ASP-DAC 2022

Pearl: Towards Optimization of DNN-accelerators Via Closed-Form Analytical Representation

Authors: Arko Dutt, Suprojit Nandy, Mohamed Mostafa Sabry Aly

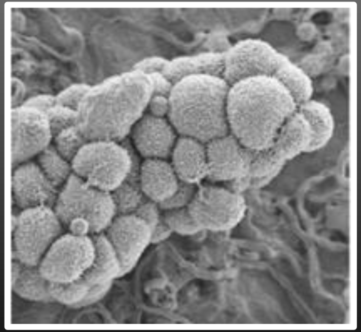
Widespread Deep Learning Applications

- Thanks to high accuracy of deep neural networks (DNN)
    - e.g. convolutional neural networks (CNN) for classification / object detection

Internet & Cloud

Autonomous Cars

Media & Entertainment

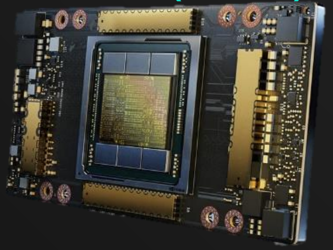

Medicines

Defense & Security

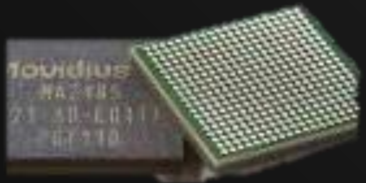
Big Market for Huge Demand

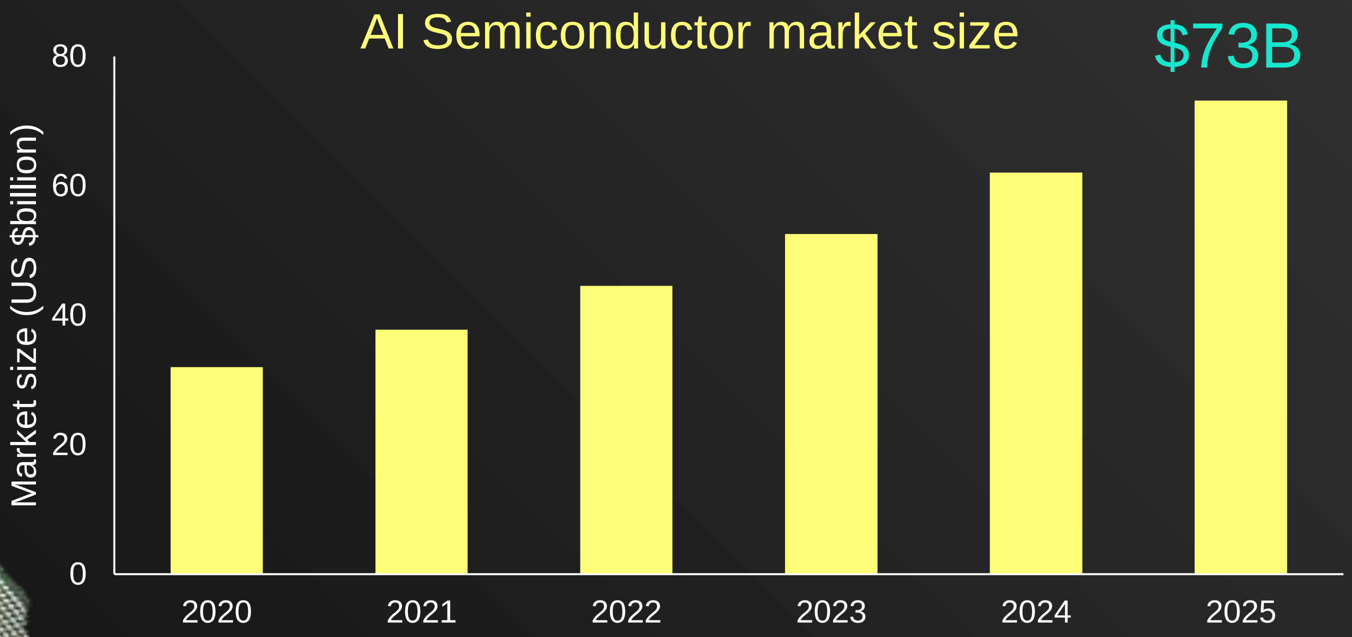
- Demand for deep learning ➔ dedicated DNN chips
- Multibillion dollar market size

GPUs (A100)

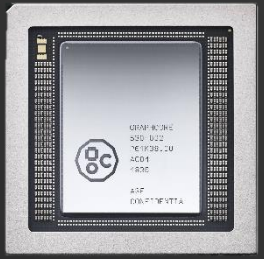


Intel Movidius Myrad X

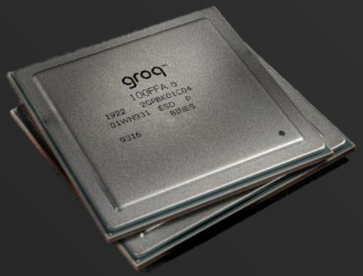




Graphcore



Groq



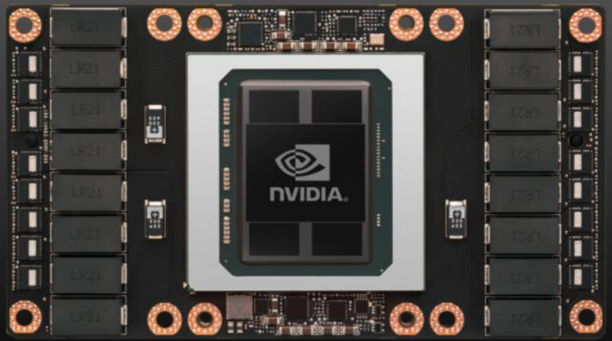
Mckinsey report, “Artificial-intelligence (AI) hardware: New opportunities for semiconductor companies,” 2019

Dedicated Hardware for Deep Learning

General Purpose

CPU: *Intel Xeon Phi*

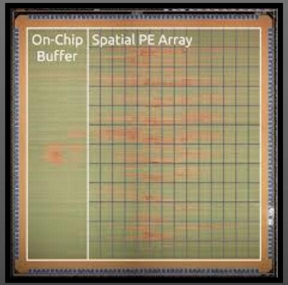
GPU: *NVIDIA Volta*



Application-Specific

Custom Accelerator

Eyeriss



Myriad 2

Nervana

Cambricon

Google TPU

Intel DLIA

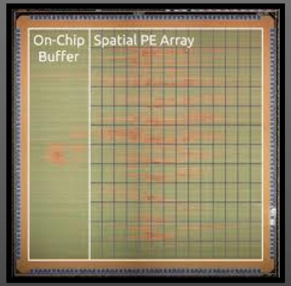
Which Hardware (HW) is Most Efficient?

Custom Accelerator

Myriad 2

Nervana

Eyeriss



Cambricon

Google TPU

Intel DLIA

- Is any HW architecture optimal in regard to applications needs?

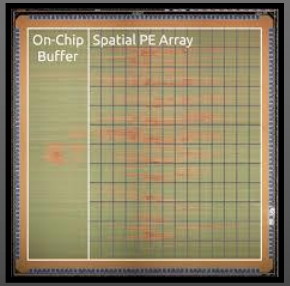
Solution

Custom Accelerator

Myriad 2

Nervana

Eyeriss



Cambricon

Google TPU

Intel DLIA

- Is any HW architecture

  optimal in regard to applications needs?

- To ensure optimality

  model, simulate, explore HW choices

Current Modelling & Simulation Approaches

Runtime Decreases

Several minutes to hours/days

Cycle-accurate

- Shao et al. 2014
- Samajdar et al. 2018, 2020
- Binkert et al. 2011

. . .

Several milli-seconds

Machine Learning

- Reagen et al. 2017
- Venkatesan et al. 2019
- Dutt et al. 2019

Several micro to milli-seconds

Analytical

- Kwon et al. 2019
- Palit et al. 2019
- Sze et al. 2020
- Parashar et al. 2019
- Wu et al. 2019
- Samajdar et al. 2020

Accuracy Increases

Similar Accuracy

Objective

- Create analytical models to estimate performance/energy that are
  - extremely fast
  - highly-accurate comparable to cycle-accurate model
  - capable to be used in fast optimization methods

Outline

- *Pearl* analytical modelling
- *Pearl* simulation and evaluation
- *Pearl* use-case in accelerator design-space exploration
- Conclusion

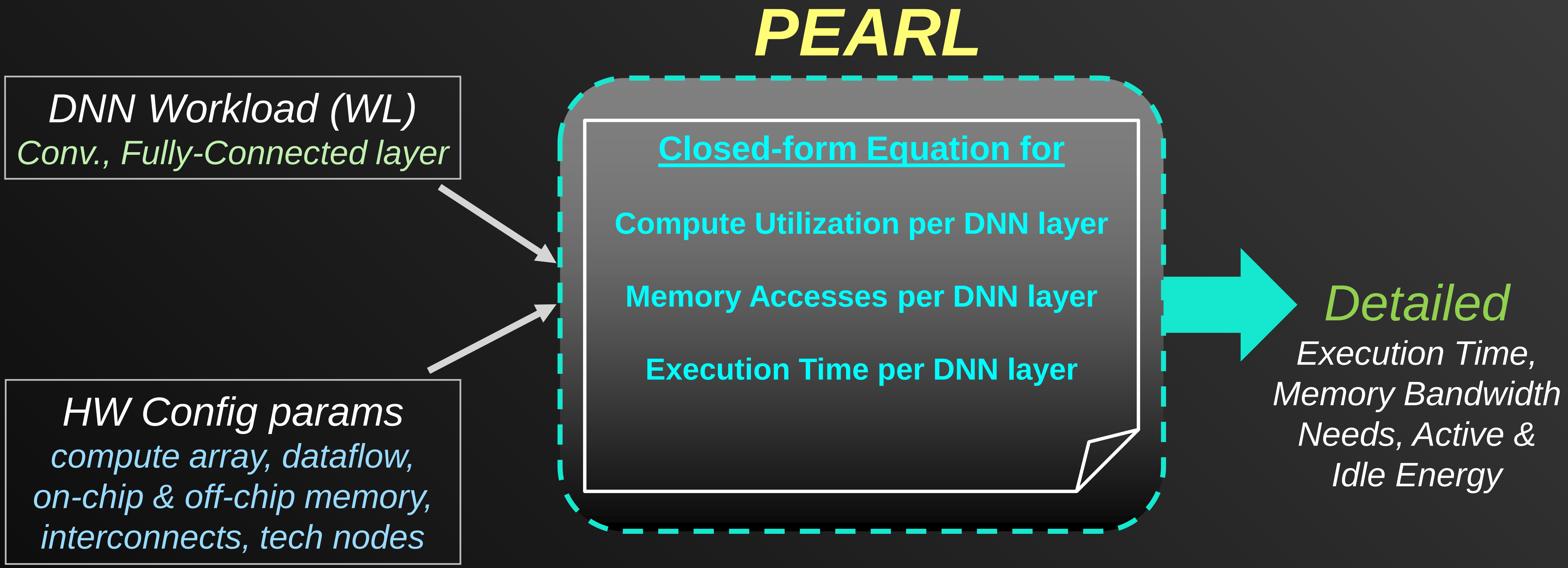
Pearl Analytical Modelling

Key Messages

- *Pearl* attains < 1.3% average error versus cycle-accurate SOTA
- *Pearl* achieves > $10^7\times$ average speedup over cycle-accurate SOTA
- *Pearl enables* fast methods to optimize DNN accelerators

SOTA: State-Of-The-Art

Fast & Accurate Analytical Model


PEARL
DNN Workload (WL)
Conv., Fully-Connected layer
HW Config params
compute array, dataflow, on-chip & off-chip memory, interconnects, tech nodes
Closed-form Equation for
Compute Utilization per DNN layer
Memory Accesses per DNN layer
Execution Time per DNN layer
Detailed
Execution Time, Memory Bandwidth Needs, Active & Idle Energy

Execution Time of DNN Workload

$$T = \sum_{\{i=0\}}^{\{N-1\}} t_t(i)$$

$$t_t(i) = \alpha(i) \cdot \max\{t_c(i), t_m(i)\} + [1 - \alpha(i)] \cdot [t_c(i) + t_m(i)]$$

A single DNN layer number ‘i’

Number of (Conv + FC) layers in a DNN ‘N’

Execution time per layer $t_t(i)$

Pipelining co-efficient per layer $\alpha(i) \epsilon [0,1]$

Compute time per layer $t_c(i)$

Memory-access time per layer $t_m(i)$

Compute Time per DNN Layer

Acronym	Description
$MACs(i)$	Multiply-accumulate operations per layer
$\beta(i) \in (0,1]$	Compute array utilization rate per layer
f_{op}	Operating frequency
PE_H	Compute Array Height
PE_W	Compute Array Width

$$t_c(i) = \frac{MACs(i)}{\beta(i) \times f_{op} \times PE_H \times PE_W}$$

Compute Time per DNN Layer

- Model PE utilization rate $\beta(i)$ for different dataflow mapping
  - output-stationary (OS), weight-stationary (WS), input-stationary (IS)

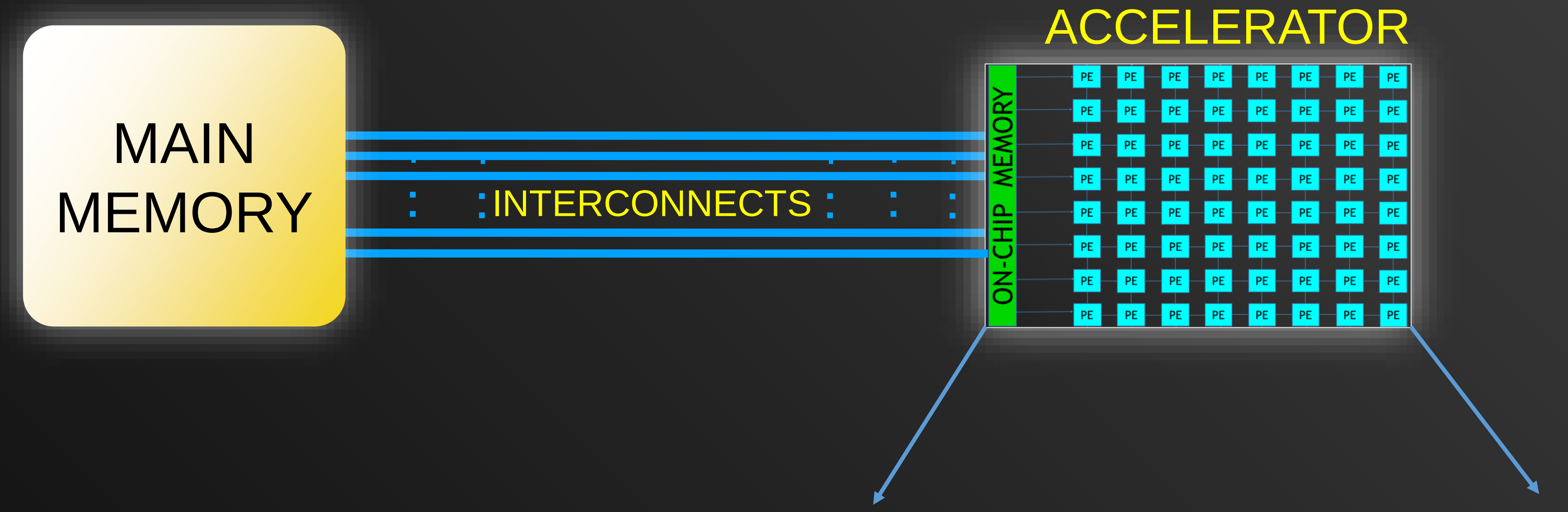
$$t_c(i) = \frac{MACs(i)}{\beta(i) \times f_{op} \times PE_H \times PE_W}$$

Utilization Rate Formulation – $\beta(i)$

$$\beta(i) = \beta_{row}(i) \times \beta_{col}(i)$$

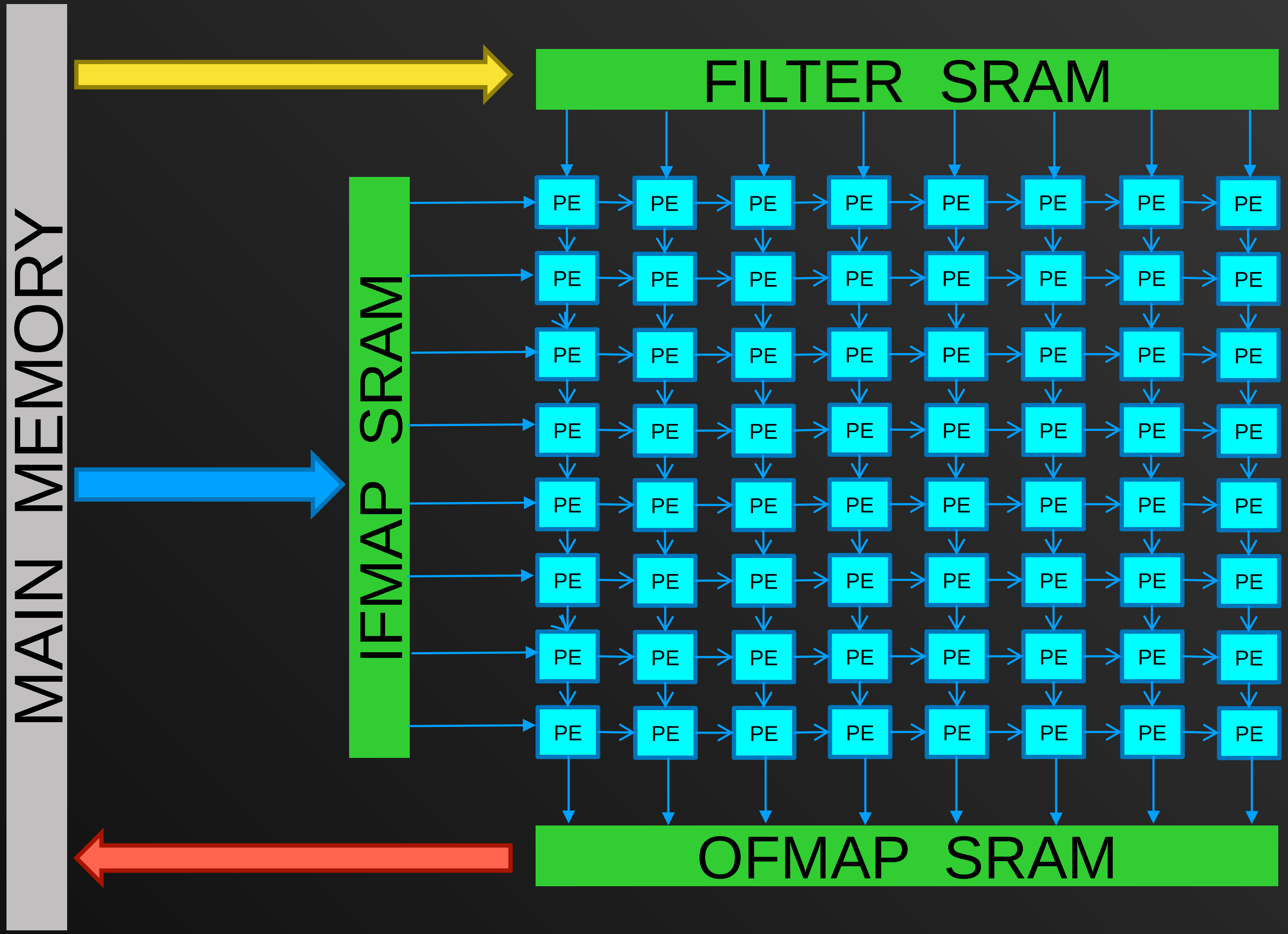
- $\beta_{col}(i)$ is PE array column utilization rate for layer i
- $\beta_{row}(i)$ is PE array row utilization rate for layer i

Case Study: Accelerator Architecture

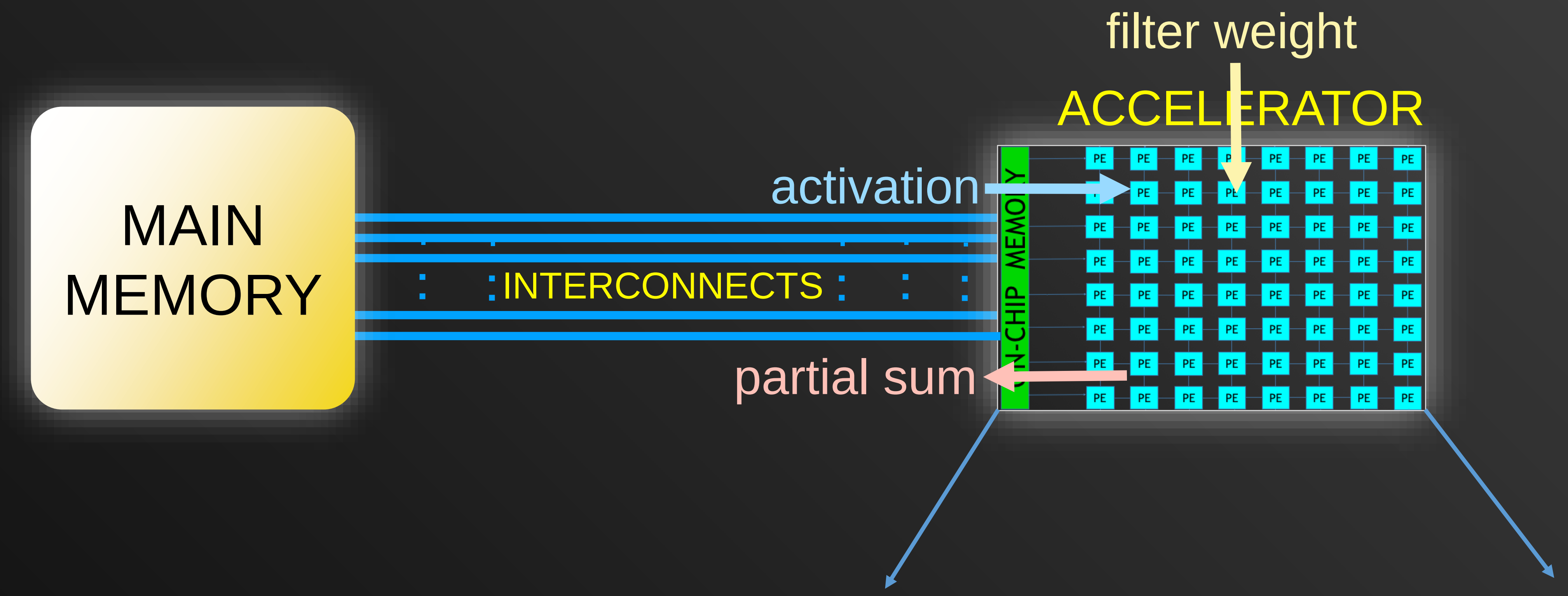


Systolic Array Architecture

Dataflow in Systolic Array Architecture



Example Dataflow



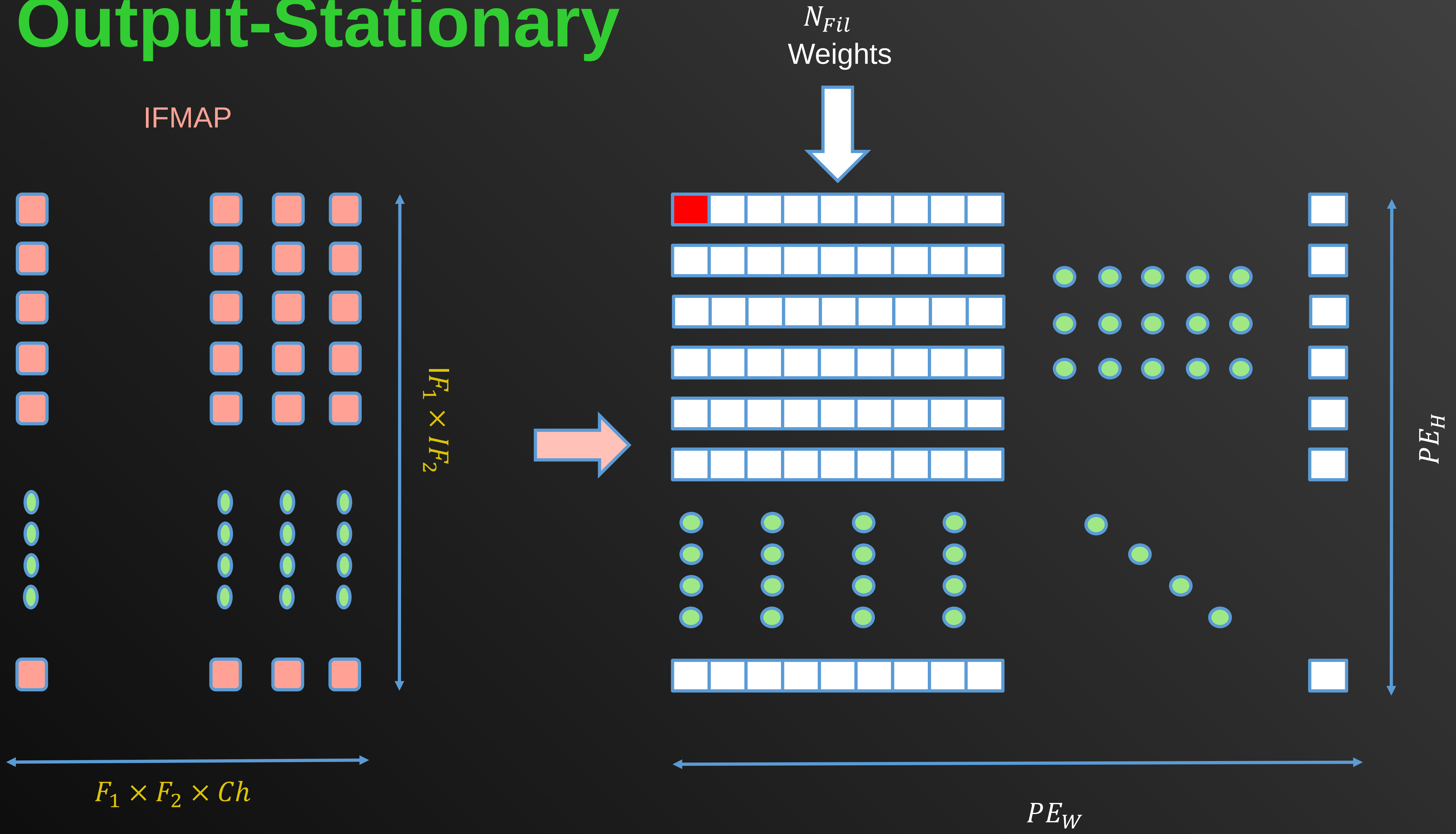
Output-stationary (OS)

OS: Input Feature Map and Filter weights are streamed-in, while each pixel of Output Feature Map is fixed onto a given PE.

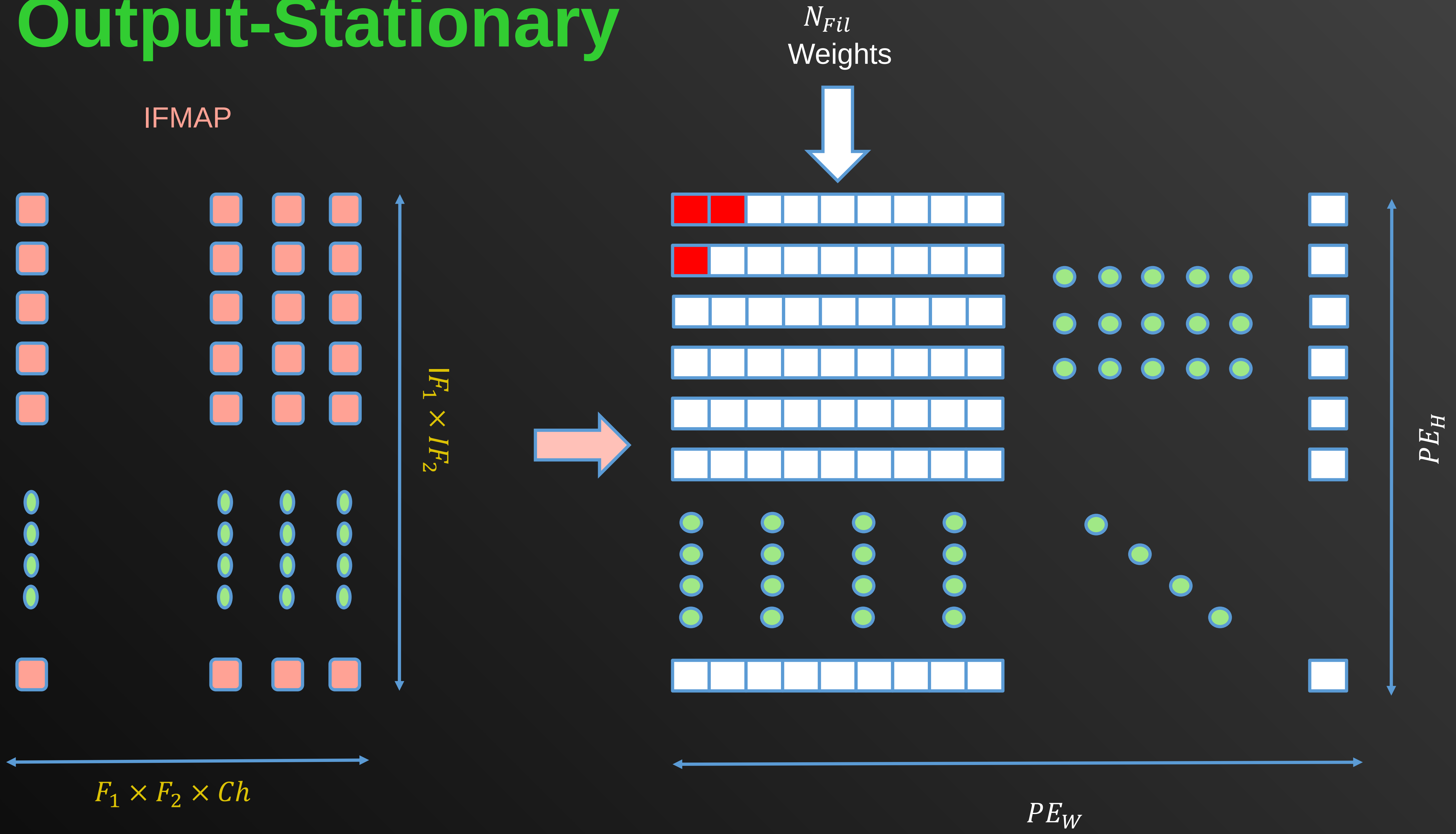
Illustrative Demonstration

Parameter	Value
Number of Filters (N_{Fil})	64
Input Activation ($IF_1 \times IF_2$)	112x112
PE Array Size ($PE_H \times PE_W$)	64 x 64
Global Buffer (SC)	128 KB
Filter Size ($F_1 \times F_2 \times Ch$)	7 x 7 x 3
Dataflow Mapping	OS

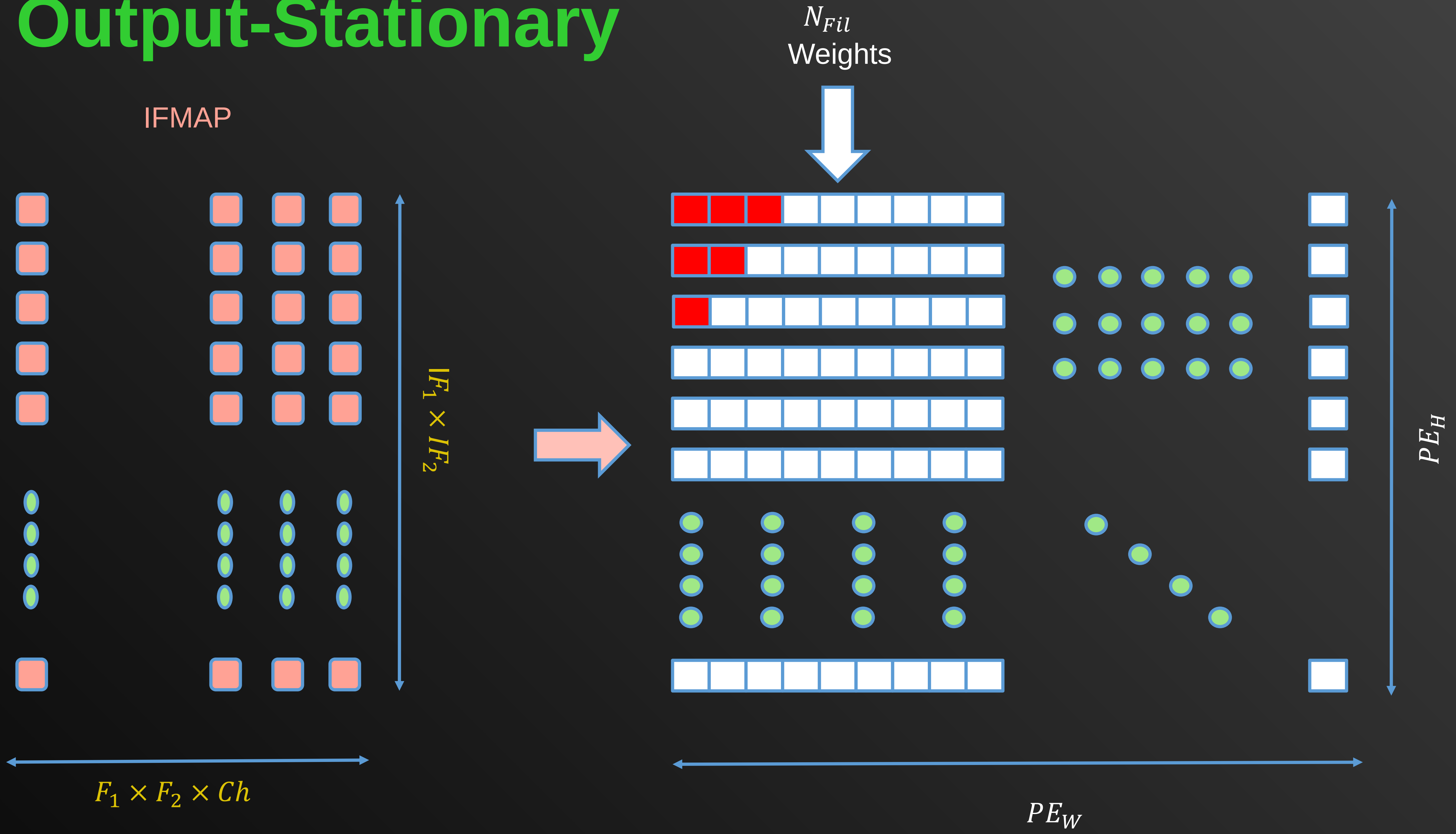
Output-Stationary



Output-Stationary

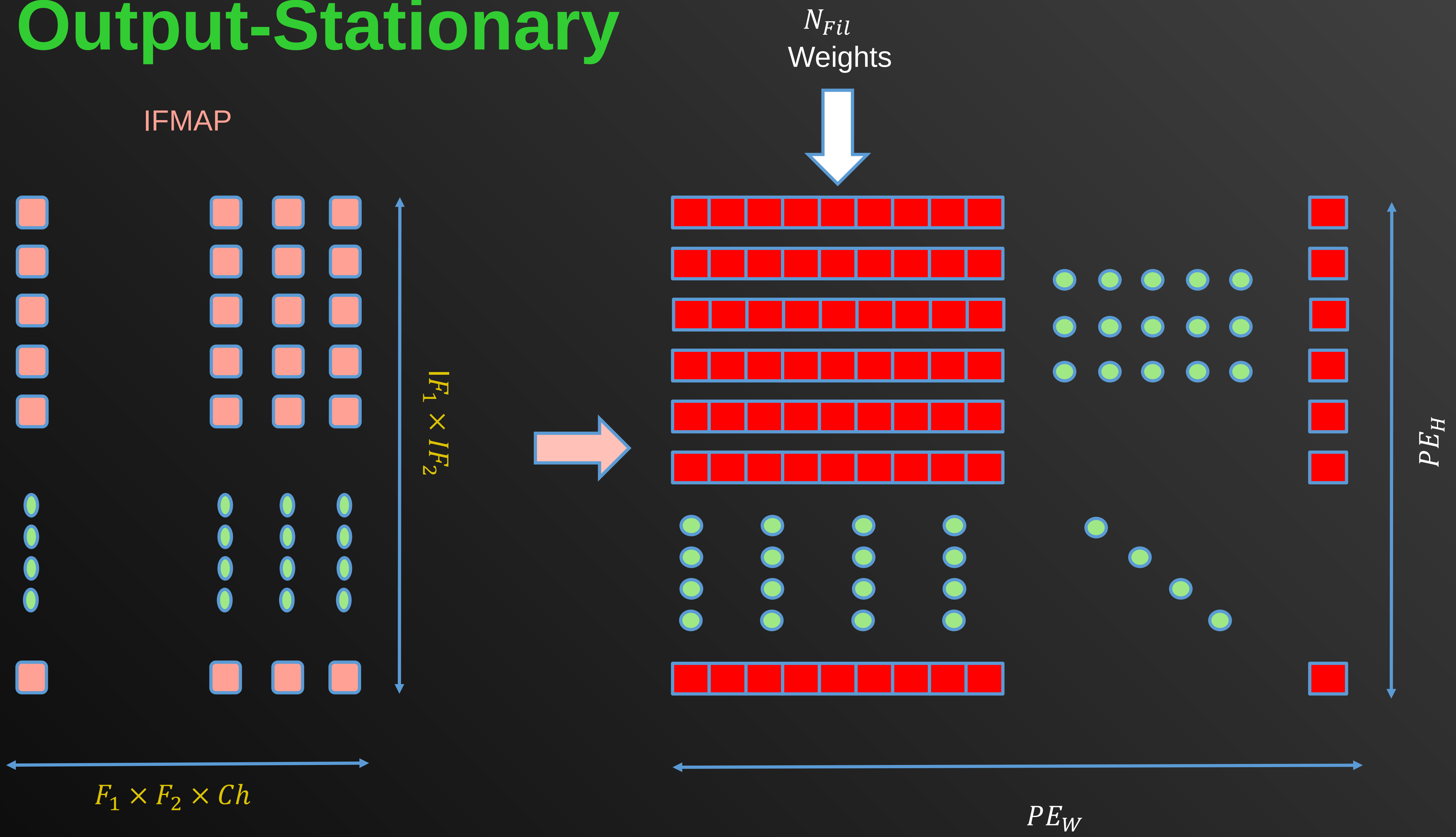


Output-Stationary

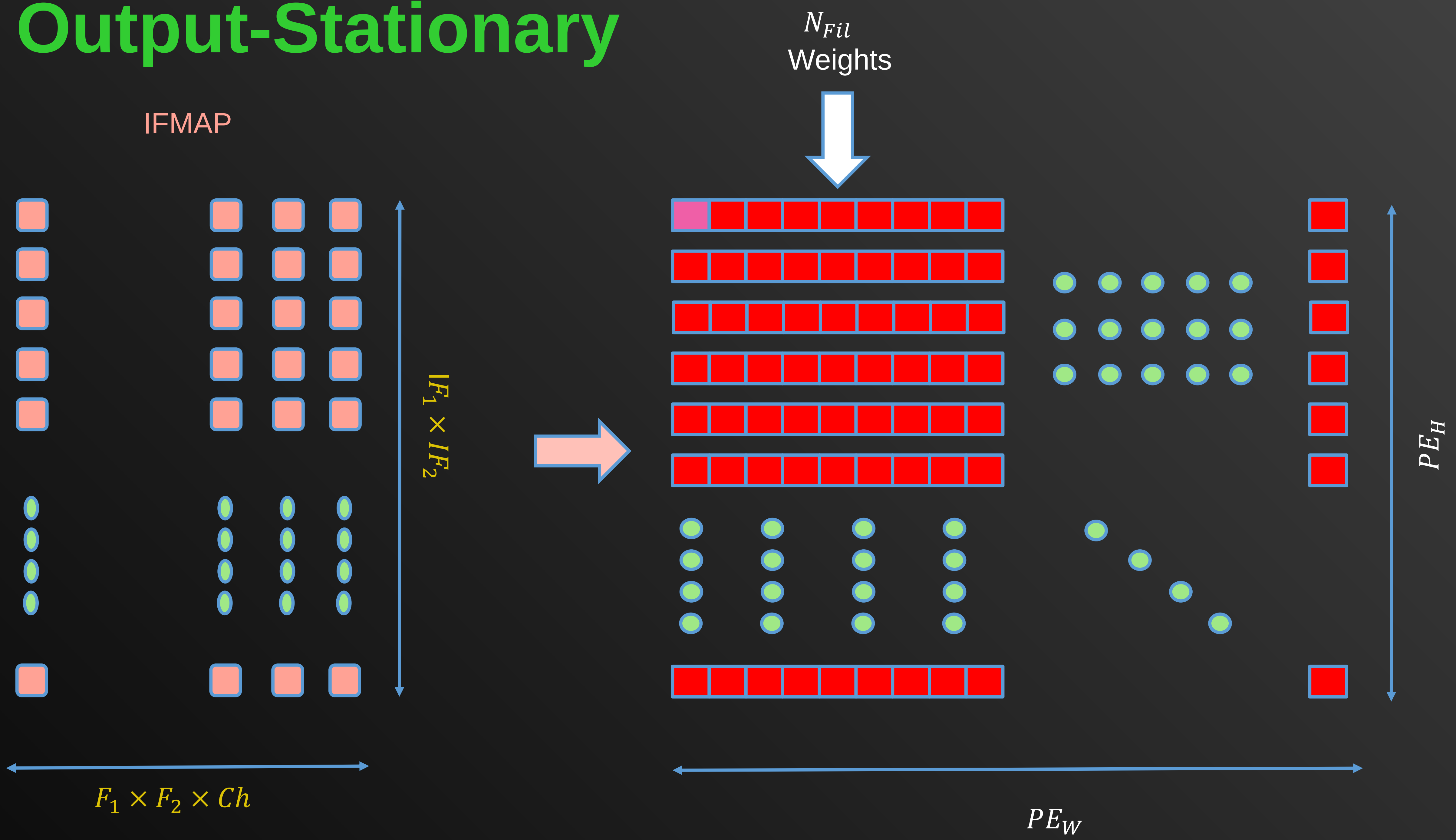


After Filter Size = 7x7x3 = 147 Clock Cycles

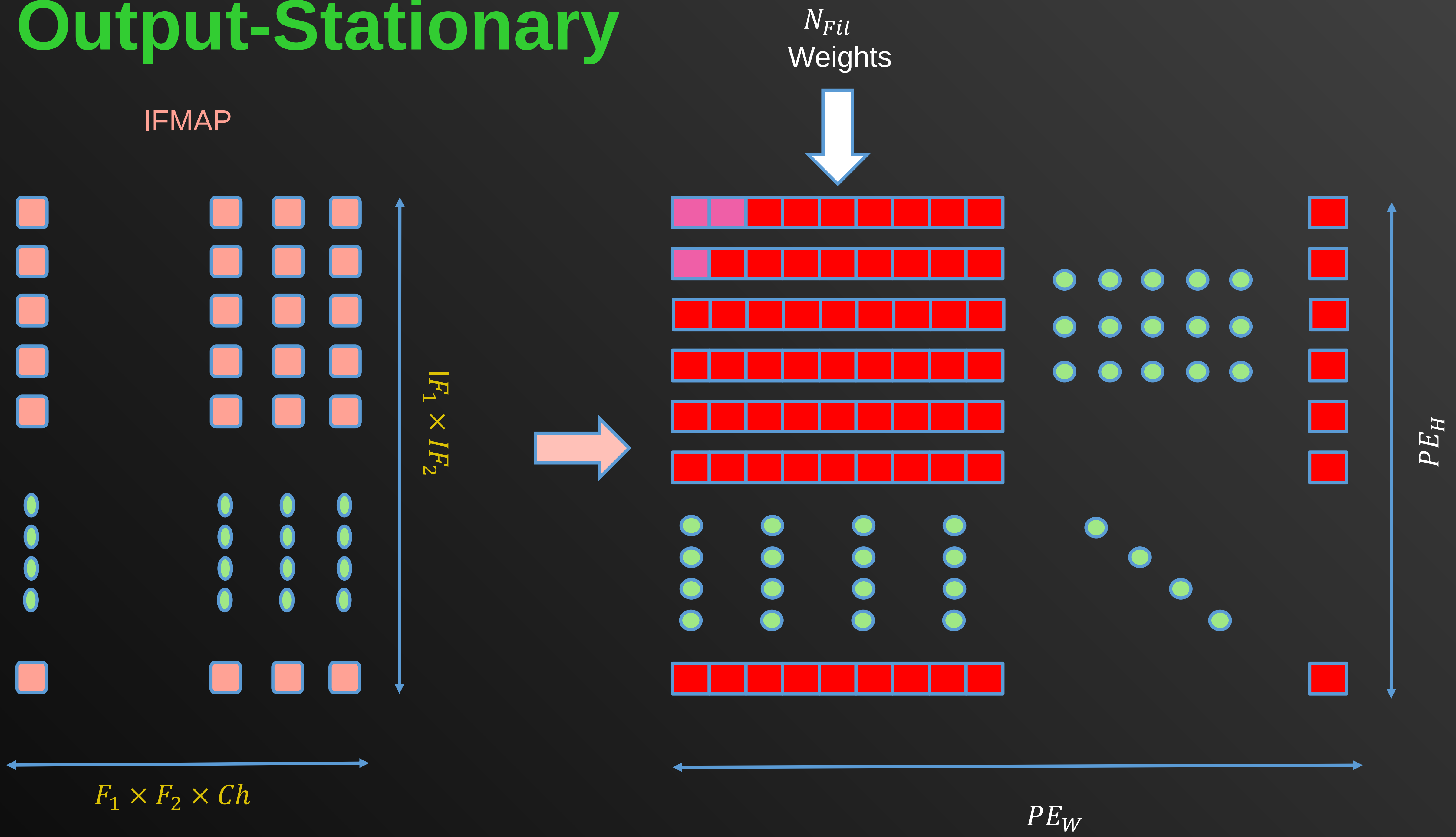
Output-Stationary



Output-Stationary



Output-Stationary



Utilization Rate for Different Dataflows

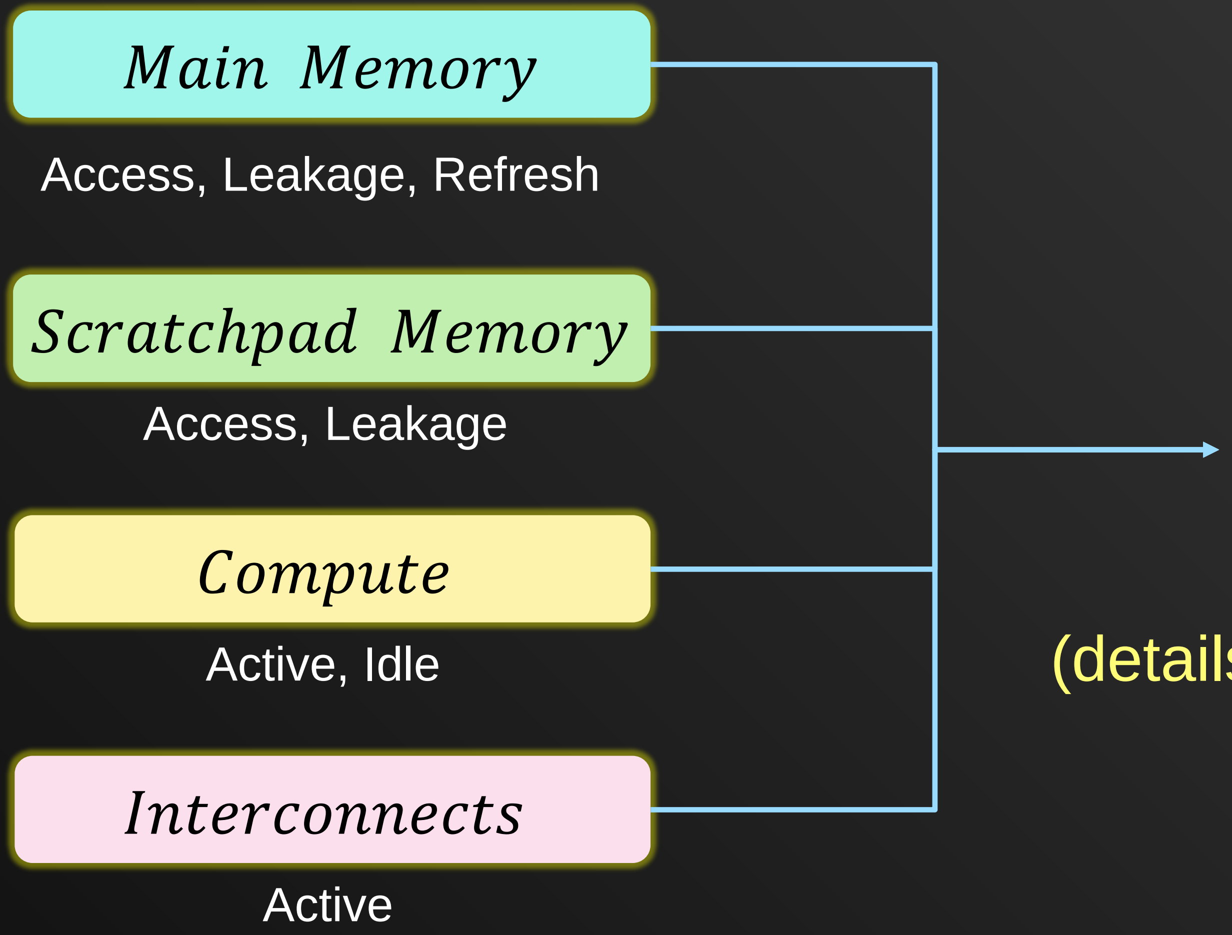
- Utilization rate $\beta(i)$ is modelled for OS, WS, IS dataflows

$$\beta(i) = \beta_{row}(i) \times \beta_{col}(i)$$

- Column & row utilization rates are analytically modelled
  - different set of equations for each dataflow (details in paper)

Energy Consumption for DNN Layer

Energy of each component is analytically modelled



Total Energy Consumption

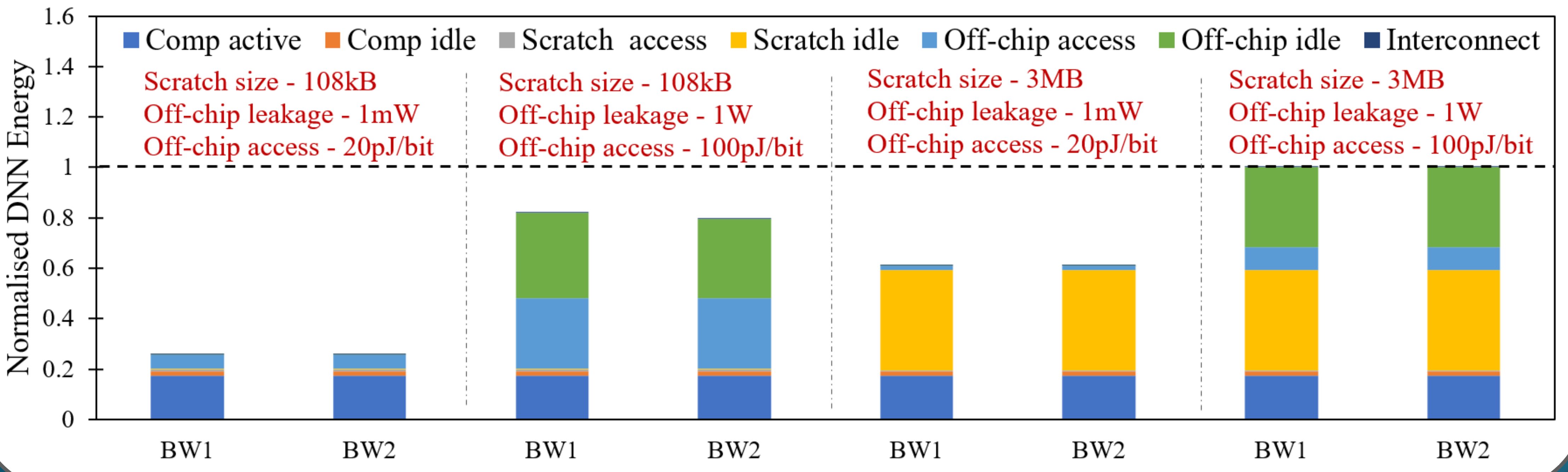
(details of energy models in paper)

Energy Breakdown & *Pearl* Capabilities

- Sensitivity analysis by varying parameters for 16x16 PE Array
  - Scratchpad size: {108kB, 3MB}
  - Off-chip main memory bandwidth: {1GB/s, 50GB/s}
  - Off-chip main memory leakage: {1mW, 1W}
  - Off-chip main memory access energy: {20pJ, 100pJ}

Energy Breakdown & *Pearl* Capabilities

16x16 PE Array



Scratch: Scratchpad Memory

Off-chip Main Memory Bandwidths: BW1 – 1GB/s, BW2 – 50GB/s

Pearl Evaluation

Accuracy: *Pearl* vs Cycle-accurate

- Cycle-accurate simulation baseline
  - *Pearl* versus SCALE-Sim*
  - assume sufficient scratchpad, i.e. $\alpha(i)$ =1

Dataflow Mapping	Workload & Hardware Configurations	Average Error (%)	
		DNN Execution Time	DNN Energy Consumption
OS	32 (4 DNN, 2 Input Data, 4 Compute Arrays)	0.60	0.66
WS		1.49	1.75
IS		0.87	1.34

* *Samajdar et al.*, in *arxiv 2018, ISPASS 2020*

Pearl Simulation Speedup

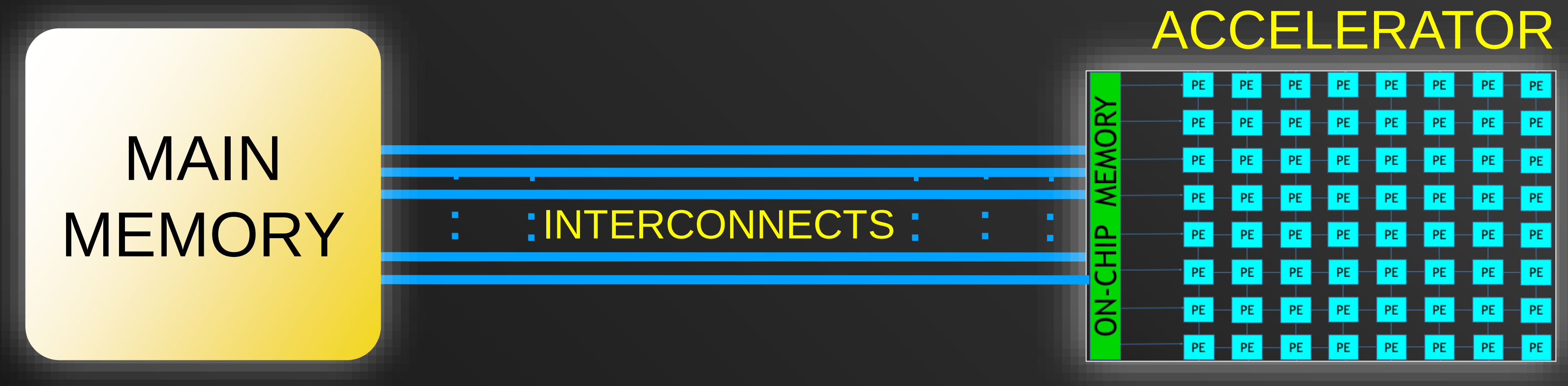
- Average simulation time in *Pearl* (Python-based)
  - fraction of a milli-second for full-DNN workload

Baseline	**Speedup**
Cycle-accurate*	$> 10^7\times$
Analytical*	$> 400\times$

* *Samajdar et al.*, in *arxiv 2018, ISPASS 2020* * *Kwon et al.*, in *MICRO, 2019*

Exploration and Optimization with *Pearl*

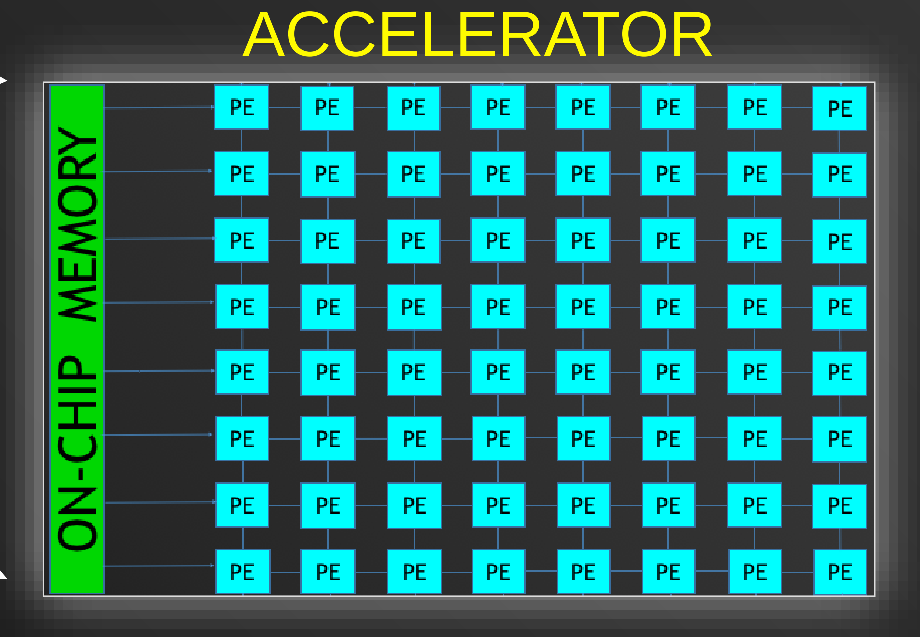
DNN System Consideration



- Main memory
  - off-chip DRAM
- Dataflow mapping
  - output-stationary
- Architecture template
  - systolic array

Apply a constraint: Accelerator Area

Apply
constraint →
area



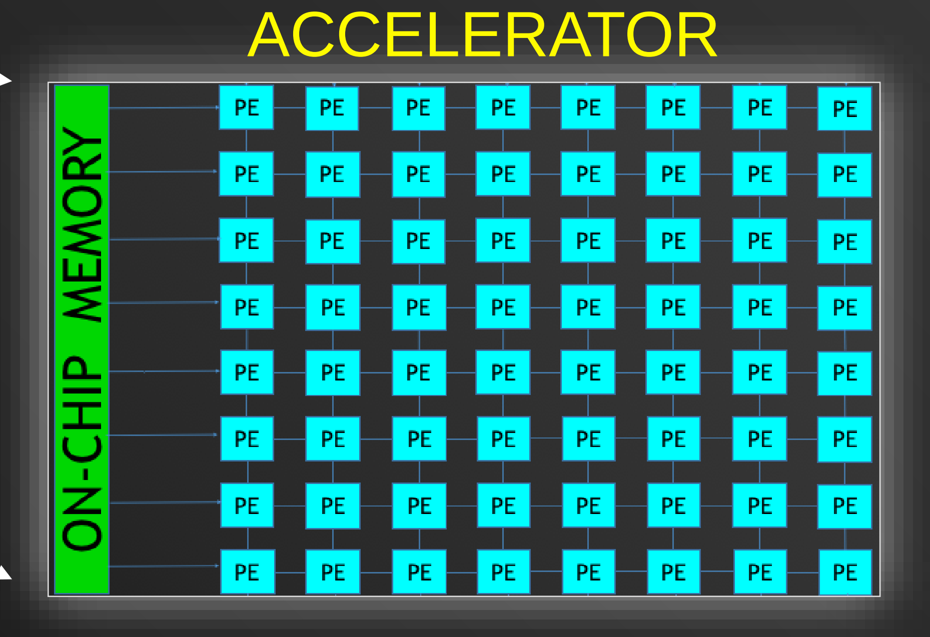
Split Area b/w PE, Buffers, Interconnects

Apply constraint area → Find possible accelerator → Range of PE size, SRAM within area limit

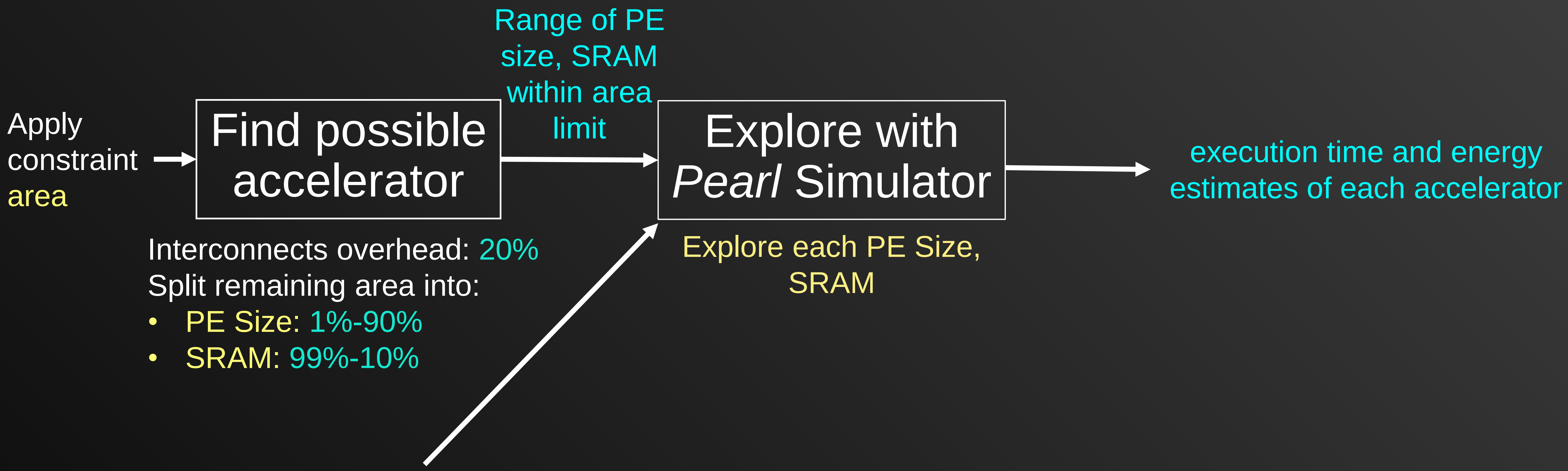
Interconnects overhead: 20%

Split remaining area into:

- PE Size: 1%-90%
- SRAM: 99%-10%



Estimate DNN time and Energy

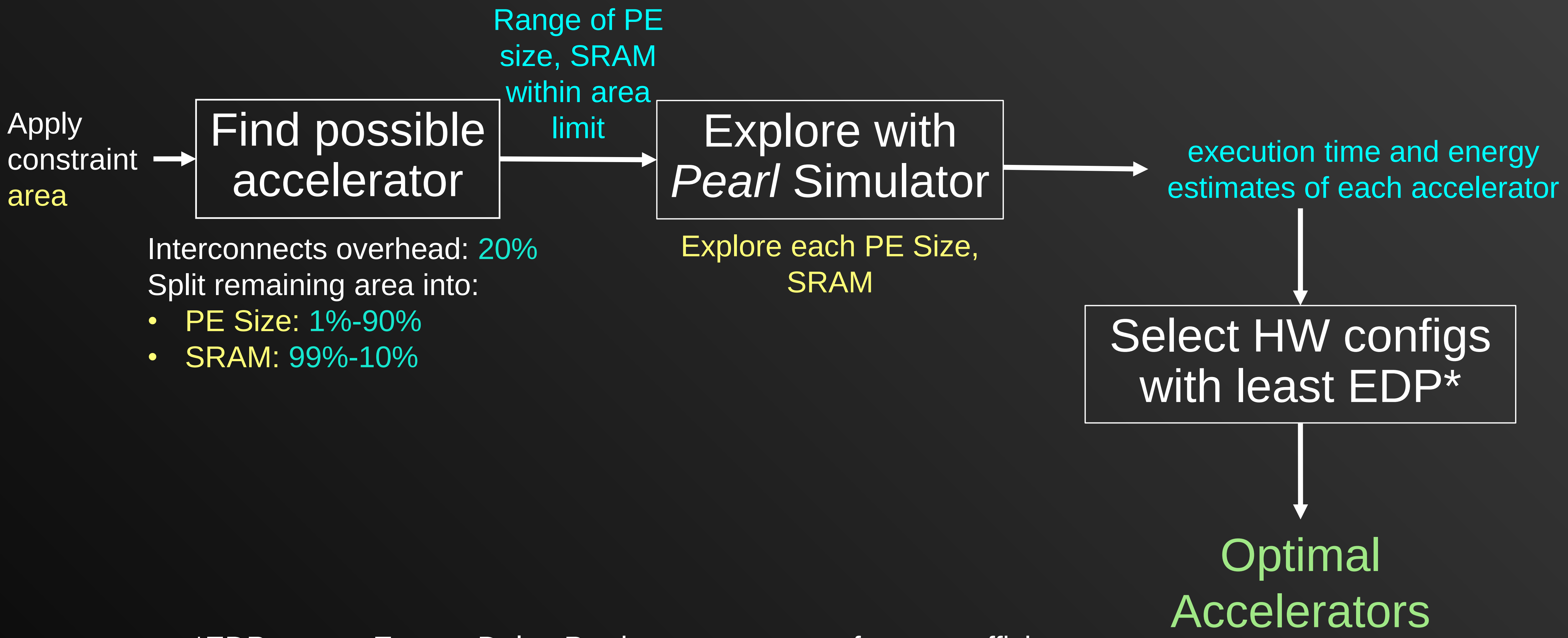

Apply constraint area
Find possible accelerator
Range of PE size, SRAM within area limit
Explore with *Pearl* Simulator
execution time and energy estimates of each accelerator
Explore each PE Size, SRAM

Interconnects overhead: 20%
Split remaining area into:

- PE Size: 1%-90%
- SRAM: 99%-10%

Technology
PE, SRAM – 28nm CMOS
Main memory – DRAM LPDDR3

Select Optimum Solutions


Apply constraint area
Find possible accelerator
Range of PE size, SRAM within area limit
Explore with *Pearl* Simulator
execution time and energy estimates of each accelerator
Select HW configs with least EDP*
Optimal Accelerators
Interconnects overhead: 20%
Split remaining area into:
• PE Size: 1%-90%
• SRAM: 99%-10%
Explore each PE Size, SRAM

*EDP means Energy-Delay-Product, a measure of energy-efficiency

Energy-efficient DNN Accelerators

- Target image classification
  - resolution – Full HD

- Consider area constraint
  - $190\,mm^2$ $(similar\ to\ TPU\ - 256 \times 256\ PE\ array,\ 24MB\ SRAM)$

DNN	PE Size		SRAM (KB)	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
Resnet101	275x275	296x296	13816	15296
Resnet152	275x275	334x334	10856	15296
Alexnet	394x394	423x423	2467	5428
VGG19	362x362	423x423	2467	8388

Conclusion

- *Pearl* attains < 1.3% average error versus cycle-accurate SOTA
- *Pearl* achieves > $10^7\times$ average speedup over cycle-accurate SOTA
- *Pearl* enables fast methods to optimize DNN accelerators

SOTA: State-Of-The-Art

Thanks