Chapter 01

AI 반도체 혁신을 위한 시스템 아키텍처 기반 소프트웨어 기술 전략

김진미_한국전자통신연구원 책임연구원
고광원_한국전자통신연구원 책임연구원
김강호_한국전자통신연구원 책임연구원

AI 반도체 산업은 칩 성능 중심의 경쟁에서, 하드웨어와 소프트웨어를 통합적으로 설계하는 아키텍처 경쟁으로 전환되고 있다. 이 과정에서 소프트웨어는 단순한 개발 도구를 넘어, 하드웨어의 잠재 성능을 구조화하고 실제 성능으로 구현하는 핵심 계층으로 작용한다. 본 고는 AI 반도체의 성능을 극대화하는 시스템 아키텍처 기반 소프트웨어 기술을 분석하고, 주요 글로벌 기업들의 전략과 기술 동향을 비교 · 정리하였다. AI 반도체 시대의 핵심 역량은 생산성 중심의 코딩 능력보다는, 하드웨어의 물리적 · 구조적 제약을 이해하고 이를 소프트웨어적으로 조직화하고 구조화할 수 있는 아키텍처 기반의 사고에 있다. 이러한 역량은 자동화된 코드 생성이나 단순한 개발 효율성으로 대체될 수 없으며, 컴퓨터 아키텍처 · 병렬 컴퓨팅 · 컴파일러 · 분산 시스템에 대한 통합적 이해를 요구한다. 미래 AI 반도체의 경쟁력은 하드웨어 성능 그 자체보다는, 이를 뒷받침하는 개발자 생태계와 시스템 아키텍처 기반의 소프트웨어 역량에 달려 있다. 이러한 역량의 격차가 향후 산업 경쟁 구도를 결정할 것이다.

I. 서론

인공지능(Artificial Intelligence: AI) 기술의 급속한 발전은 컴퓨팅 인프라 전반에 구조적 재편을 요구하고 있다. 대규모 언어 모델(Large Language Model: LLM), 생성형 이미지 · 영상 모델, 자율주행, 과학 계산 등 첨단 AI 응용은 과거와 비교할 수 없을 만큼 방대한 연산 성능과 메모리 대역폭 그리고 고효율 병렬 처리가 필요하다. 이러한 변화 속에서 AI 반도체는 더 이상 단순한 연산 가속 장치가 아니라, 국가와 기업의 AI 경쟁력을 좌우하는

* 본 내용은 김진미 책임연구원(☎ 042-860-4885, jinmee@etri.re.kr)에게 문의하시기 바랍니다.
** 본 내용은 필자의 주관적인 의견이며 IITP의 공식적인 입장이 아님을 밝힙니다.
***본 연구는 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임
(No. RS-2024-00460762, 패브릭 메모리 기반 연산 스토리지 시스템SW 핵심 기술 개발)

핵심 전략 인프라로 부상하고 있다[1][2].

초기 AI 반도체 경쟁은 이론적 연산 성능, 전력 효율 등 하드웨어 지표에 집중되었으나, 실제 산업 현장에서는 모델 호환성, 개발 생산성, 최적화 완성도, 운영 안정성 및 생태계 성숙도 등이 기술 채택을 결정하는 복합적인 요인으로 작용한다. 이는 하드웨어의 잠재 성능을 실질적인 성능으로 전환하는 핵심 동인이 소프트웨어에 있음을 보여준다[3].

이러한 변화는 AI 반도체 경쟁의 중심축이 하드웨어에서 소프트웨어, 특히 시스템 소프트웨어로 이동하고 있음을 의미한다. 컴파일러, 런타임, 드라이버, 라이브러리로 구성되는 시스템 소프트웨어는 하드웨어의 복잡성을 추상화하는 동시에, AI 워크로드 특성에 맞추어 연산 구조와 메모리 접근 방식을 재구성하고 병렬 실행을 정교하게 조율한다. 즉, 시스템 소프트웨어는 하드웨어의 물리적 한계를 극복하고 성능 효율을 극대화하는 핵심 계층이다[4]. AI 반도체 경쟁의 중심은 단순 칩 성능을 넘어 하드웨어와 소프트웨어의 통합 최적화 능력으로, 물리적 한계를 소프트웨어 설계로 극복하는 '아키텍처 사고'가 핵심 역량이 될 것이다.

본 고는 이에 기반하여 AI 반도체 혁신을 실질적으로 견인하는 시스템 소프트웨어 기술 동향을 분석한다. GPU(Graphics Processing Unit) · TPU(Tensor Processing Unit) · NPU(Neural Processing Unit) 환경을 중심으로 통합 아키텍처 관점의 기술 전략을 살펴보고, 시스템 소프트웨어가 산업 경쟁력의 핵심 변수로 작용하고 있음을 조명한다. II장에서는 AI 반도체 패러다임 전환의 구조적 배경과 시스템 소프트웨어의 필요성을 설명하고, III장에서는 주요 글로벌 및 국내 AI 반도체 사례를 통해 컴파일러 · 런타임 · 라이브러리 기술 동향을 비교한다. IV장에서는 생태계 확장을 위한 전략적 과제를 도출하며, 마지막 V장에서 결론과 향후 시사점을 제시한다.

II. AI 반도체 패러다임 전환과 시스템 소프트웨어의 부상

AI 반도체 산업의 변화는 단순한 하드웨어 세대교체가 아니라, 컴퓨팅 패러다임의 구조적 전환을 의미한다. 범용 CPU(Central Processing Unit) 중심의 컴퓨팅 모델에서 출발한 AI 연산은 GPU 기반의 대규모 병렬 처리로 이동하였으며, 최근에는 특정 AI 워크로드에 최적화된 가속기 중심 구조로 재편되고 있다. 이러한 플랫폼과 인프라의 변화는 자연스럽게 컴파일러와 런타임을 포함한 시스템 소프트웨어의 전략적 중요성을 부각하고 있다. 하드웨어의

구조가 복잡해질수록, 이를 추상화하고 통합하는 소프트웨어 계층의 역할은 더 결정적이기 때문이다.

1. CPU에서 GPU, NPU로의 구조적 진화

AI 반도체는 CPU 중심의 범용 구조에서 GPU 기반 병렬 연산 구조를 거쳐, AI 특화 가속기로 진화하고 있다. GPU의 발전은 A100 세대의 아키텍처적 전환을 거쳐, Hopper의 혼합 정밀 및 메모리/실행 구조 고도화로 이어졌으며, Blackwell은 초대규모 모델을 염두에 둔 확장형 아키텍처를 제시한다[5]-[7].

CPU는 범용 연산에 적합하지만, AI의 대규모 행렬 · 텐서 연산에는 한계가 있으며, GPU는 대규모 병렬 처리로 이를 극복해 AI의 표준 인프라로 자리잡았다. 그러나 초대규모 모델 확산으로 메모리, 통신, 전력, 확장성 등의 한계로 TPU · NPU와 같은 AI 연산 특화 아키텍처가 등장했다. 이에 따라 하드웨어의 발전뿐 아니라 이를 통합 · 제어하는 시스템 소프트웨어의 중요성이 더욱 커지고 있다[6][7].

2. AI 워크로드 특성 변화와 소프트웨어 요구사항

AI 워크로드는 학습, 추론, 분산 학습 환경으로 구분되며, 각기 다른 기술적 요구를 한다. 특히, 학습 단계에서는 대규모 행렬 연산과 역전파 계산이 반복적으로 수행되며, FP32, FP16, BF16, FP8 등 다양한 정밀도가 혼합적으로 활용된다. 이때 성능은 단순한 연산량보다 메모리 대역폭 활용, 데이터 재사용, 커널 스케줄링, 병렬성 극대화와 같은 요소에 의해 좌우된다. 따라서 동일한 하드웨어 환경에서도 연산 병합 및 스케줄링 등 컴파일러 최적화 수준에 따라 실제 성능은 크게 달라질 수 있다[3][4].

추론 단계에서는 저지연 응답과 전력 효율이 핵심 지표가 된다. 특히, LLM 서빙에서는 메모리 관리와 배치 처리 전략이 서비스 품질을 좌우하며, PagedAttention 기반 최적화가 대표적이다. 또한, 어텐션 연산의 메모리 병목을 줄이기 위한 FlashAttention과 같은 입출력 인식(IO-aware) 최적화 기법도 핵심으로 부상하고 있다[8].

분산 학습 환경에서는 다수의 가속기를 연결하는 통신 스택과 병렬화 전략이 필수적이며, 데이터 · 모델 · 파이프라인 병렬은 컴파일러 및 런타임과 긴밀히 연동된다. 특히, 파이프라인

병렬은 메모리 효율성과 확장성을 동시에 확보하는 핵심 기법이다. 이러한 복잡성 증가는 시스템 소프트웨어 설계 역량의 중요성을 더욱 부각시킨다[9].

3. AI 반도체 지향 시스템 소프트웨어의 전략적 중요성

이와 같은 환경 변화는 AI 반도체 경쟁의 본질을 재정의한다. 하드웨어 세대가 변화하더라도 일관된 개발 환경을 제공하고, 새로운 모델과 알고리즘을 신속히 수용할 수 있는 유연성을 확보하는 것은 시스템 소프트웨어의 역할이다. 컴파일러, 런타임, 라이브러리 계층은 하드웨어 성능을 현실의 처리 성능으로 구현하는 핵심 계층이다. 하드웨어가 성능의 상한선을 정의한다면, 시스템 소프트웨어는 그 상한선을 현실의 산업 경쟁력으로 구현하는 결정적 변수이다.

III. GPU · TPU · NPU 기반 시스템 소프트웨어 기술

AI 반도체 혁신은 물리적인 하드웨어 성능을 넘어, 시스템 소프트웨어를 통해 연산 자원을 효율화하고 실제 운용 구조를 재정의하는 소프트웨어적 설계 역량에서 완성된다. 동일한 하드웨어라도 컴파일러와 런타임의 설계 수준에 따라 실제 성능과 확장성은 크게 달라진다. 결국, 소프트웨어 스택의 완성도가 플랫폼 경쟁력을 좌우한다.

1. AI 반도체 소프트웨어의 계층적 스택 구조

AI 반도체 소프트웨어는 일반적으로 [그림 1]과 같이 하드웨어(L0)에서부터 애플리케이션(L5) 계층으로 구성되며, 아래에서 위로 올라갈수록 추상화 수준이 높아진다. 최하위 계층인 L0(하드웨어)는 실제 연산을 수행하는 AI 가속기 하드웨어이다. GPU, TPU, NPU 등이 여기에 해당하며, 연산 유닛 구조, 온칩 · 오프칩 메모리 계층, 인터커넥트 구조가 전체 시스템 성능의 물리적 상한선을 결정한다[5]-[7]. 이는 연산 처리량, 메모리 대역폭, 통신 지연과 같은 물리적 제약을 규정하는 계층이다. L1(하드웨어 추상화)는 하드웨어 자원을 소프트웨어가 활용할 수 있도록 추상화하는 계층으로 메모리 관리, 명령 큐 제어, 디바이스 동기화, 오류 처리

〈자료〉 한국전자통신연구원 자체 작성

[그림 1] AI 반도체 기반 소프트웨어 스택 및 사례

등을 담당한다. 가상 명령 체계, 메모리 모델, 인터커넥트 규격이 이 계층과 밀접하게 연결되어 하드웨어의 기능을 소프트웨어 인터페이스로 노출하여 통제한다[5]-[7].

L2(최적화 라이브러리)는 하드웨어 특성에 맞춰 반복적으로 사용되는 핵심 연산을 최적화된 형태로 제공하는 고성능 연산 커널 계층이다. 이 계층에는 수치 연산 라이브러리와 집단 통신 라이브러리가 포함되며, 엔비디아(NVIDIA) GPU에서 널리 활용되는 FlashAttention, Transformer Engine, FasterTransformer와 같은 특화 커널도 함께 속한다[8].

L3(런타임 및 컴파일러)는 AI 반도체 아키텍처 기반 소프트웨어 스택의 핵심 계층이다. 컴파일러인 구글 XLA(eXecutable Linear Algebra), 아파치 TVM(Tensor Virtual Machine), OpenAI의 GPU 커널 컴파일러 Triton, MLIR, LLVM Backend 등은 연산 그래프를 분석하여 커널을 생성하고 메모리 레이아웃을 최적화한다. 이 과정에서 타일링 전략, 메모리 배치, 병렬화 방식, 연산 병합과 같은 핵심 구조가 컴파일 단계에서 결정된다. TensorRT, ONNX (Open Neural Network Exchange) Runtime, vLLM과 같은 런타임은 실행 스케줄링, 메모리 재사용, 디바이스 간 통신을 조율하는 역할을 수행한다. 결국 정적 최적화를 담당하는 컴파일러와 동적 자원 관리를 수행하는 런타임이 긴밀히 결합될 때, 하드웨어의 잠재 성능이 실제 성능으로, 효과적으로 전환된다[10]-[12].

L4(AI 프레임워크) 계층에는 PyTorch, TensorFlow, JAX 등 AI 프레임워크가 위치하며, 개발자는 이 단계에서 모델을 정의하고 학습 및 추론 로직을 구현한다.

최상위 L5(애플리케이션) 계층에는 ChatGPT, Gemini, Claude, Copilot, Stable Diffusion 과 같은 AI 서비스가 포함되며, 이 단계에서 사용자 경험과 서비스 품질이 결정된다. 따라서 하위 소프트웨어 스택의 완성도는 곧 서비스 경쟁력으로 직결된다.

2. GPU: 수직 통합 기반의 범용 플랫폼 전략

CUDA(Compute Unified Device Architecture) 스택은 [그림 2]와 같이 GPU 하드웨어와 시스템 소프트웨어가 긴밀히 결합한 수직 통합 구조이다. GPU 전략은 범용성을 유지하면서도, 컴파일러와 런타임을 통해 워크로드를 하드웨어 친화적으로 재구성하는 것에 초점을 둔다. CUDA 스택은 학습과 추론을 동일한 플랫폼 위에서 지원하며, 분산 학습 프레임워크와의 긴밀한 연계를 통해 데이터센터급 확장성을 확보하고 있다. 라이브러리, 통신 스택, 개발 도구가 유기적으로 통합되어 있기 때문에, 개발자는 환경 전환 없이 모델 규모를 확장할 수 있다. 이러한 생태계 일관성은 기술 채택 비용을 낮추고 플랫폼 락인 효과를 강화하는

L5	**Application & Solutions**	Nvidia GPU 기반 AI 서비스	DGX Cloud, NVIDIA AI Enterprise, Omniverse, DRIVE (자율주행) Clara (헬스케어), Isaac (로보틱스)
L4	**AI Frameworks & Tools**	AI 프레임워크 통합 · 개발 도구: NVIDIA 주관 최적화 및 직접 제공 프레임워크 및 개발 도구	PyTorch (NGC 최적화), TensorFlow (NGC 최적화), JAX, NVIDIA Nsight, NGC Containers, Triton Inference Server, NeMo
L3	**Runtime & Compiler**	컴파일러 · 런타임 · 추론 엔진: 연산 그래프 최적화, 커널 생성, 실행 스케줄링 담당	TensorRT, NVCC, NVRTC, PTX JIT Compiler, CUDA Runtime API, CUDA Graphs, cuDLA
L2	**Optimized Libraries**	고성능 최적화 라이브러리: 핵심 연산을 GPU 아키텍처에 최적화하여 제공	cuDNN, cuBLAS, CUTLASS, NCCL, cuFFT, cuSPARSE, cuRAND, Thrust, FasterTransformer, TensorRT-LLM
L1	**Driver & HW Abstraction**	하드웨어 추상화: 소프트웨어 접근 GPU 추상화	CUDA Driver API, PTX ISA, SASS, GPU Memory Model, NVLink Protocol, NVSwitch, GPUDirect RDMA, MIG
L0	**GPU Architecture**	GPU 마이크로아키텍처: 연산 유닛 및 메모리 구조	SM (Streaming Multiprocessor), Tensor Core, RT Core, L1/L2 Cache, HBM (High Bandwidth Memory), Warp Scheduler
	Hardware	GPU 제품 라인업	Blackwell (B200, GB200), Hopper (H100, H200), Ada Lovelace (L40S), DGX B200, HGX, Grace Hopper Superchip

〈자료〉 한국전자통신연구원 자체 작성

[그림 2] NVIDIA CUDA 컴퓨팅 플랫폼 스택

전략적 요소로 작용한다. 결과적으로 GPU 환경의 소프트웨어 전략은 개별 구성 요소의 우수성에 있지 않다. 그것은 하드웨어 설계와 시스템 소프트웨어를 통합적으로 조율하는 아키텍처 역량 그리고 이를 장기간 축적해 온 생태계의 성숙도에 있다. CUDA는 단순한 개발 도구가 아니라, GPU를 중심으로 한 통합 컴퓨팅 플랫폼 전략의 구현체라 할 수 있다[5]-[7][10].

3. TPU: 컴파일러 중심의 전역 최적화 전략

TPU 구조에서는 [그림 3]과 같이 하드웨어 제어의 세밀함보다 컴파일러 기반 전역 최적화에 초점을 둔다. 구글 TPU 스택의 핵심 특징은 컴파일러 중심 설계로 GSPMD(General and Scalable Parallelization for ML Computation Graphs) 기반의 자동 병렬화 그리고 드라이버와 런타임(libtpu) 등 핵심 구성 요소가 비공개인 폐쇄적 생태계를 기반으로 한다. XLA를 중심으로 연산 그래프를 전체 단위에서 분석 · 변환하여 실행 구조를 결정하며, 개발자가 저수준 최적화를 직접 수행하지 않아도 되도록 설계되었다. 이는 하드웨어 특화 구조와 강하게 결합한 자동 최적화 전략이다. GPU가 범용 생태계 확장을 전략으로 삼는다면, TPU는 구글 클라우드 내 효율성과 최적화를 중시하는 전략을 취한다[11].

L5	**Application & Solutions**	Google TPU 기반 AI 서비스 및 클라우드 인프라	Gemini, Google Search AI, Google Translate, Vertex AI, Google Cloud TPU API, Imagen
L4	**Frameworks & Tools**	TPU에 최적화된 프레임워크 및 대규모 학습 인프라	JAX, TensorFlow, PyTorch/XLA, Flax, Optax, T5X, Pax, MaxText
L3	**Runtime & Compiler**	연산 그래프를 TPU 하드웨어에 최적화된 실행 코드로 변환하는 핵심 계층	XLA (Accelerated Linear Algebra), GSPMD (자동 병렬화), Pallas (커스텀 커널), Mosaic (저수준 커널 컴파일러), HLO (High Level Operations IR), StableHLO
L2	**Libraries**	연산 라이브러리: TPU 하드웨어 최적화 핵심 연산 구현, XLA 자동 생성	XLA 내장 커널 (GEMM, Conv, Attention), libtpu, TPU Embeddings API, ICI 통신 라이브러리 (All-Reduce, All-Gather), Megascale XLA Collectives
L1	**Driver & HW Abstraction**	하드웨어 추상화(비공개): 소프트웨어 접근 TPU 추상화	TPU Runtime (libtpu), TPU Driver, TPU Memory Model, ICI (Inter-Chip Interconnect), TPU Pod 네트워크 추상화, TPU VM Interface
L0	**TPU Architecture**	TPU 마이크로아키텍처: AI 연산 특화 연산 유닛 및 메모리 구조	MXU(Matrix Multiply Unit), SparseCore, HBM(High Bandwidth Memory), VMEM(Vector Memory), SMEM(Scalar Memory), ICI Interconnect Fabric
	Hardware	TPU 제품 라인업	TPU v6e (Trillium), TPU v5p, TPU v5e, TPU v4, TPU Pod (v4: 4096칩), Cloud TPU Multislice

〈자료〉 한국전자통신연구원 자체 작성

[그림 3] 구글 TPU 기반 소프트웨어 플랫폼 스택

4. NPU: 워크로드 특화 및 선택적 생태계 연계 전략 및 시사점

NPU 환경에서는 기업별 하드웨어 구조와 목표 시장에 따라 특정 워크로드 중심의 차별화 전략이 전개되고 있다. 범용 플랫폼을 지향하기보다는 추론 특화, 저전력 환경, 초저지연 처리 등 명확한 적용 영역에서 경쟁 우위를 확보하는 방식이다. 이 과정에서 PyTorch, ONNX, vLLM 등 표준 인터페이스와의 호환성을 확보해 기존 생태계와의 연결성을 유지하면서도 독자적 컴파일러와 런타임을 통해 자사 하드웨어에 최적화된 실행 모델을 구축한다.

[표 1]의 국내외 사례를 종합하면, NPU의 시스템 소프트웨어 전략은 GPU와 동일한 범용 생태계를 재현하기보다는 특화된 실행 구조를 통해 차별화된 경쟁 구도를 형성하는 방향으로 전개되고 있음을 확인할 수 있다.

특히, 국외 NPU 기업들은 CUDA 중심 생태계와의 최소한의 호환성을 유지하면서도 정적 컴파일, 양자화, 메모리 배치 최적화 등 하드웨어 구조에 밀착된 소프트웨어 기술을 핵심

[표 1] 국내외 NPU 기업의 소프트웨어 스택 전략 비교

기업/NPU	목표	AI 프레임워크	컴파일러 및 런타임	전략/핵심 기능	장점	단점
(국내) 퓨리오사AI/ Warboy, Renegade	저전력 · 고효율 추론	PyTorch/ ONNX	Furiosa Compiler/ FuriosaRT	고효율 추론/그래프 최적화, INT8 중심 양자화 및 전력 효율 최적화, 다중 NPU 파이프라이닝	전력 효율 · 운용 현실성 중심	범용 학습 생태계 GPU 대비 제한, 워크로드 범위가 추론 중심으로 제한
(국내) 리벨리온/ ATOM, ION, REBEL	LLM 추론 특화	PyTorch/ TensorFlow/ ONNX (vLLM 연계)	RBLN 컴파일러/런타임	LLM 추론 특화/ PagedAttention, Continuous Batching, 멀티칩 스케줄링, 서빙 친화	GPU 중심 LLM 서빙 생태계(vLLM 등)와의 접점 확대, 메모리 효율 최적화	LLM 외 범용 워크로드 확대에 추가 생태계 필요
(국외) Cerebras/ WSE-3	초대규모 학습(웨이퍼 스케일)	PyTorch 등 → 단일 초거대 칩에 최적화된 실행 그래프	CGC (Cerebras Graph Compiler)	초대규모 학습/정적 병렬화·통신 · 배치 결정, 런타임 복잡도 최소화	거대 온칩 메모리 기반 분산 없이 대규모 학습 지향, 통신 병목 구조 회피	범용성 제한적, 특정 HW 아키텍처 종속, 투자/운영 진입장벽
(국외) Graphcore/ IPU	새로운 병렬 계산 모델 제시	PyTorch/ TensorFlow → Poplar	Poplar SDK(컴파일러+런타임)	정밀 병렬 제어/ 정적 스케줄링, 온칩 메모리 활용 극대화	예측 가능한 성능, 특정 워크로드에서 효율 · 제어력 강화	CUDA 생태계 대비 커뮤니티 · 호환성 약함, 워크로드 확장성 제약
(국외) Groq/LPU	초저지연 추론(결정적 실행)	모델 → 정적 컴파일	Groq Compiler+ 런타임	초저지연 추론/ deterministic execution, 지연 · 일관성 최적화	일관된 저지연 응답, 결정적 예측 가능, 실시간 추론 특화	범용성 부족, 확장성 제약, 학습 시장 영향력 제한적

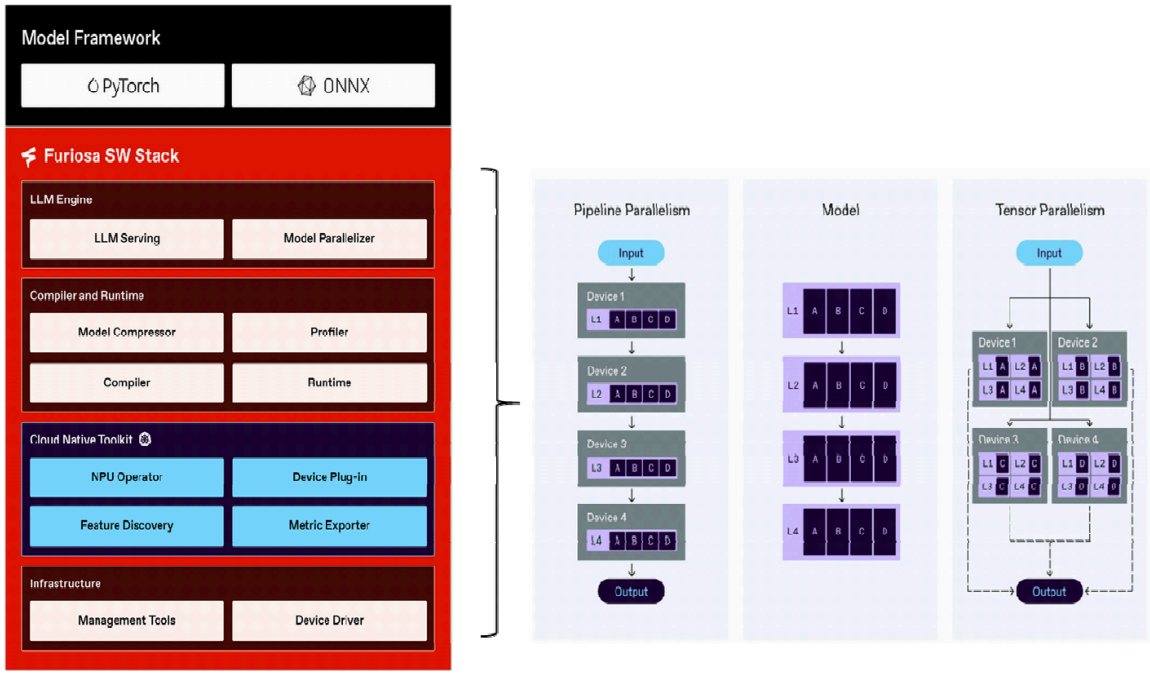
〈자료〉 한국전자통신연구원 자체 작성

경쟁력으로 내세우고 있다[12]-[17].

국내 AI 반도체 산업 역시 단순한 칩 성능 향상이나 공정 경쟁에 머무르기보다, 컴파일러 · 런타임 · 최적화 도구를 포함한 시스템 소프트웨어 역량을 전략적으로 축적해야 한다. 특히, 표준 프레임워크와의 안정적 연계, LLM 추론 환경과의 호환성 확보, 분산 실행 및 메모리 최적화 기술 고도화는 필수 과제로 자리 잡고 있다.

퓨리오사AI의 소프트웨어 스택 [그림 4]는 대규모 모델 처리를 위한 병렬 처리와 분산 실행 기법을 기반으로 한다. 여기에 더해 모델 압축, KV 캐시 최적화, 배치 처리, 하드웨어 맞춤형 코드 생성 및 스케줄링 등 다양한 최적화 기술을 적용하여 소프트웨어와 하드웨어 전반을 유기적으로 통합한다. 이를 통해 확장성과 비용 효율성을 동시에 갖춘 AI 추론 시스템을 구축하고 운영한다. 리벨리온의 소프트웨어 스택은 기존 코드를 수정하지 않고도 다양한 AI 모델을 원활하게 실행할 수 있다. 특히, 효율적인 컴파일과 런타임 최적화를 통해 연산 속도를 높이고 전력 효율을 극대화하는 고성능 솔루션을 지향한다.

결국, NPU 경쟁력은 하드웨어 설계 능력과 더불어, 해당 하드웨어의 구조적 특성을 소프



〈자료〉 FuriosaAI Blog, “FuriosaAI’s Software Stack”, Furiosa Docs, 2026.

[그림 4] 퓨리오사AI 소프트웨어 스택

트웨어적으로 조직화할 수 있는 아키텍처 설계 역량에 달려 있다. 이는 향후 AI 반도체 산업의 경쟁 구도가 칩 자체의 성능이 아니라, 시스템 소프트웨어를 포함한 통합 아키텍처 설계 능력에 의해 결정될 것임을 시사한다.

Ⅳ. AI 반도체 기반 소프트웨어 발전 전략

AI 반도체 산업의 경쟁 구도가 하드웨어 중심에서 통합 아키텍처 중심으로 전환됨에 따라 소프트웨어 전략은 선택적 보완 요소가 아니라 산업 경쟁력을 좌우하는 핵심 축으로 자리 잡고 있다. 특히, GPU 중심의 CUDA 생태계가 장기간 축적한 소프트웨어 자산을 기반으로 시장을 지배하고 있는 상황에서, 후발 주자 및 NPU 기반 플랫폼은 단순한 하드웨어 성능 개선만으로는 경쟁 우위를 확보하기 어렵다. 따라서 성능 최적화를 넘어 생태계 설계와 아키텍처 주도형 소프트웨어 역량 확보를 중심으로 수립되어야 한다.

1. 컴파일러 중심 아키텍처 전략

AI 반도체 시대의 소프트웨어 경쟁력은 컴파일러 기술에 있다고 해도 과언이 아니다. 컴파일러는 단순한 코드 번역 도구가 아니라, 하드웨어 구조와 AI 워크로드 특성을 동시에 고려하여 연산 그래프를 재구성하는 핵심 설계 계층이다. 연산 병합(fusion), 정밀도 변환, 메모리 레이아웃 최적화, 자동 병렬화, 커널 생성 및 스케줄링과 같은 기능은 모두 컴파일러 수준에서 구현되며, 이러한 최적화 수준에 따라 동일한 하드웨어에서도 실제 성능 차이가 크게 발생한다.

최근의 초거대 모델 환경에서는 연산량 자체보다 메모리 접근 패턴과 데이터 이동 비용이 시스템 성능을 결정하는 핵심 요소로 작용한다. 이때 컴파일러는 전역 그래프 분석을 통해 불필요한 연산을 제거하고, 메모리 재사용을 극대화하며, 하드웨어 자원에 맞는 실행 경로를 생성해야 한다. 따라서 MLIR, XLA, TVM과 같은 중간 표현(Intermediate Representation: IR) 기반의 모듈형 컴파일러 구조를 확보하고, 정적·동적 최적화를 결합한 고도화 전략을 추진하는 것이 필수적이다.

컴파일러는 하드웨어와 소프트웨어를 연결하는 아키텍처 중심 계층이며, 이 영역의 성숙

도가 곧 AI 반도체 플랫폼의 경쟁력을 결정한다. 특히, 국내 AI 반도체 기업의 경우, ONNX 기반 단순 변환 수준을 넘어 자체 IR 설계 역량과 그래프 수준 전역 최적화 기술 확보가 필수적이다.

2. 런타임 및 분산 스케줄링 역량 강화

초대규모 모델 환경에서 성능을 결정하는 요소는 단일 칩 성능이 아니라, 다중 가속기 간의 확장성이다. 이에 따라 런타임 계층은 단순한 실행 관리 기능을 넘어, 시스템 전반의 자원 배분과 성능 최적화를 책임지는 전략적 요소로 작용한다.

대규모 언어 모델 추론 환경에서는 동적 배치, 메모리 풀 관리, 연산과 통신의 중첩 실행, 파이프라인 병렬화 기술이 실제 서비스 품질을 좌우한다. 이러한 기술은 하드웨어 사양과는 별개로 소프트웨어 설계 역량에 의해 구현되며, 플랫폼의 확장성과 직결된다. 실제로 국내 NPU 기업들 역시 LLM 추론 최적화와 멀티칩 스케줄링을 중심으로 소프트웨어 전략을 전개하고 있으며, 이는 기존 GPU 기반 생태계와의 접점을 확보하기 위한 현실적 접근으로 볼 수 있다.

분산 학습 환경에서도 데이터 병렬, 모델 병렬, 파이프라인 병렬 전략은 컴파일러 및 런타임 설계와 긴밀히 연동되며, 통신 오버헤드를 최소화하는 알고리즘이 전체 시스템 효율을 결정한다. 따라서 런타임 및 스케줄링 기술은 단순한 부가 기능이 아니라, 통합 아키텍처의 완성도를 좌우하는 핵심 역량이다.

3. 표준 생태계와의 전략적 정렬

AI 반도체가 기존 CUDA와 같은 독자 생태계를 완전히 구축하는 것은 현실적으로 매우 어렵다. 독자 생태계 구축이라는 장기 목표와 함께 단기적으로는 기존 표준 생태계와의 전략적 정렬을 병행해야 한다. CUDA가 강력한 이유는 PyTorch · TensorFlow · ONNX 등 주요 프레임워크와의 긴밀한 통합 그리고 방대한 개발자 기반에 있다. 따라서 새로운 AI 반도체 플랫폼은 PyTorch 완전 호환성 확보, ONNX 기반 변환 안정성, HuggingFace 모델 지원, vLLM 및 분산 학습 프레임워크와의 연계, Kubernetes 기반 서빙 환경 통합 등을 통해 기존 워크플로우와의 마찰을 최소화해야 한다. 개발자가 새로운 하드웨어를 도입하는 과정

에서 발생하는 전환 비용을 낮추는 것이 시장 확산의 핵심 조건이기 때문이다. 이와 동시에, 소프트웨어 개발 도구와 문서화 체계, 디버깅 및 프로파일링 환경의 완성도는 개발자 경험을 좌우하는 중요한 요소이다. 생태계 전략은 단순한 기술 연계가 아니라, 장기적 신뢰와 지속적 개선 체계를 포함하는 플랫폼 전략으로 접근되어야 한다.

4. 하드웨어-소프트웨어 공동 설계(Co-Design) 체계 확립

AI 반도체 소프트웨어 전략의 핵심은 하드웨어와 소프트웨어의 분리적 개발이 아니라, 설계 단계부터의 공동 최적화이다. 설계 단계에서부터 컴파일러 요구사항, 메모리 접근 패턴, 데이터 흐름 구조를 함께 고려하는 공동 설계 체계가 구축되어야 한다. 예를 들어, ISA 설계 시 컴파일러 최적화 가능성을 반영하고, 메모리 계층 구조 설계 시 데이터 재사용 전략을 함께 고려하는 접근은 전체 시스템 효율을 획기적으로 향상할 수 있다. 인터커넥트 설계와 통신 라이브러리 개발 역시 병행되어야 하며, 정밀도 지원 전략은 양자화 알고리즘과 동시에 설계되어야 한다. 이러한 공동 설계 체계는 단순한 협업 수준을 넘어, 시스템 아키텍트 중심의 조직 역량을 요구한다. 이는 AI 반도체 시스템 소프트웨어가 단순한 개발 영역이 아니라 설계 영역임을 보여준다.

V. 결론

AI 반도체 시대의 핵심 역량은 단순히 코드를 많이 작성하는 능력이 아니라, 하드웨어의 물리적 · 구조적 제약을 이해하고 이를 소프트웨어적으로 조직화할 수 있는 설계 역량에 있다. 결국, 소프트웨어 발전은 기술 개발을 넘어, 인재와 생태계를 설계하는 장기적 전략 과제이다.

본 고와 같이, AI 반도체 산업은 공정과 연산 성능 중심의 경쟁 단계를 지나, 하드웨어와 소프트웨어가 긴밀히 결합된 통합 아키텍처 경쟁으로 진입하고 있다. GPU · TPU · NPU 등 다양한 가속기가 공존하는 환경에서 차별화는 이론적 성능 수치가 아니라, 컴파일러 · 런타임 · 라이브러리로 구성된 시스템 소프트웨어 스택의 완성도와 생태계 성숙도에 있다. 하드웨어가 성능의 상한선을 규정한다면, 소프트웨어는 그 상한선을 실제 산업 현장에서 구현이 가능한 경쟁력으로 전환하는 결정적 계층이다.

특히, AI 워크로드의 복잡성과 규모가 급격히 증가하는 상황에서, 시스템 소프트웨어는 단순한 지원 기술이 아니라 하드웨어 구조를 실질적으로 규정하는 아키텍처 계층으로 작용한다. 컴파일러의 전역 최적화 전략, 런타임의 분산 스케줄링 기술, 표준 프레임워크와의 호환성 확보 여부는 곧 플랫폼의 확장성과 시장 채택 가능성을 좌우한다. 이는 AI 반도체의 경쟁 본질이 칩의 성능뿐만 아니라 아키텍처 설계 역량에 있음을 분명히 보여준다.

또한, 생성형 AI 확산으로 일반 소프트웨어 개발의 생산성이 비약적으로 향상되고 있는 현시점에서, AI 반도체 시스템 소프트웨어는 오히려 더욱 고도화된 전문성을 요구하는 영역으로 자리매김하고 있다. 이 분야는 자동화된 코드 생성이나 단순한 개발 효율성으로 완전하게 대체될 수 없는 설계 중심의 영역이며, 컴퓨터 아키텍처, 병렬 컴퓨팅, 컴파일러 기술, 분산 시스템에 대한 통합적 이해를 전제로 한다. 결국, AI 반도체 산업의 경쟁력은 얼마나 많은 개발자를 보유하고 있는가가 아니라, 얼마나 많은 시스템 아키텍트를 양성하고 확보하고 있는가에 의해 결정될 것이다. AI 반도체 혁신은 시스템 아키텍처 수준에서 하드웨어와 소프트웨어를 설계하고 통합하는 능력의 문제이며, 이 인식의 전환이야말로 향후 AI 반도체 산업의 방향을 결정짓는 출발점이 될 것이다.

참고문헌

[1] Deepak Narayanan et al., “Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM”, Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis(SC), Article No.58, 2021, pp.1-15.

[2] Jeff Rasley et al., “DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters”, Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining(KDD), 2020, pp.3505-3506.

[3] Chris Lattner et al., “MLIR: Scaling Compiler Infrastructure for Domain Specific Computation”, IEEE/ACM International Symposium on Code Generation and Optimization(CGO), 2021, pp.1-13.

[4] Tianqi Chen et al., “Learning to Optimize Tensor Programs”, Proceedings of the 32nd International Conference on Neural Information Processing Systems(NIPS), 2018, pp.3393-3404.

[5] Pudi Dhilleswararao et al., “Efficient Hardware Architectures for Accelerating Deep Neural Networks: Survey”, IEEE Access, Vol.10, 2022, pp.131788-131828.

[6] Weile Luo et al., “Benchmarking and Dissecting the Nvidia Hopper GPU Architecture”, IEEE International Parallel and Distributed Processing Symposium(IPDPS), 2024, pp.656-667.

[7] Aaron Jarmusch et al., “Dissecting the NVIDIA Blackwell Architecture with Microbenchmarks”,

2025, pp.1-11.
[8] Tri Dao et al., "FLASHATTENTION: fast and memory-efficient exact attention with IO-awareness", Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS), Article No.1189, 2022, pp.16344-16359.
[9] Deepak Narayanan et al., "Memory-Efficient Pipeline Parallel DNN Training", International Conference on Machine Learning(ICML), 2021, pp.6302-6313.
[10] Philippe Tillet et al., "Triton: an intermediate language and compiler for tiled neural network computations", Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages(MAPL), 2022. pp.10-19.
[11] Yuanzhong Xu et al., "GSPMD: General and Scalable Parallelization for ML Computation Graphs", 2021, pp.1-16.
[12] Woosuk Kwon et al., "Efficient Memory Management for Large Language Model Serving with PagedAttention", 2023, pp.1-16.
[13] Hanjoon Kim et al., "TCP: A Tensor Contraction Processor for AI Workloads Industrial Product", ACM/IEEE 51st Annual International Symposium on Computer Architecture(ISCA), 2024, pp.890-902.
[14] Rebellions Press Release, "Rebellions Debuts REBEL-Quad at Hot Chips 2025, breaking AI's Energy Tax with High-Performance Chiplet Innovation", Hot Chips Symposium, 2025.
[15] Zheng Xu et al., "WSC-LLM: Efficient LLM Service and Architecture Co-exploration for Wafer-scale Chips", Proceedings of the 52nd Annual International Symposium on Computer Architecture(ISCA), 2025, pp.1-17.
[16] Zhe Jia et al., "Dissecting the Graphcore IPU Architecture via Microbenchmarking", 2019, pp.1-91.
[17] Santosh Raghavan., "AI Inference Technology, Delivers More Energy Efficiency Than GPUs for AI Inference. And LPUs Are Faster Too. Here's Why", Groq Whitepaper, 2024, pp.1-4.