Tiling

- Θέλουμε δεδομένα να αποθηκεύονται ιδανικά μόνο στους on-chip buffers και τους registers
- FMs, Βάρη χωρίζονται σε **Tiles**.
- Το μέγεθος του Tile, εξαρτάται:
  - ◆ Από το μέγεθος της on-chip μνήμης
  - ◆ Καθορίζει memory accesses
- Ένα **μεγάλο Tile**:
  - ◆ Απαιτεί μεγαλύτερη on-chip μνήμη
  - ◆ Ελαχιστοποιεί τα Memory Accesses από DRAM
- Γειτονικά Tile μοιράζονται μέρος των δεδομένων:
  - ◆ Input FM
  - ◆ Βάρη



$$\mathcal{M}_{\text{conv}} = T_C T_H T_W + T_N T_C T_J T_K + T_N T_V T_U$$



# FPGA

## Approximate Computing

- Κβάντιση Δεδομένων
- Μείωση Πράξεων



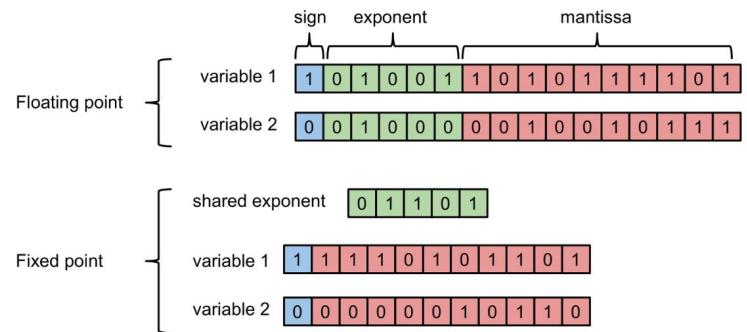
# Approximate Computing

- Κβάντιση Δεδομένων

# Approximate Computing

## Fixed Point Arithmetics - Static Fixed Point

- Χρήση Fixed Point αντί Floating Point (32 bit)
  - ◆ Π.χ. Αναπαράσταση με Custom Data Types
    - 16 bit, 8 bit, 4 bit κλπ
- **Static Fixed Point**
  - ◆ Διαμοιραζόμενο exponent για όλο το νευρωνικό.
  - ◆ Εύρος των αριθμών και της ακρίβειας είναι προκαθορισμένο.
  - ◆ Μεγαλύτερο throughput και λιγότερη κατανάλωση.
- Π.χ. Σε ενα DSP μπορούν να γίνουν:
  - ◆ 1 32x32 bit πολλαπλασιασμος με FLP
  - ◆ 2 18x19bit πολλαπλασιασμοί με SFP



Σύγκριση Αναπαραστάσεων:  
**Floating Point vs. Fixed Point**

# Approximate Computing

---

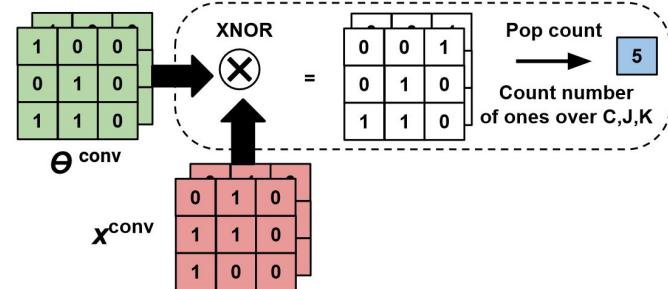
## Fixed Point Arithmetics - Dynamic Fixed Point

- Τα FMs και τα βάρη απαιτούν διαφορετικό εύρος αριθμών και ακρίβειας.
- **Dynamic Fixed Point:**
  - ◆ Διαφορετικό exponent για FMs και βάρη ανά επίπεδο.
- Τα δεδομένα έχουν την αναπαράσταση που χρειάζονται.
- Μεγαλύτερη ευελιξία και precision, με υψηλό throughput.
- Brute Force ή Ευριστική αναζήτηση των shared exponent.

# Approximate Computing

## Fixed Point Arithmetics - Binary Representation

- **Binary Representation:**
  - ◆ Δυαδική αναπαράσταση των βαρών ή των FMs ή και των δύο.
- Trade off επίδοσης με accuracy.
- ImageNet:
  - ◆ Μόνο binary βάρη
    - 3200% Απαιτήσεις bandwidth ↓
    - 19.2% accuracy ↓
  - ◆ Binary FMs και βάρου
    - 29.8% accuracy ↓
- XNOR ακολουθούμενο από pop-counter.
- Συνέλιξη πάνω στο προγραμματιζόμενο υλικό και όχι στα DSPs.
  - ◆ Γενικά όσο μειώνεται το precision μειώνεται η χρήση των DSPs.
  - ◆ Energy Efficient και καλύτερη χρήση hardware.



**Binary Neural Networks:** Συνέλιξη δυαδικού πίνακα με δυαδικό φίλτρο με χρήση XNOR operator και pop counter

# Approximate Computing

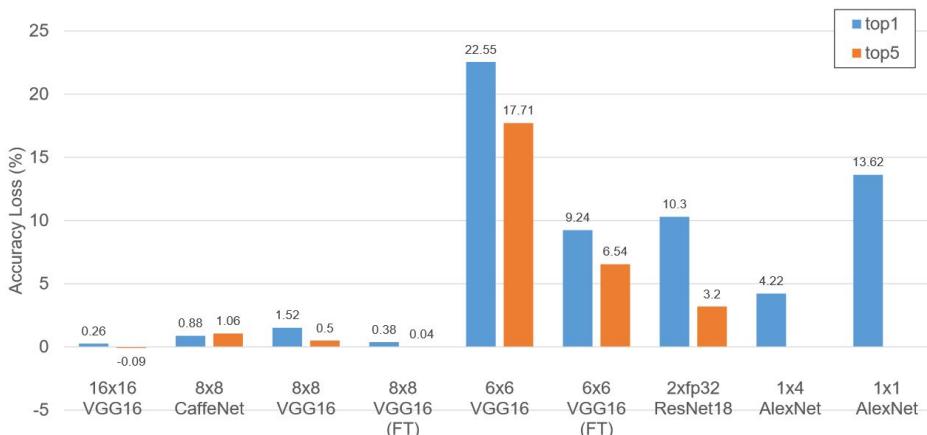
## Fixed Point Arithmetics - Ternary Representation

### → Ternary Representation:

- ◆ Τριαδική αναπαράσταση των βάρων (-1, 0, +1).
- ◆ Συνδυάζεται με μεγαλύτερη αναπαράσταση για FMs.
- ◆ Τα MAC operations αντικαθίστανται από αθροίσεις.

### → Για INT-8-2 :

- ◆ 8 bit για activation και 2-bit για βάρον.
- ◆ + Σχεδόν ίδιες επιδόσεις με Binary.
- ◆ + Καλύτερο Accuracy από Binary.
- ◆ **76 TOPs** που προσεγγίζει τις δυνατότητες των TPU (**96 TOPs**).



Συγκριτικό διάγραμμα μεταξύ διαφορετικών τρόπων κράντισης των δεδομένων.

To κάθε configuration συμβολίζει **(weight bit-width)x(activation bit-width)**



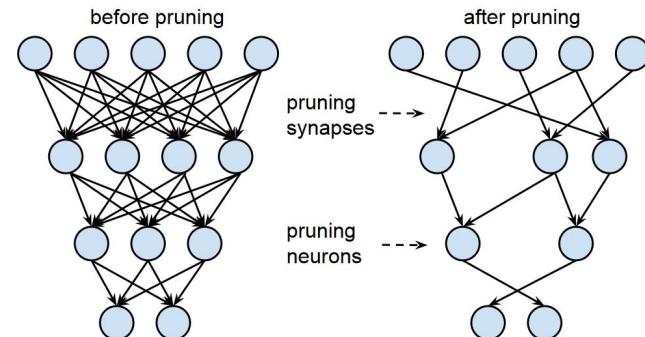
# Approximate Computing

- Μείωση Πράξεων

# Approximate Computing

## Reduce Computations - Weight Pruning

- Τα CNNs έχουν πολλές παραμέτρους και κάποιες μπορεί να αφαιρεθούν
- **1η μέθοδος:** Βάρη πολύ κοντά στο 0 γίνονται 0
- **2η μέθοδος:** Αφαίρεση βαρών
- Π.χ. Μείωση κατά 53% και 85% στα βάρη των Convolutional και FC επίπεδων => 0.5% accuracy ↓



Αναπαράσταση νευρώνων ενός δικτύου:

Πριν το Weight Pruning

Μετά το Weight Pruning

# Approximate Computing

---

## Reduce Computations - Low Rank Approximation

- Kernel ως separable filters
- Μετασχηματισμός 2D NxM kernel σε δύο 1D πίνακες Nx1 και 1xM.
- Convolution σε εικόνα AxB:
  - ◆ **2D:**  $O(A \cdot B \cdot N \cdot M)$
  - ◆ **1D:**  $O(A \cdot B \cdot (N+M))$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [1 \ 0 \ -1] = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Παράδειγμα Sobel φίλτρου 3x3 διασπασμένο σε γινόμενο ενός 3x1 και 1x3 πινάκων



# FPGA Accelerators

## Results

Table 5: FPGA-based CNN accelerators implementing loop optimization

## Results - Loop Optimizations

- Loop Unroll / Loop Tiling
- Έτσι Lc, Ln

### Σύγκριση:

- [65] 1 Zynq ZC706 + 4 Virtex 7-VC709  
→ 0.8 TOPs, 0.006 TOPs/W, 78% Accur.
- GPU: Nvidia Titan X  
→ 1.4 TOPs, 0.004 TOPs/W, 79% Accur.

### FPGA:

- Παρόμοιο Accuracy
- 40% ↓ Through.
- 60% ↓ Power

Network	Network Workload		Bitwidth	Desc.	Device	Freq (MHz)	Through (GOPs)	Power (W)	LUT (K)	DSP	Memory (MB)	
	Comp. (GOP)	Param. (M)										
[39]	AlexNet-C	1.3	2.3	Float 32	HLS	Virtex7 VX485T	100	61.62	18.61	186	2240	18.4
[9]	VGG16SVD-F	30.8	50.2	Fixed 16	HDL	Zynq Z7045	150	136.97	9.63	183	780	17.5
	AlexNet-C	1.3	2.3					187.24		138	635	18.2
[28]	AlexNet-F	1.4	61.0	Fixed 16	OpenCL	Stratix5 GSD8	120	71.64		272	752	30.1
	VGG16-F	31.1	138.0					117.9	33.93	524	1963	51.4
[77]	AlexNet-C	1.3	2.3	Float 32	HLS	Virtex7 VX485T	100	75.16		28	2695	19.5
	AlexNet-F	1.4	61.0					825.6	126.00		14400	
[65]	VGG16-F	31.1	138.0	Fixed 16	HLS	Virtex7 VX690T	150	1280.3	160.00		21600	
	NIN-F	2.2	61.0					114.5	19.50	224	256	46.6
[78]	AlexNet-F	1.5	7.6	Fixed 16	HDL	Stratix5 GX-A7	100	134.1	19.10	242	256	31.0

Table 2: Comparison of CPU, GPU, FPGA implementations

Device	CPU	GPU+CPU	Work [12]	Work [13]	Design A	Design B	Design C	Design D	Design E
Technology	32nm	28nm	28nm	28nm	28nm	28nm	28nm	28nm	28nm
CNN Model	AlexNet	AlexNet	VGG & AlexNet	VGG& AlexNet	Alex	Alex	VGG	VGG	VGG
Precision	float	float	fixed(16b)	fixed(8-16b)	fixed(16b)	fixed(16b)	fixed(16b)	fixed(16b)	fixed(16b)
Accuracy Top-1	-	54.3%	68.02%	66.58%	52.4%	52.4%	66.51%	66.52%	66.51%
Accuracy Top-5	-	78.7%	87.94%	87.48%	77.83%	77.83%	86.89%	86.92%	86.88%
Frequency (MHz)	3,800	~ 1,000	150	120	150	150	150	150	150
Power (Watt)	87.3	328.3	9	19.1	126	126	35	35	160
Objective	-	-	-	Throughput	Energy*	Latency*	Energy*	Latency*	Through.*
Batch Size	16	256	1	1	16	1	2	1	2
# of FPGAs	-	-	1	1	1+4 <sup>‡</sup>	1+4 <sup>‡</sup>	1+1 <sup>‡</sup>	1+1 <sup>‡</sup>	1+6 <sup>‡</sup>
Through. (GOPs)	34.23	1385.5	137.0	117.8	825.6	128.8	290	203.9	1280.3
Latency (ms)	83.6	89.7	224.6	262.9	104	30.6	213.6	151.8	200.9
E.-E.(GOPs/J)	0.39	4.22	15.2	6.17	6.55 <sup>†</sup>	1.02 <sup>†</sup>	8.28 <sup>†</sup>	5.83 <sup>†</sup>	8.00 <sup>†</sup>

Table 7: FPGA-Based CNN accelerators employing Approximate arithmetic

## Results - Approximate Comp.

### Cifar10

#### □ Binary NN [106]

- 8 bits για In/Out FMs
- 1 bit για Βάρη / Fully Connected
- **9 TOPs, 0.34 TOPs/W, 86% Accur.**

#### □ Ternary NN [107]

- 8 bits για In/Out FMs
- 2 bit για Βάρη / Fully Connected
- **10 TOPs, 0.73 TOPs/W, 90% Accur.**
  - 10% ↑ Through
  - 50% ↓ Power

### ImageNet

#### □ SVD (Low Rank Approx.) [9]

- VGG16-SVD model
- 16 bits Fixed - Bitwidth
- **0,1 TOPs, 0.02 TOPS/W, 88% Accur.**

#### □ Pruning [7]

- Μηδενικά δεν μπαίνουν για MAC processing
- 32 bits Float - Bitwidth
- **12 TOPs, 0.08 TOPS/W, 80% Accur.**
  - 11900% ↑ Through
  - 27% ↓ Power

Dataset	Network Workload			Bitwidth			Acc	Device	Freq (MHz)	Through. (GOPs)	Power (W)	LUT (K)	DSP	Memory (MB)		
	Comp. (GOP)	Param. (M)	In/Out	FMs	W-C	W-FC										
FP32	[61]	ImageNet	30.8	138.0	32	32	32	90.1	Arria10 GX1150	370	866	41.7	437	1320	25.0	
FP16	[30]	ImageNet	1.4	61.0	16	16	16	79.2	Arria10 GX1150	303	1382	44.3	246	1576	49.7	
	[80]	ImageNet	30.8	138.0	16	16	8	8	Arria10 GX1150	150	645		322	1518	38.0	
DFP	[84]	ImageNet	30.8	138.0	16	16	16	16	Arria10 GX1150	200	720		132	1518	44.5	
	[61]	ImageNet	30.8	138.0	16	16	16	16	Arria10 GX1150	370	1790		437	2756	29.0	
BNN	[104]	Cifar10	1.2	13.4	20	2	1	1	Zynq Z7020	143	208	4.7	47	3		
	[52]	Cifar10	0.3	5.6	20/16	2	1	1	Zynq Z7045	200	2465		11.7	83	7.1	
		MNIST	0.0	9.6	8	2	1	1		5905			364	20		
	[106]	Cifar10	1.2	13.4	8	8	1	1	Stratix5 GSD8	150	9396		26.2	438	20	44.2
		ImageNet	2.3	87.1	8	32	1a	1		1964			462	384		
TNN		Cifar10	1.2	13.4					Zynq Z7045	89.4	10962		13.6	275		39.4
	[107]	SVHN	0.3	5.6	8	2	2	2	Xilinx7 VX690T	97.6	86124		7.1	155		12.2
		GTSRB	0.3	5.6						99.0	86124		6.6	155		12.2

Table 8: FPGA-Based CNN accelerators employing pruning and low rank approximation

Dataset	Network Workload			Removed	Bitwidth	Acc (%)	Device	Freq (MHz)	Through. (GOPs)	Power (W)	LUT (K)	DSP	Memory (MB)	
	Comp. (GOP)	Param. (M)	Param. (%)											
SVD	[9]	ImageNet	30.5	138.0	63.6	16 Fixed	87.96	Zynq 7Z045	150	137	9.6	183	780	17.5
Pruning	[45]	Cifar10	0.3	132.9	89.3	8 Fixed	91.53	Kintex 7K325T	100	8621	7.0	17	145	15.1
	[7]	ImageNet	1.4	61.0	85.0	32 Float	79.70	Stratix 10	500	12000	141.2			

# Results

## [4] High Performance Scalable FPGA Accel.

- 8 bits για Activation + 2 bits για Βάρη (Dynamic FP)
- Χρήση μόνο Logic Blocks για NN compute
- Υλοποίηση του ResNet50
- **76 AI-TOPs, 0.7 TOPs/W, 71% Accur.**

### GPU: Nvidia Tesla P100

- 16 bits Fixed - Bitwidth
- **18.7 TOPs, 0.075 TOPs/W**
  - ◆ 75% ↓ Through + 90% ↓ TOPs/W

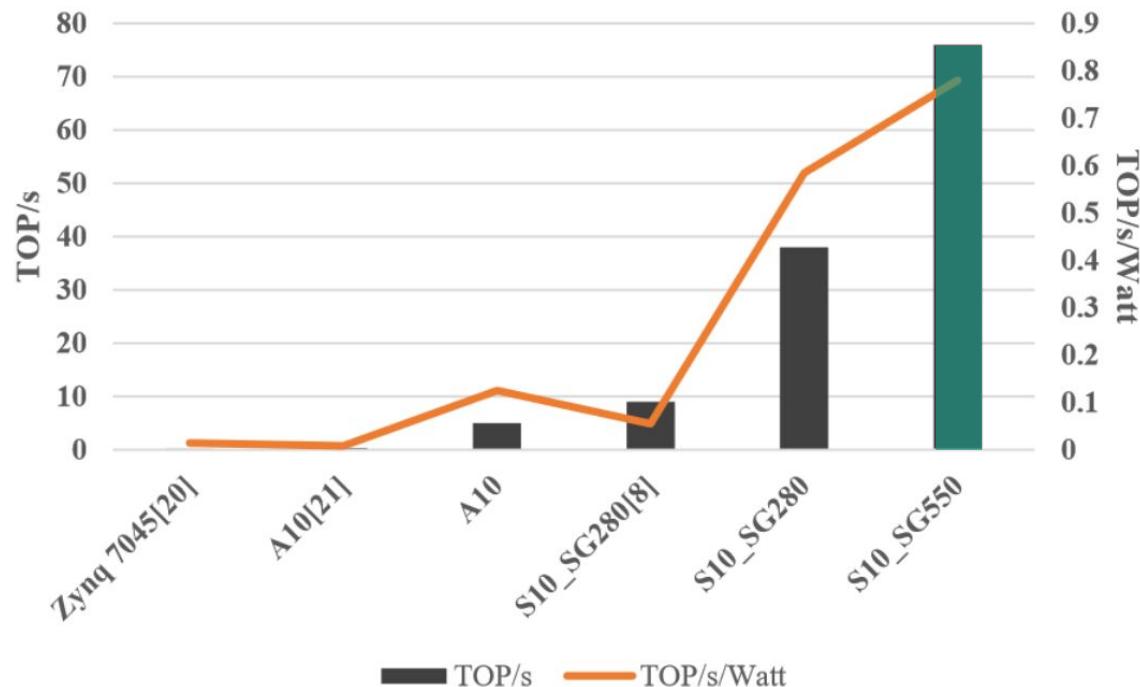
### GPU: Nvidia V100

- 16 bits Fixed - Bitwidth
- **125 TOPs, 0.42 TOPs/W**
  - ◆ 65% ↑ Through + 40% ↓ TOPs/W

GPU	TOP/s	Power (W)	Precision	TOP/s/Watt
V100	125	300	FP16	0.416667
P4	22	75	INT8	0.293333
P40	47	250	INT8	0.188
P100	18.7	250	FP16	0.0748

Θεωρητικά Peak **TOP/s** και **TOP/s/Watt** τελευταίων GPU της NVIDIA για Inferencing με χαμηλό Precision

**Performance/Watt** για τις καλύτερες υλοποιήσεις στα Arria 10 (A10) και Stratix 10 (S10) FPGA για το ResNet50 δίκτυο μαζί με προηγούμενες δουλειές.





# Ευχαριστούμε!

**WHEN YOU THINK THE SLIDES ARE OVER**



imgflip.com

# References

---

- [1] [A Survey of FPGA-Based Neural Network Inference Accelerator](#)
- [2] [Accelerating CNN inference on FPGAs: A Survey](#)
- [3] [FPGA-Based CNN Inference Accelerator Synthesized from Multi-Threaded C Software](#)
- [4] [High Performance Scalable FPGA Accelerator for Deep Neural Networks](#)
- [5] [FPGA Acceleration of Binary Weighted Neural Network Inference](#)
- [6] [Where The FPGA Hits The Server Road For Inference Acceleration](#)
- [7] [In-Datacenter Performance Analysis of a Tensor Processing Unit](#)
- [8] [A Gentle Introduction to Batch Normalization for Deep Neural Networks](#)
- [9] [Introduction to FPGA and It's Programming Tools](#)
- [10] [Xilinx All Programmable Devices:A Superior Platform for Compute-Intensive Systems](#)
- [11] [ImageNet Classification with Deep Convolutional Neural Networks](#)
- [12] [Training Deep Neural Networks With Low Precision Multiplications](#)
- [13] [BinaryConnect: Training Deep Neural Networks with binary weights during propagations](#)
- [14] [Double MAC: Doubling the Performance of Convolutional Neural Networks on Modern FPGAs](#)
- [15] [Energy-Efficient CNN Implementation on a Deeply Pipelined FPGA Cluster](#)