

# ConfuciuX: Autonomous Hardware Resource Assignment for DNN Accelerators using Reinforcement Learning (MICRO 2020)

Presentation: Constant Park (Sang-Soo Park)

# 컴퓨터 내부의 연산을 위한 장치 프로세서

- **프로세서 (Processor)**

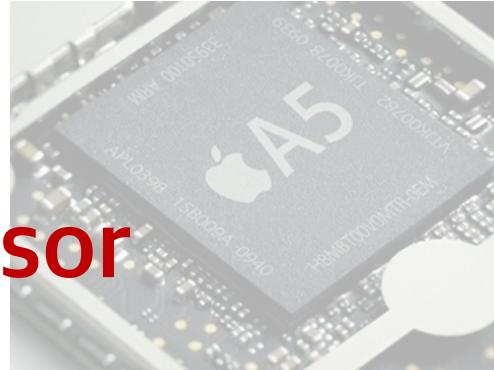
- 컴퓨터 내에서 프로그램 (연산)을 수행하는 하드웨어 장치
- CPU/GPU, DSP, NPU (Neural Processing Unit)



CPU: Ryzen Threadripper



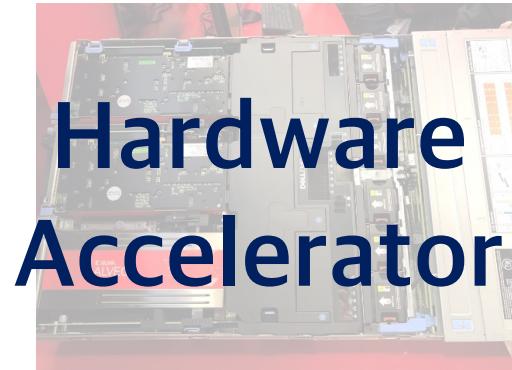
GPU: VOTLA, TURING



SoC: Smart Phone

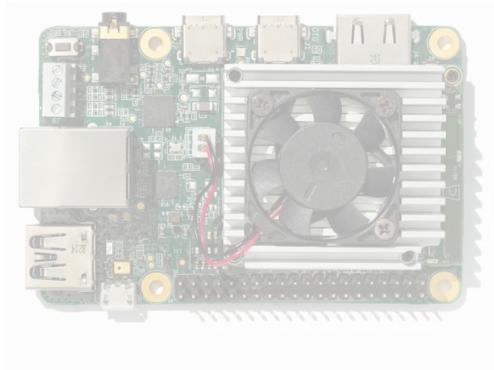


NPU: Graphcore (ASIC)



Hardware Accelerator

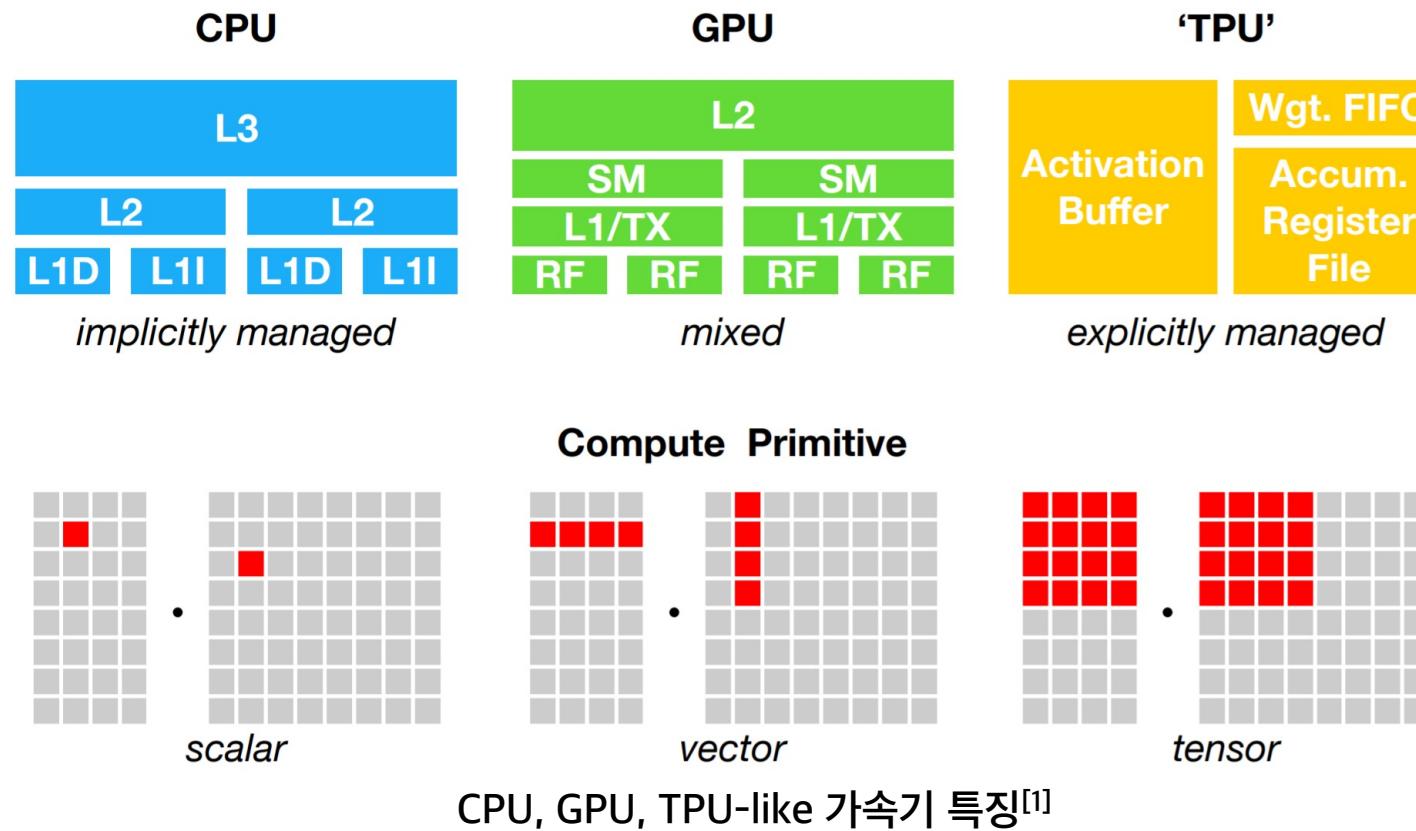
NPU: ALVEO (FPGA)



NPU: Edge TPU (ASIC)

# 심층신경망 프로세서 특징

- CPU, GPU와 심층신경망 프로세서 비교
  - CPU: Integrated multi-core (Good for complex problem)
  - GPU: Many parallel computing unit (Simple and Massive problem)
  - NPU: Under  $10^4$  processing element (Optimized for matrix multiplication)

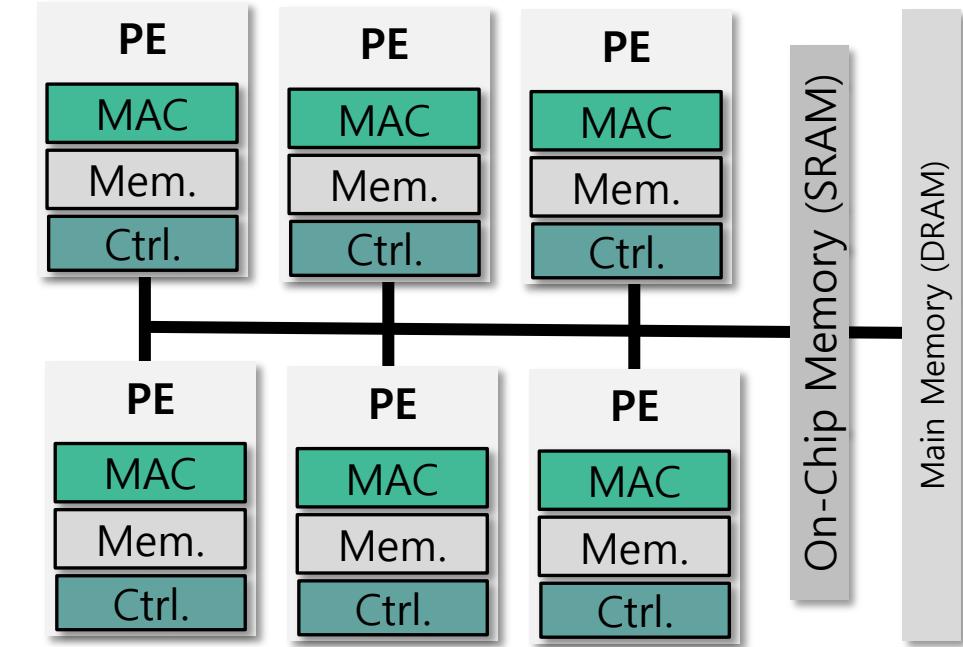
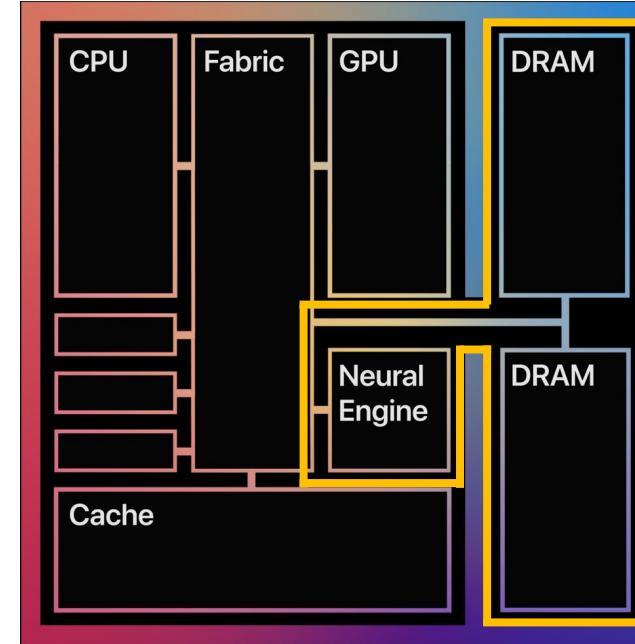
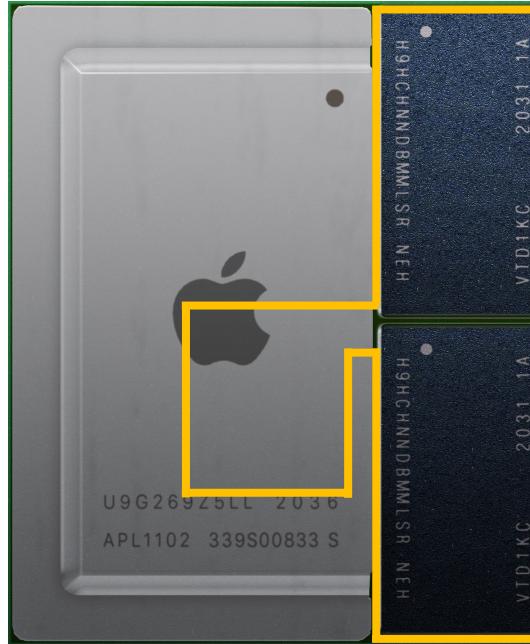


[1] TVM: An Automated End-to-End Optimizing Compiler for Deep Learning, arXiv, 2018.

# 심층신경망 프로세서 구조

- 다수의 PE와 메모리로 구성된 병렬 프로세서

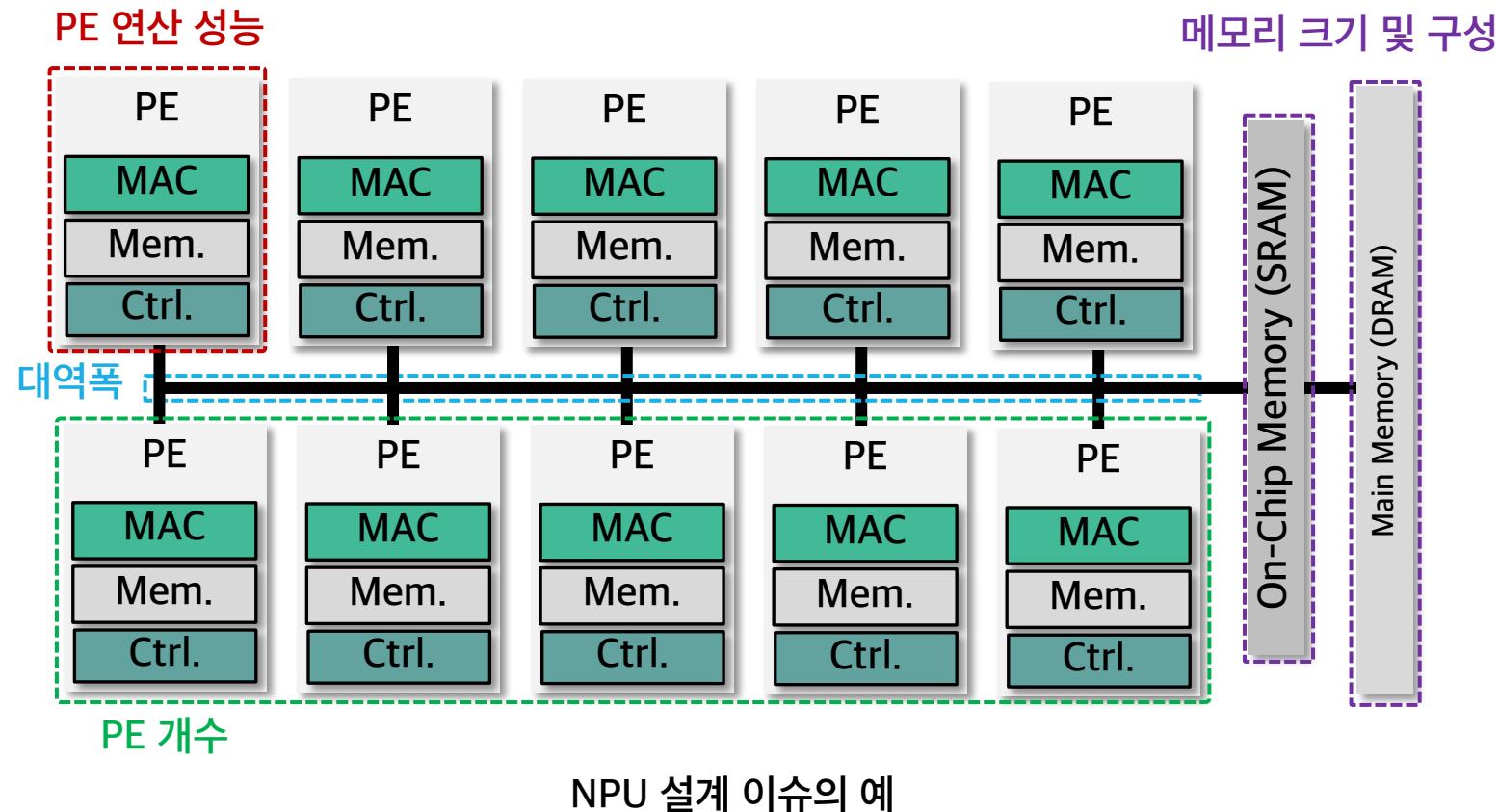
- Processing element (PE): 덧셈과 곱셈 연산 (MAC Operation)을 수행하는 연산 유닛
- On-Chip Memory (SRAM): Feature map, Weight의 일부분을 칩 내부에 저장하는 메모리
- Main Memory (DRAM): 연산에 사용되는 모든 데이터를 저장하는 메모리



실리콘 맥 내부의 NPU 구조

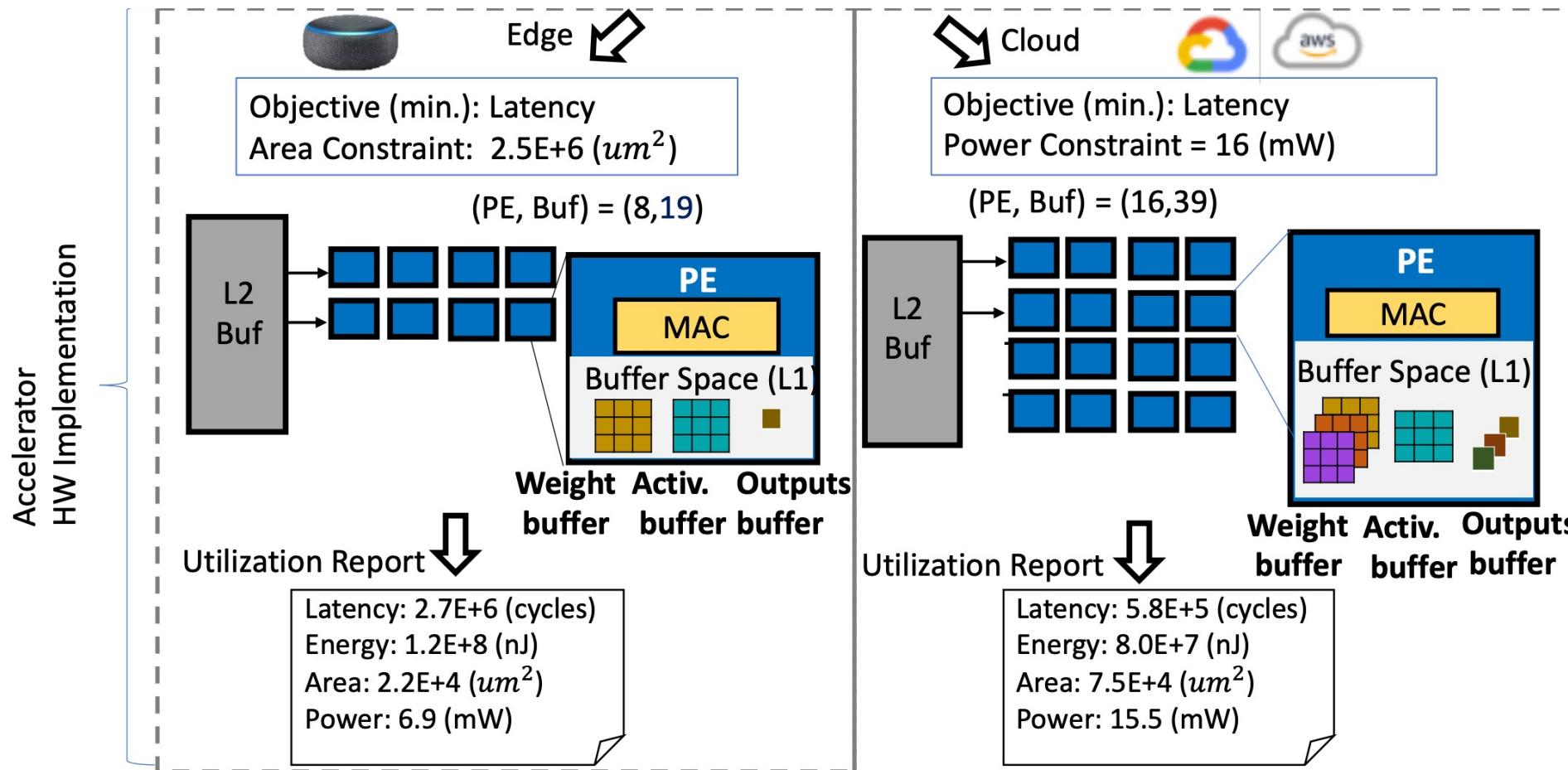
# 심층신경망 프로세서 설계 CHALLENGE

- NPU는 연산 성능을 높이면서 소모되는 전력을 줄이기 위한 방향으로 설계
  - 연산성능: PE의 개수, PE 하나당 구현 가능한 연산 성능
  - 메모리: 메모리 사이즈 및 구성, PE와 메모리 간의 대역폭



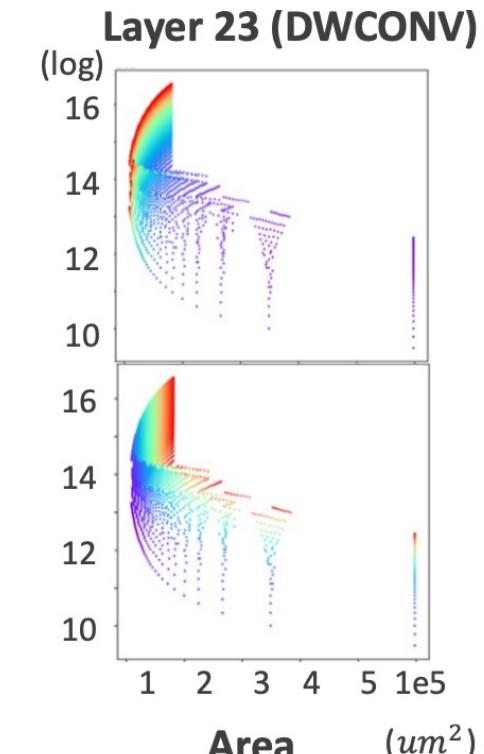
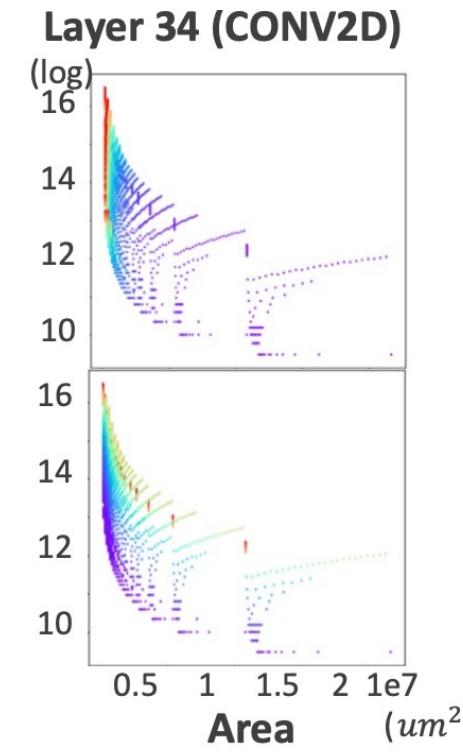
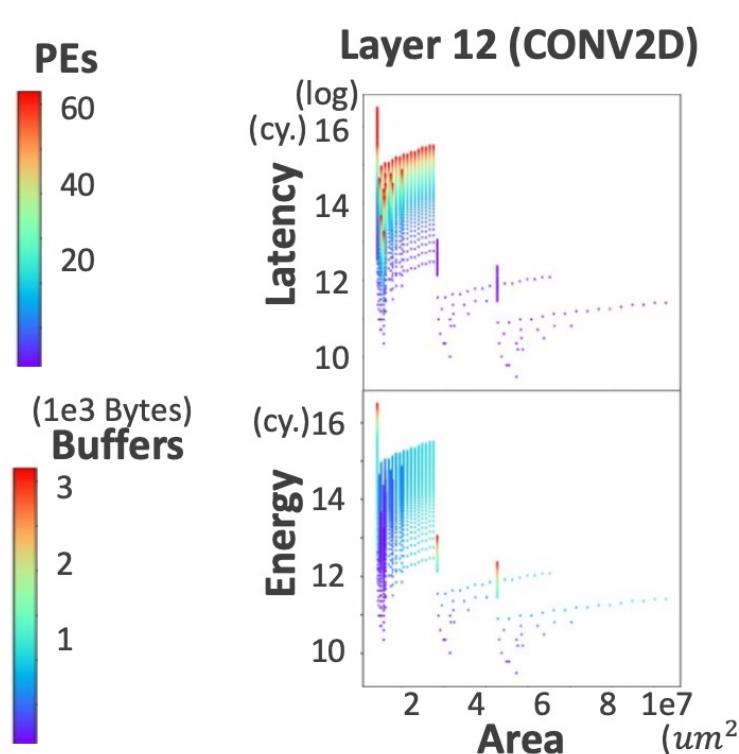
# 심층신경망 프로세서 설계 CHALLENGE

- NPU는 연산 성능을 높이면서 소모되는 전력을 줄이기 위한 방향으로 설계
  - 연산성능: PE의 개수, PE 하나당 구현 가능한 연산 성능
  - 메모리: 메모리 사이즈 및 구성, PE와 메모리 간의 대역폭



# 강화학습을 이용한 심층신경망 프로세서 설계 #1

- 심층신경망 가속을 위한 최적의 하드웨어 Resource Combination
  - PE의 개수와 Buffer의 크기가 Latency, Area, Power consumption에 영향
  - 기존의 설계 방법에서는 모든 경우를 구현하고 실제 성능 확인
  - 논문에서는 Simulator (MAESTRO)를 사용하여 Estimation



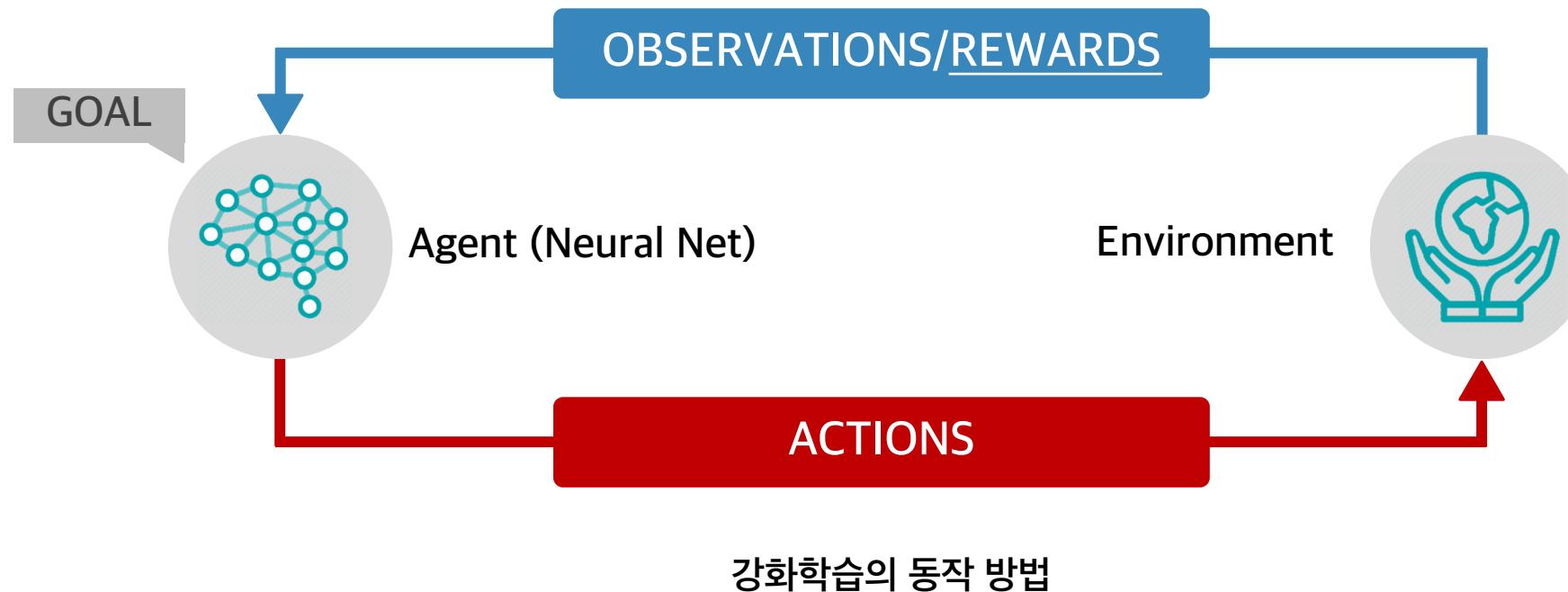
PE의 갯수, Buffer의 크기가 하드웨어 성능에 미치는 정도

Selected layer from MobileNet-V2

# 강화학습을 이용한 심층신경망 프로세서 설계 #2

## ■ 보상을 통해 학습하는 강화학습

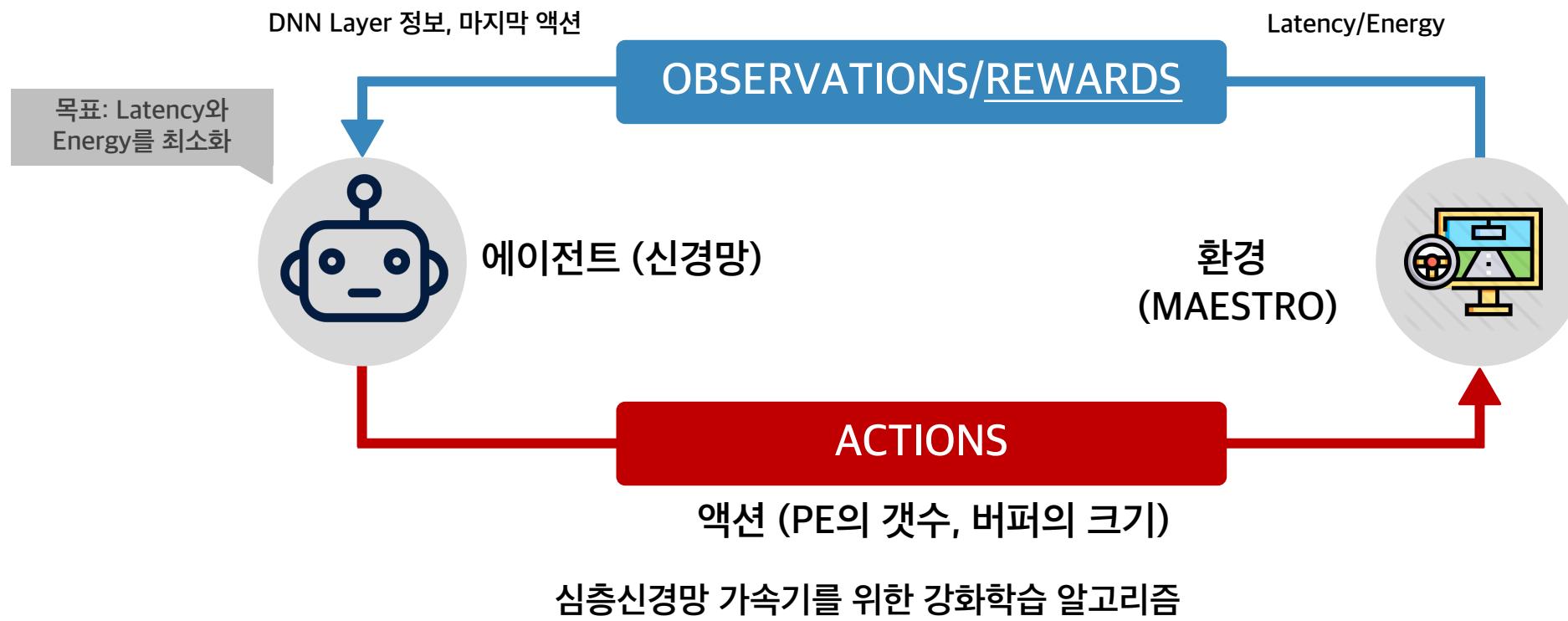
- 주어진 환경에서 에이전트가 특정 행동, 환경으로부터 얻은 보상을 바탕으로 보상을 최대화하는 행동을 학습
- 에이전트 (학습하는 시스템), 환경 (에이전트가 동작하는 환경), 보상 (행동의 좋고 나쁨을 알려주는 대상)
- 에이전트는 상태를 확인하고 정책에 따라 행동을 선택, 환경으로부터 보상을 받음





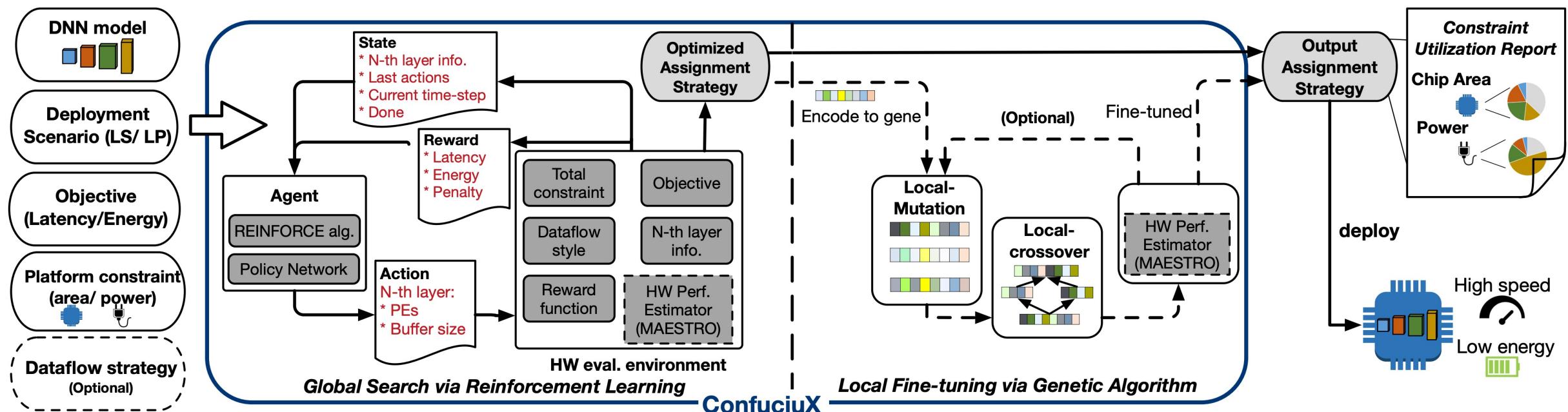
# 강화학습을 이용한 심층신경망 프로세서 설계 #2

- Latency와 Energy를 최소로 하는 강화학습 알고리즘
  - 제약사항: 칩의 Area와 Power는 적정 수준 이상이 되면 다시 학습!
  - 액션: PE의 갯수의 버퍼의 크기 (사용자가 사전에 정의한 Discrete한 값)
  - 환경: MAESTRO (심층신경망 시뮬레이터 프로그램)
  - 보상: Latency/Reward



# 강화학습을 이용한 심층신경망 프로세서 설계 #2

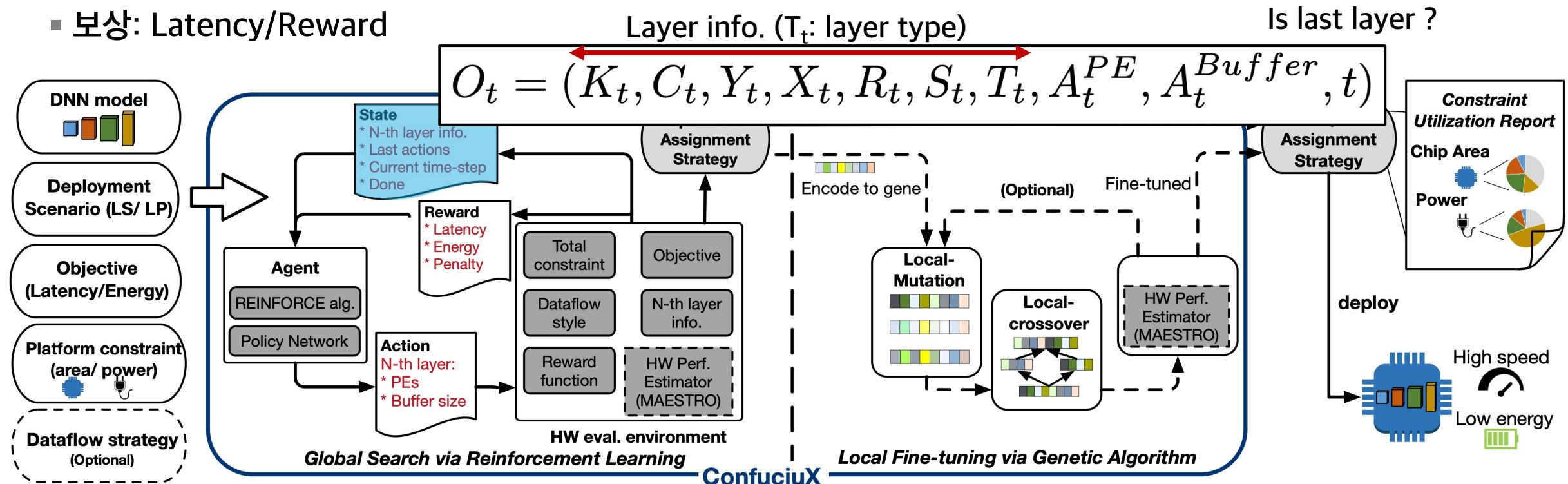
- Latency와 Energy를 최소로 하는 강화학습 알고리즘
  - 제약사항: 칩의 Area와 Power는 적정 수준 이상이 되면 다시 학습!
  - 액션: PE의 갯수와 버퍼의 크기 (사용자가 사전에 정의한 Discrete한 값)
  - 환경: MAESTRO (심층신경망 시뮬레이터 프로그램)
  - 보상: Latency/Reward



심층신경망 가속기를 위한 강화학습 알고리즘

# 강화학습을 이용한 심층신경망 프로세서 설계 #3

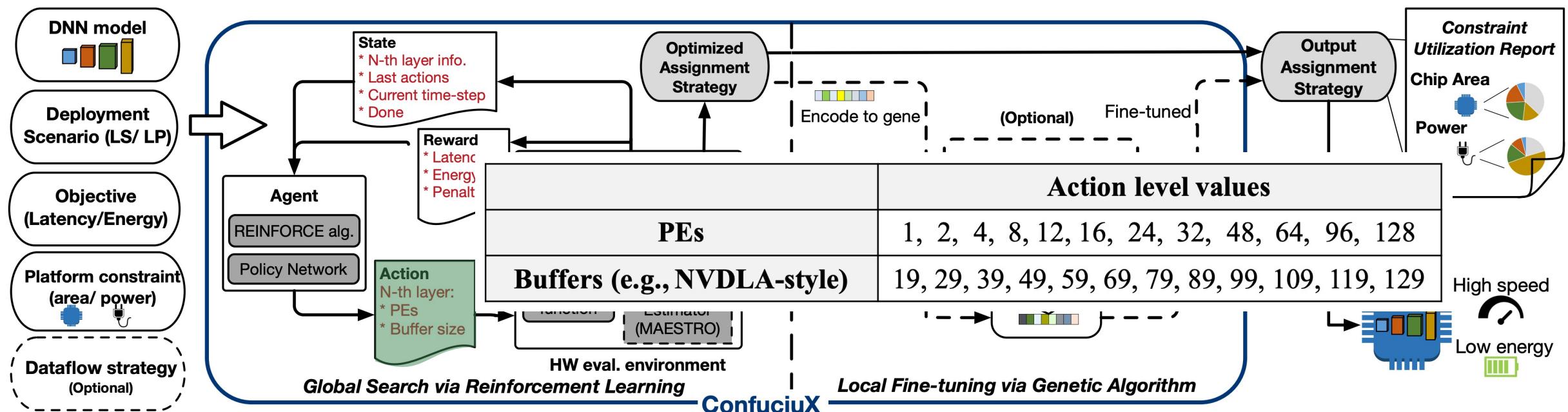
- Latency와 Energy를 최소로 하는 강화학습 알고리즘
  - 제약사항: 칩의 Area와 Power는 적정 수준 이상이 되면 다시 학습!
  - 액션: PE의 갯수의 버퍼의 크기 (사용자가 사전에 정의한 Discrete한 값)
  - 환경: MAESTRO (심층신경망 시뮬레이터 프로그램)
  - 보상: Latency/Reward



심층신경망 가속기를 위한 강화학습 알고리즘

# 강화학습을 이용한 심층신경망 프로세서 설계 #3

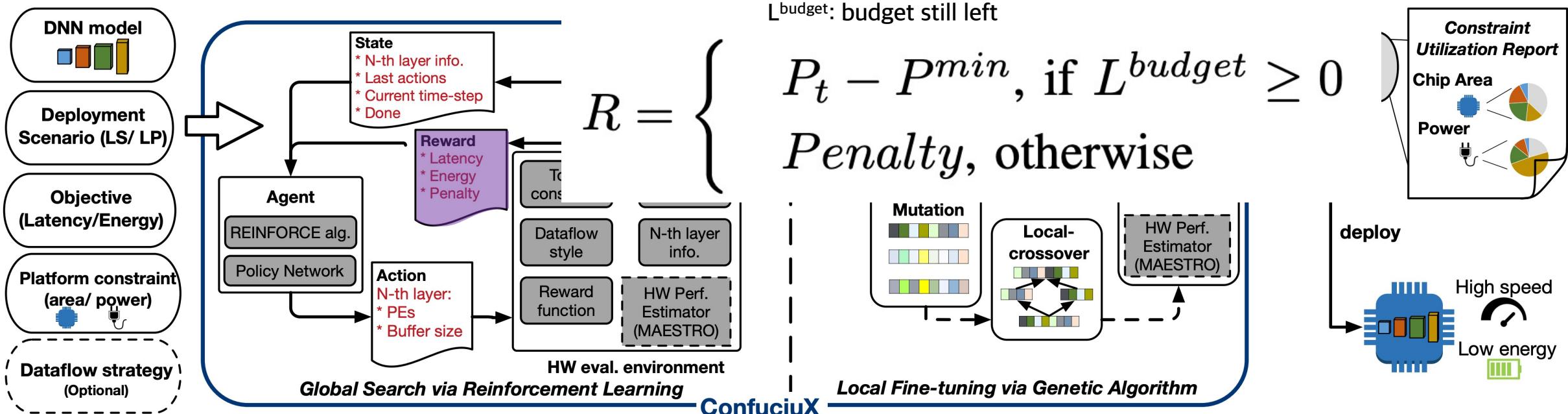
- Latency와 Energy를 최소로 하는 강화학습 알고리즘
  - 제약사항: 칩의 Area와 Power는 적정 수준 이상이 되면 다시 학습!
  - 액션: PE의 갯수의 버퍼의 크기 (사용자가 사전에 정의한 Discrete한 값)
  - 환경: MAESTRO (심층신경망 시뮬레이터 프로그램)
  - 보상: Latency/Reward



심층신경망 가속기를 위한 강화학습 알고리즘

# 강화학습을 이용한 심층신경망 프로세서 설계 #3

- Latency와 Energy를 최소로 하는 강화학습 알고리즘
  - 제약사항: 칩의 Area와 Power는 적정 수준 이상이 되면 다시 학습!
  - 액션: PE의 갯수와 버퍼의 크기 (사용자가 사전에 정의한 Discrete한 값)
  - 환경: MAESTRO (심층신경망 시뮬레이터 프로그램)
  - 보상: Latency/Reward



심층신경망 가속기를 위한 강화학습 알고리즘

# 실험결과

- Latency와 Energy를 최소로 하는 강화학습 알고리즘
  - 두 종류의 타겟 플랫폼으로 실험 진행 (Cloud FPGA, Edge FPGA)
    - Cloud FPGA (4,096PEs, 8KB), Edge FPGA (256PEs, 4KB)
  - DNN models (MobileNet v2, MNasNet, ResNet-50, GNMT, NCF)
  - Accelerator platforms (NVDLA-style, Eyeriss-style)
  - RL로 Global Search, GA알고리즘을 사용하여 Tuning

Objective: Latency		Baseline-dla			ConfuciuX-dla					
					1st: global search			2nd: local-finetuning		
Platform Constraint	Model	Used Cstr.		Optimized Results (cy.)	Used Cstr.		Optimized Results(cy.)	Used Cstr.		Optimized Results(cy.)
		PEs	Bufs		PEs	Bufs		PEs	Bufs	
Cloud FPGA Cstr: PE: 4096, Buf: 8KB	ResNet50	4081	7786	2.352E+08	4080	6338	<b>2.346E+08</b>	4096	7958	<b>2.2E+08</b>
	MbnetV2	4056	7716	2.5E+07	4032	3710	<b>2.3E+07</b>	4088	7912	<b>2.0E+07</b>
Edge FPGA Cstr: PE: 256, Buf: 4KB	ResNet50	212	3206	1.8E+09	256	2290	<b>1.5E+09</b>	256	3962	<b>1.2E+09</b>
	MbnetV2	208	3356	1.13E+08	256	2468	<b>1.08E+08</b>	256	3838	<b>6.9E+07</b>

**감사합니다**