과목명

국문: 조선공학특론 (부제: “차세대 AI 컴퓨팅: 탈-CUDA 시대를 위한 시스템 설계”)

영문: Special Topics in Naval Architecture and Ocean Engineering (Subtitle: "Next-Generation AI Computing: System Design for the Post-CUDA Era")

교과목 번호	414.761
강좌번호	001
학점	3-3-0

대표 교수

성명	김태완 교수
이메일	taewan@snu.ac.kr
전화 번호	02-880-1434
homepage	조선해양공학과 딥러닝연구실 http://dllab.snu.ac.kr/

1) 교과목 해시 태그 (Course Hashtags)

AI시스템아키텍처` (AI System Architecture)
하드웨어소프트웨어공동설계` (Hardware-Software Co-design)
차세대컴퓨팅` (Next-Generation Computing)
LLM가속기` (LLM Accelerator)
컴파일러최적화` (Compiler Optimization)

2) 강좌 해시 태그 (Lecture Hashtags)

탈CUDA` (Post-CUDA)
에이전틱AI` (Agentic AI)
풀스택AI설계` (Full-Stack AI Design)

AI칩만들기` (Making AI Chips)
시스템의미래` (The Future of Systems)

1 수업 목표

본 강의는 단순한AI 모델 활용을 넘어, 차세대AI 시스템의 두뇌와 심장을 직접 설계하는 것을 목표로 합니다. 현재AI 기술의 정점인 에이전틱(Agentic) AI는 사고-도구호출-관찰로 이어지는 복잡한 추론 과정에서 기존 컴퓨팅 시스템의 한계를 여실히 드러내고 있습니다. 반복적인 추론 루프는'KV 캐시의 폭발적인 증가'를 유발하며, 이는NVIDIA의 GPU/CUDA 패러다임조차 감당하기 어려운 메모리 장벽과 비용 문제를 야기합니다. 이것이 바로 우리가 마주한 가장 뜨겁고 중요한 문제입니다. 본 강의는 이 문제를 해결하기 위해 기존의 틀을 깨는 근본적인 접근법을 탐구합니다. 우리는 더 이상 주어진 하드웨어 위에서 소프트웨어를 최적화하는 수준에 머무르지 않을 것입니다. 대신, 알고리즘, 모델 아키텍처, 그리고 하드웨어 시스템 전체를 아우르는'풀스택 공동 설계(Full-stack Co-design)'의 관점에서 문제의 핵심을 파고듭니다.

수강생들은 본 강의를 통해 다음의 핵심 역량을 갖추게 될 것입니다:

1) 문제 정의 능력: 에이전틱AI와 같은 최신 워크로드가 왜 기존 하드웨어를 한계에 부딪히게 하는지 시스템 수준에서 명확히 분석합니다.

2) 혁신적 알고리즘 설계: BitNet b1.58과 같은 극단적 양자화 모델을 넘어, '콜레스키 파라미터화(Cholesky Parameterization)'와 같은 구조화된 행렬(Structured Matrix)을 통해 선천적으로 효율적인 어텐션 메커니즘을 설계하는 수학적 원리를 탐구합니다.

3) 안정적인 훈련 방법론: 강화학습의 정수인 '신뢰 영역 최적화(Trust Region Optimization)'개념을 LLM 사전 훈련에 적용하여, 더 안정적이고 일반화 성능이 뛰어난 모델을 만드는 고급 최적화 이론을 학습합니다.

4) 풀스택 시스템 구현: 위 알고리즘에 최적화된 맞춤형 하드웨어(FPGA/ASIC) 아키텍처와, 그 잠재력을100% 끌어내는 최적화 컴파일러(LLVM/MLIR) 및SDK를 설계하는 전 과정을 실습을 통해 경험합니다.

궁극적으로 본 강의를 마친 수강생들은NVIDIA의CUDA 생태계 너머를 바라보며, 특정 문제를 해결하기 위한 독자적인AI 가속기 시스템을 처음부터 끝까지 구상하고 MVP를 개발할 수 있는, 시장에서 가장 필요로 하는 차세대AI 시스템 아키텍트로서의 역량을 확보하게 될 것입니다.

2 교재 및 참고 문헌

본 강의는 빠르게 발전하는 분야의 최신 기술을 다루므로 지정된 단일 교재는 없습니다. 대신, 컴퓨터 아키텍처와 컴파일러 분야의 고전적인 명저 두 권을 주교재로 삼아 이론적 기틀을 다지고, 각 주차별 주제에 맞춰 최신 핵심 논문들을 함께 깊이 있게 탐구할 것입니다.

1) 주교재 (Required Textbooks)

● Patterson, David A., and John L. Hennessy. Computer Architecture: A Quantitative Approach. 6th ed., Morgan Kaufmann, 2017.

* 컴퓨터 아키텍처 분야의 바이블입니다. 폰 노이만 구조부터 메모리 계층, 병렬 처리까지 AI 가속기 설계를 위한 하드웨어의 근본 원리를 이해하는 데 필수적인 교재입니다.

● Aho, Alfred V., et al. Compilers: Principles, Techniques, & Tools. 2nd ed., Pearson, 2006.

* '드래곤 북'으로 알려진 컴파일러 분야의 고전입니다. Part 3에서 다룰 LLVM/MLIR의 구조를 이해하고, 하드웨어의 성능을 극한으로 끌어내는 최적화 기법의 이론적 배경을 학습하는 데 사용됩니다.

2) 참고 문헌 (References)

Part 1: 패러다임의 전환 (Paradigm Shift)

● Vaswani, Ashish, et al. Attention is all you need." Advances in neural information processing systems 30 (2017).

* 현대 LLM 시대를 연 트랜스포머 아키텍처를 최초로 제안한 논문입니다.

● Ma, Shuming, et al. "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits." (2024).

* 1.58비트 LLM의 개념을 제시하고, '곱셈 없는' AI 연산의 가능성을 연 기념비적인 논문

입니다.

● Meng, Fanxu, et al. "TransMLA: Multi-Head Latent Attention Is All You Need." (2025).

* KV 캐시 문제를 해결하는 MLA 아키텍처와 기존 GQA 모델을 MLA로 전환하는 실용적인 프레임워크를 제안합니다.

● LangChain Official Blog. "LangChain Research Reveals AI Agents Face Bottlenecks in Tool Usage." (2025).

* 에이전틱 AI가 다중 도구 및 컨텍스트 환경에서 겪는 실제 성능 저하 문제를 분석한 보고서로, 새로운 시스템 설계의 필요성을 명확히 보여줍니다.[2]

Part 2: 하드웨어 엔진 설계 (Hardware Engine Design)

●[PIM] "PIM-AI: A Novel Architecture for High-Efficiency LLM Inference." (2024).

* 메모리 내 연산(PIM) 기술을 LLM 추론에 적용하여 TCO와 에너지 소비를 획기적으로 절감하는 최신 아키텍처를 제안합니다.[3]

●[CXL] "Exploring CXL-based KV Cache Storage for LLM Serving." Proceedings of the 2nd Conference on Machine Learning and Systems (2024).

* CXL을 활용하여 에이전틱 AI의 핵심 병목인 KV 캐시 문제를 해결하고, GPU 비용을 절감하는 구체적인 시스템 설계 및 ROI 분석을 제시합니다.

● "A 3.1 POp/s/W Fully Digital Hardware Accelerator for Ternary Neural Networks." IEEE Journal of Solid-State Circuits (2021).

* BitNet과 같은 3진 가중치 네트워크에 최적화된 하드웨어 가속기 설계 사례를 통해 에너지 효율을 극대화하는 방법을 탐구합니다.

Part 3: 소프트웨어 두뇌 설계 (Software Brain Design)

● [LLVM] Lattner, Chris, and Vikram Adve. "LLVM: A compilation framework for lifelong program analysis & transformation." Proceedings of the 2004 international symposium on Code generation and optimization (2004).

* 현대 컴파일러의 표준이 된 LLVM의 설계 철학과 모듈식 구조를 설명한 원 논문입니다.

● Lattner, Chris, et al. "MLIR: A compiler infrastructure for the end of Moore's law." (2020).

* 다양한 하드웨어 가속기를 지원하기 위한 차세대 컴파일러 인프라 MLIR의 비전과 구조를 제시합니다.

Part 4: 이론적 돌파구 및 심층 연구 (Theoretical Breakthroughs & Advanced Topics)

● Dao, Tri, et al. "Monarch: Expressive structured matrices for efficient and accurate training." International Conference on Machine Learning. PMLR, 2022.

* '콜레스키 파라미터화'와 같은 구조화된 행렬을 통해 딥러닝 모델의 효율성과 성능을 동시에 높이는 하드웨어 인식 알고리즘 설계의 대표적인 연구입니다.

● [Natural Policy Gradient] Kakade, Sham M. "A natural policy gradient." Advances in neural information processing systems 14 (2001).

* '신뢰 영역 최적화'의 이론적 토대가 된 자연 정책 경사법을 최초로 제안한 논문으로, 안정적인 모델 훈련의 수학적 원리를 다룹니다.

● Schulman, John, et al. "Trust region policy optimization." International conference on machine learning. PMLR, 2015.

* 자연 정책 경사법을 대규모 신경망에 적용 가능하도록 만든 TRPO 알고리즘을 제안하여, 단조적 개선을 보장하는 최적화의 길을 열었습니다.

● [Information Geometry] Amari, Shun-ichi, and Hiroshi Nagaoka. Methods of information geometry. Vol. 191. American Mathematical Soc., 2000.

* 자연 경사법의 이론적 배경이 되는 정보 기하학 분야의 필독서입니다. 파라미터 공간의 기하학적 구조를 이해하는 데 깊이를 더해줍니다.

3 강의 계획

주요 수업 방식: 플립러닝, 이론 위주 수업, 프로젝트 수업

Part 1: 패러다임의 전환 - 왜 새로운 시스템이 필요한가? (1-4주차)

● 1주차: AI 연산의 현주소와 한계

○ (화) 강의 소개 및 목표 설정. 폰 노이만 구조의 역사와 '메모리 장벽' 문제.

○ (목) 트랜스포머의 등장과 NVIDIA GPU/CUDA 생태계의 지배력 분석. LLM 추론의 TCO(총소유비용)와 전력 소모 문제 심층 분석.

● 2주차: 새로운 알고리즘의 등장: 효율성의 재정의

○ (화) 구조화된 어텐션 메커니즘: MLA(Multi-Head Latent Attention)와 TransMLA 심층 분석. KV 캐시 압축의 원리와 한계.

○ (목) BitNet b1.58 혁명: 1.58비트(삼진) 양자화의 원리와 '곱셈 없는' 산술 연산의 파괴적인 잠재력.

● 3주차: 현재 가장 뜨거운 문제 - 에이전틱 AI의 도전

○ (화) 에이전틱 AI의 작동 원리와 병목 현상: Think -> Act -> Observe 추론 루프와 'KV 캐시 폭발' 문제 분석. LangChain, AutoGen 등 현존 프레임워크의 시스템적 한계.

○ (목) 우리의 핵심 철학 - 풀스택 공동 설계: 소프트웨어 최적화(vLLM)의 한계. 에이전틱 AI 문제를 해결하기 위한 알고리즘-하드웨어 통합 솔루션의 필요성.

● 4주차: 이론적 돌파구 탐색

○ (화) 선천적으로 효율적인 아키텍처: '콜레스키 파라미터화'(K=LLT)를 이용한 어텐션 구조 설계. 구조적 제약이 모델 학습에 미치는 영향.

○ (목) 안정적인 훈련을 위한 고급 최적화: 신뢰 영역(Trust Region)과 PPO 개념을 LLM 사전 훈련에 적용하기. 단조적 개선(Monotonic Improvement) 보장의 의미와 중요성.

Part 2: 하드웨어 엔진 설계 - '에이전트 가속기'의 탄생 (5-8주차)

● 5주차: AI를 위한 디지털 논리 회로

○ (화) 논리 게이트 복습. 덧셈기와 곱셈기의 구조적 차이 (면적, 전력, 속도).

○ (목) 하드웨어 기술 언어(Verilog/SystemVerilog) 기초. [실습 1] 간단한 산술 논리 장치(ALU) 설계.

● 6주차: AI를 위한 컴퓨터 아키텍처

○ (화) 데이터플로우 아키텍처 vs. 제어 흐름 아키텍처.

○ (목) NPU 타일(Tile) 설계: MAC 유닛을 대체할 '삼진 산술 유닛(Ternary Arithmetic Unit, TAU)'의 개념 설계.

● 7주차: 메모리 시스템과 인터커넥트

○ (화) 메모리 계층 구조 (SRAM, DRAM, HBM). PIM(Processing-in-Memory)의 잠재력.

○ (목) 칩 내부의 고속도로: 온칩 네트워크(NoC)와 CXL을 활용한 '에이전틱 스크래치패드' 개념.

● 8주차: FPGA 프로토타이핑

○ (화) FPGA 아키텍처의 이해 (LUTs, BRAMs, DSPs).

○ (목) [실습 2] 6주차에 설계한 'TAU'를 Verilog로 코딩하고, FPGA에 합성(Synthesis)하여 리소스 사용량 분석.

Part 3: 소프트웨어 두뇌 설계 - CUDA 해자를 우회하라 (9-12주차)

● 9주차: 현대 컴파일러의 이해

○ (화) Chris Lattner의 철학: LLVM의 모듈식 구조 (프론트엔드, 옵티마이저, 백엔드).

○ (목) AI 시대를 위한 컴파일러: MLIR의 개념 (다계층 중간 표현, Dialect).

● 10주차: LLM 가속기 컴파일러 프론트엔드 설계

○ (화) 파이토치(PyTorch) 모델 그래프 분석 및 Torch-MLIR 활용법.

○ (목) LLM 가속기 커스텀 MLIR Dialect 정의: BitNet과 콜레스키 파라미터화 연산을 표현하는 우리만의 언어 만들기. [실습 3] LLM 가속기 Dialect 구현

● 11주차: 최적화의 기술과 시스템 소프트웨어

○ (화) 하드웨어 독립/종속 최적화: 연산자 융합(Operator Fusion), 타일링(Tiling) 전략.

○ (목) MLIR Dialect를 우리 NPU와 RISC-V 제어 코어를 위한 저수준 명령어로 변환(Lowering)하기. 디바이스 드라이버와 런타임(Runtime)의 역할.

● 12주차: 최종 사용자 경험 - SDK 설계

○ (화) 개발자 친화적인 API 설계: model.to('llm_core_ai')의 마법.

○ (목) [중간 프로젝트 발표]

Part 4: 종합 및 미래 비전 (13-15주차)

● 13주차: 심층 사례 연구

○ (화) 구글 TPU & XLA 분석: 성공적인 하드웨어-소프트웨어 공동 설계의 교훈.

○ (목) Groq LPU 분석: Tensor Streaming Processor 아키텍처와 Deterministic Execution 모델의 시사점.

● 14주차: 차세대 AI 시스템 종합

○ (화) 종합 설계: 에이전틱 AI 가속기: 4주차의 이론, 5-8주차의 하드웨어, 9-12주차의 소프트웨어를 결합하여 에이전틱 AI의 병목을 해결하는 풀스택 시스템 아키텍처 완성.

○ (목) PIM/CIM: 메모리와 연산의 경계가 허물어지는 미래.

● 15주차: 최종 프로젝트 발표 및 종합

○ (화) [최종 프로젝트 발표] LLM 가속기 시스템 전체 설계 제안.

○ (목) 강의 요약 및 AI 하드웨어의 미래.

4 평가 방법

성적부여 방식	절대 평가
등급제 여부	A~F
출석 규정	수업일수의 1/3을 초과하여 결석하면 성적은 "F" 또는 "U"가 됨. (학칙 85조) 결석에 대하여 교원에게 별도로 출석인정을 받은 경우 예외로 함.
기타 사항	부정 행위를 한 사실이 발견될 경우 "F" 학점 처리되고 수강생의 소속 학과장(또는 학부장)에게 보고함. 단, 특정 Homework, Programming Assignment, 프로젝트에 대하여 다른 수강생과 논의 또는 협력을 권장하는 경우는 부정 행위가 아님. 중간 고사 및 기말 고사가 open book으로 치러질 경우에는 부정 행위가 아님.

구분	출석	과제	중간	기말	수시 평가	태도	기타	합계
비율	10%	30%	20%	20%	0%	0%	20%	100%
비고		Homework 10회, Programming Assignment 5회	지필 논술형	지필 논술형			개인 프로젝트 (마지막 주 발표)	

5 정원외 신청

추가 수용 인원: 5명

6 수강생 참고 사항

1) 선수 이수 과목 (Prerequisites)

본 강의는 알고리즘, 소프트웨어, 하드웨어의 경계를 넘나드는 최첨단 주제를 다루므로, 여러 분야에 걸친 탄탄한 기초 지식을 필요로 합니다. 성공적인 수강을 위해 학생들은 다음 영역에 대한 이해를 갖추고 있어야 합니다.

필수 (Required):

● 컴퓨터 구조 (Computer Architecture): 프로세서 파이프라인, 캐시, 메모리 계층 구조 등 현대 컴퓨터 시스템의 작동 원리에 대한 깊이 있는 이해가 필수적입니다. 학부 수준의 '컴퓨터 구조' 과목 이수에 준하는 지식이 요구됩니다.

● 디지털 논리 회로 (Digital Logic Design): 논리 게이트, 조합 및 순차 회로, ALU 설계 등 하드웨어의 기본 구성 요소에 대한 지식이 필요합니다. Verilog 또는 VHDL과 같은 하드웨어 기술 언어(HDL)에 대한 경험이 있다면 큰 도움이 됩니다.

● 프로그래밍 및 자료구조 (Programming & Data Structures): Python에 능숙해야 하며, C++에 대한 기본적인 이해가 있으면 좋습니다. 알고리즘 및 자료구조에 대한 탄탄한 지식은 컴파일러 최적화를 이해하는 데 필수적입니다.

● 선형대수 및 확률 (Linear Algebra & Probability): 행렬 곱셈, 고유값, 벡터 공간 등 선형대수의 핵심 개념과 확률 및 통계의 기본 원리는 AI 알고리즘을 이해하는 데 기본이 됩니다.

권장 (Recommended):

● 기계학습/딥러닝 개론 (Introduction to Machine Learning/Deep Learning): 경사 하강법, 역전파, 신경망의 기본 구조에 대한 이해가 권장됩니다. 특히 트랜스포머 (Transformer) 아키텍처에 대한 사전 지식이 있다면 강의 내용을 따라오는 데 매우 유리합니다.

● 컴파일러 (Compilers): 컴파일러의 기본 단계(어휘 분석, 구문 분석, 최적화, 코드 생성)에 대한 개념적 이해가 있으면 Part 3의 내용을 학습하는 데 도움이 됩니다.

● 운영체제 (Operating Systems): 시스템 소프트웨어와 하드웨어 간의 상호작용, 특히 디바이스 드라이버의 역할에 대한 기본적인 이해가 있으면 강의의 전체적인 그림을 이해하는 데 유용합니다.

본 강의는 도전적인 내용을 다루지만, 위의 모든 요건을 완벽하게 충족하지 않더라도 새로운 AI 컴퓨팅 패러다임을 구축하려는 강한 열정과 학습 의지가 있는 학생이라면 성공적으로 수강할 수 있습니다.

2) 수강 시 필요사항 (Course Requirements):

본 강의는 이론과 더불어 실제 시스템을 설계하고 구현하는 실습을 포함하므로, 원활한 수강을 위해 다음과 같은 환경과 도구가 필요합니다.

● 하드웨어 (Hardware):

* 개인 노트북 (Windows/macOS/Linux)

* CPU: 4코어 이상 권장

* RAM: 16GB 이상 권장 (FPGA 설계 툴 및 컴파일러 빌드 시 상당한 메모리가 필요합니다.)

* 저장 공간: 100GB 이상의 여유 공간 (각종 개발 툴 설치를 위해 필요합니다.)

* ※ 일부 무거운 작업은 학과 실습실 서버를 이용할 수 있도록 안내할 예정입니다.*

● 소프트웨어 (Software):

* 운영체제: Linux 환경 (Ubuntu 22.04 LTS 권장). Windows 사용자의 경우 WSL2 (Windows Subsystem for Linux) 설치가 필수적입니다.

* 프로그래밍 언어: Python 3.10 이상, C++ (gcc/clang 컴파일러)

* AI 프레임워크: PyTorch

* 컴파일러 인프라: LLVM & MLIR (소스 코드 빌드 방법은 강의에서 안내)

* 하드웨어 설계 툴: Xilinx Vivado 또는 Intel Quartus (설치 및 라이선스 관련 안내 예정)

* 버전 관리 시스템: Git

● 계정 (Accounts):

* GitHub 계정: 모든 실습 및 프로젝트 과제는 GitHub를 통해 제출 및 관리됩니다.

3) 면담 시간 및 장소: 수업 직후 34동 405호

7 장애 학생 지원 사항

강의 수강 관련	○ 시각장애: 교재 제작(디지털교재, 점자교재, 확대교재 등), 대필도우미 허용 ○ 지체장애: 교재 제작(디지털교재), 대필도우미 및 수업보조 도우미 허용 ○ 청각장애: 대필 및 문자통역 도우미 활동 허용, 강의 녹취 허용 ○ 건강장애: 질병 등으로 인한 결석에 대한 출석 인정, 대필도우미 허용 ○ 학습장애: 대필도우미 허용 ○ 지적장애/자폐성장애: 대필도우미 및 수업 멘토 허용
과제 및 평가 관련	○ 시각장애/지체장애/청각장애/건강장애/학습장애: 과제 제출기한 연장, 과제 제출 및 응답 방식의 조정, 평가 시간 연장, 평가 문항 제시 및 응답 방식의 조정, 별도 고사실 제공 ○ 지적장애/자폐성장애: 개별화 과제 제출 및 대체 평가 실시
비고	본 강의를 수강하는 장애학생들에게는 이상의 지원 서비스 이외에도 장애학생 개개인의 특성과 요구에 따라, 지도교수 및 장애학생지원센터와의 상담을 통하여 적절한 수준의 지원 서비스를 제공합니다. 장애학생에 대한 지원서비스와 관련하여 문의사항이 있는 학생들은 담당교수 김태완(02-880-1434) 혹은 장애학생지원센터(02-880-8787)로 문의바랍니다.