ECE 6775
High-Level Digital Design Automation
Fall 2025

Final Project

Cornell University

CSL

Announcements

- Instructor OH cancelled today

- **Lab 4 deadline: due Wed Nov 5**
  - Additional notes on data reuse with linebuffer posted on Ed
  - Focus on reducing HLS-estimated latency first before generating bitstream
  - When generating the bitstream, note that designs with high resource utilization can sometimes lead to routing failures

Agenda

- Project logistics and recommendations
- Closing remarks on the lecture content

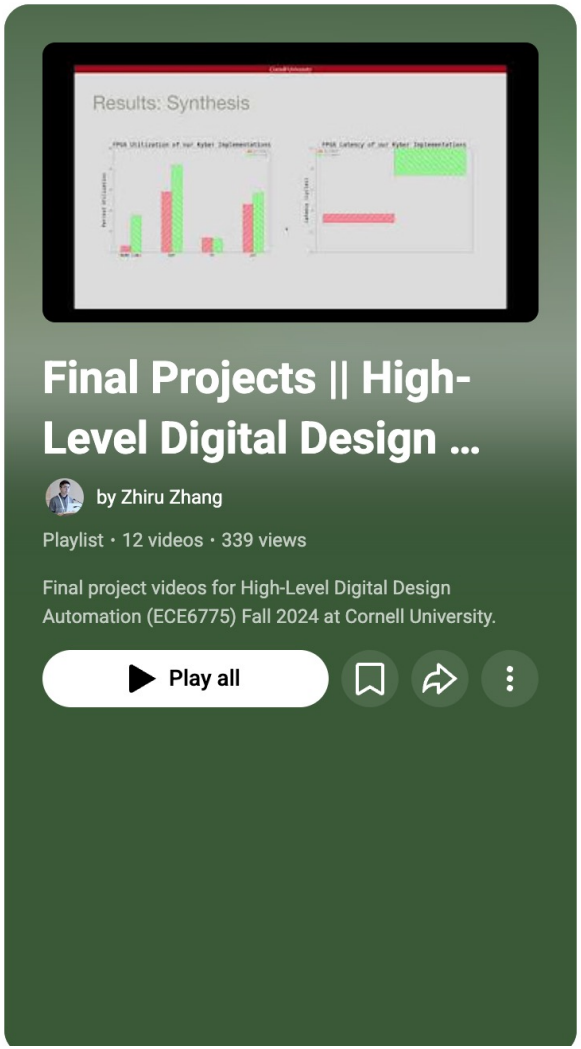
Project Logistics

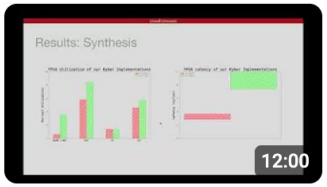
- Fill out project teaming sheet by Saturday noon
  - 3-4 students per teams (13 teams total)

- Project meetings start on Tuesday at **Rhodes 471 (CSL)**
- First meeting next week: Define the project scope
  - **Each team should bring a project idea to discuss with the instructor and, ideally, have a fallback plan ready**

- Due dates
  - Project abstract due Friday 11/7 (**no extension**)
  - Presentation due Tuesday 12/9 in a recorded video
  - Final report due Friday 12/12

Project Presentations from 5775 FA24



1 

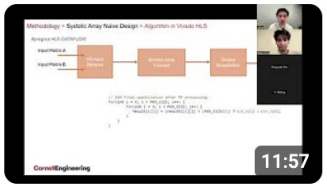
FPGA Acceleration of Post-Quantum Cryptography || Final Project || ECE6775...

Zhiru Zhang • 277 views • 10 months ago

2 

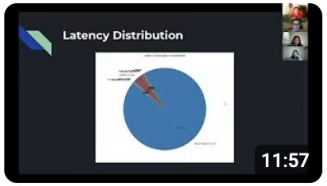
Accelerating Mandlebrot Computation on FPGAs || Final Project || ECE6775 FA24

Zhiru Zhang • 109 views • 10 months ago

3 

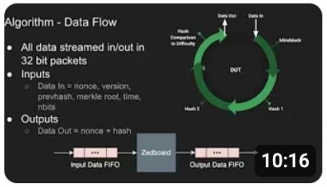
Systolic Array with Binarized Matrix Multiplication || Final Project || ECE6775...

Zhiru Zhang • 357 views • 10 months ago

4 

BitNet LLM Attention Kernel Exploration: Vivado HLS, Allo, Catapult || Final...

Zhiru Zhang • 168 views • 10 months ago

5 

Accelerating Hashing Algorithm in Bitcoin Mining Simulation || Final Project ||...

Zhiru Zhang • 197 views • 10 months ago

https://www.youtube.com/playlist?list=PLRvJfry30-22rMD4iBcWdypyyIiDOvpGq

Project Abstract (due Friday 11/7)

- Two project themes
  - **App**: Accelerator design for compute/data-intensive applications
  - **Tool**: Accelerator design automation tools/algorithms

- Abstract format
  - Write a concise one-page project overview consisting of 2-3 paragraphs
  - Include the project title, theme, and list of team members
  - Summarize the project, outlining key approaches
  - Justify the project's feasibility within the given time constraints

Project Grading (35%)

- 35% = 1% abstract + 6% report + 4% presentation + 24% project development

- More detailed guidelines on project report and presentation will be posted next month

- **Project development (24%)**
  - **Difficulty score (8%)**
  - **Execution score (16%)**
    - Baseline setup (5%)
    - Optimization (6%)
    - Evaluation (5%)

Theme 1 (App) Accelerator Design

- Utilize HLS to create FPGA-based hardware accelerators for compute- or data-intensive applications
- Design languages: C++ or Python DSL
- HLS tools: AMD Vivado/Vitis HLS, Allo
- FPGA platforms
  - ZedBoard: a small device; relatively short compile time
  - Alveo U280: a much larger device equipped with high-bandwidth memory (HBM); long compile time (on the order of hours)

Theme 1: Topics to Consider

- New application-specific accelerators
  - Build accelerators for compute or data-intensive apps, e.g., AI/ML, cryptography, image/video processing, genomics

- New domain-specific accelerator architectures
  - Example 1: Extend a systolic array to handle sparsity or add configurability to support multiple kernels (e.g., MM, MV, Conv)
  - Example 2: Create specialized memory architecture to handle irregular data access patterns (e.g., custom cache support for sparse workloads)

Choosing the Right Application

Ideal characteristics for hardware acceleration

(a) Custom (low-bitwidth) numeric types

(b) Abundant parallelism

(c) Distributed memory accesses

Examples

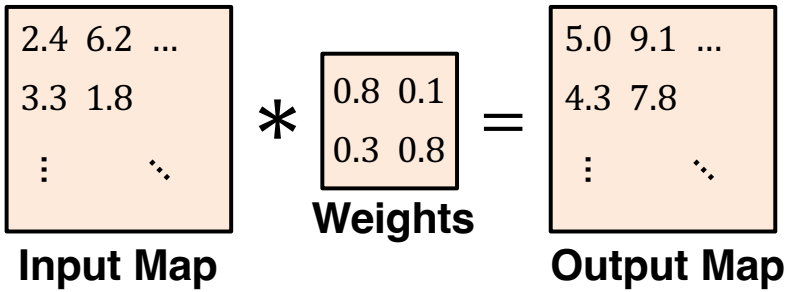
Lab 1 – CORDIC **(a)**

Labs 2, 3 – K-Nearest Neighbors **(b) (c)**

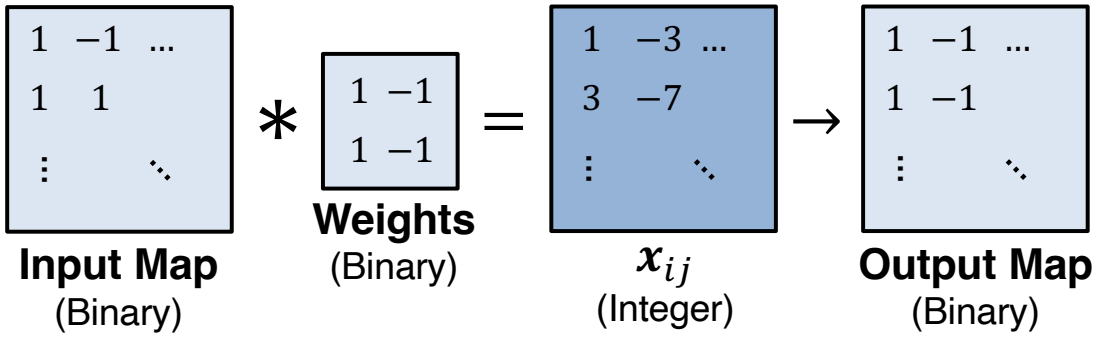
Lab 4 – Binary Neural Network **(a) (b) (c)**

Lab 4: Extreme Quantization with Binarization

CNN



BNN



Key Differences

1. Inputs are binarized (–1 or +1)
2. Weights are binarized (–1 or +1)
3. MAC becomes XNOR+Popcount

BNNs are well suited for FPGAs (rich in LUTs)

Theme 1: Do's and Don'ts

- Choose an application you are familiar with, or one that's relatively easy to grasp
- Minimize "setup" time for the baseline implementation (under 1 week)
- Analyze parallelism and OI before HLS coding
- Add test harness before applying optimizations
- Focus on HLS-level performance optimization and minimize runs of bitstream generation

Theme 1: Anticipated Project Schedule

- Week 1 (Nov 3): Discuss project ideas and write the project abstract
- Week 2 (Nov 10): Complete baseline design with proper testing
- Week 3 (Nov 17): Perform design optimizations at the HLS level
- Week 4 (Nov 24): Implement the design on board
- Week 5 (Dec 1): Improve results and work on the project report

Theme 2 (Tool) Accelerator Desing Automation

- Develop new compilation and synthesis techniques
  - Compiler analysis & transformations for accelerators
  - Improving core HLS algorithms: scheduling, pipelining, binding
  - Optimizing new design metrics (e.g., security)

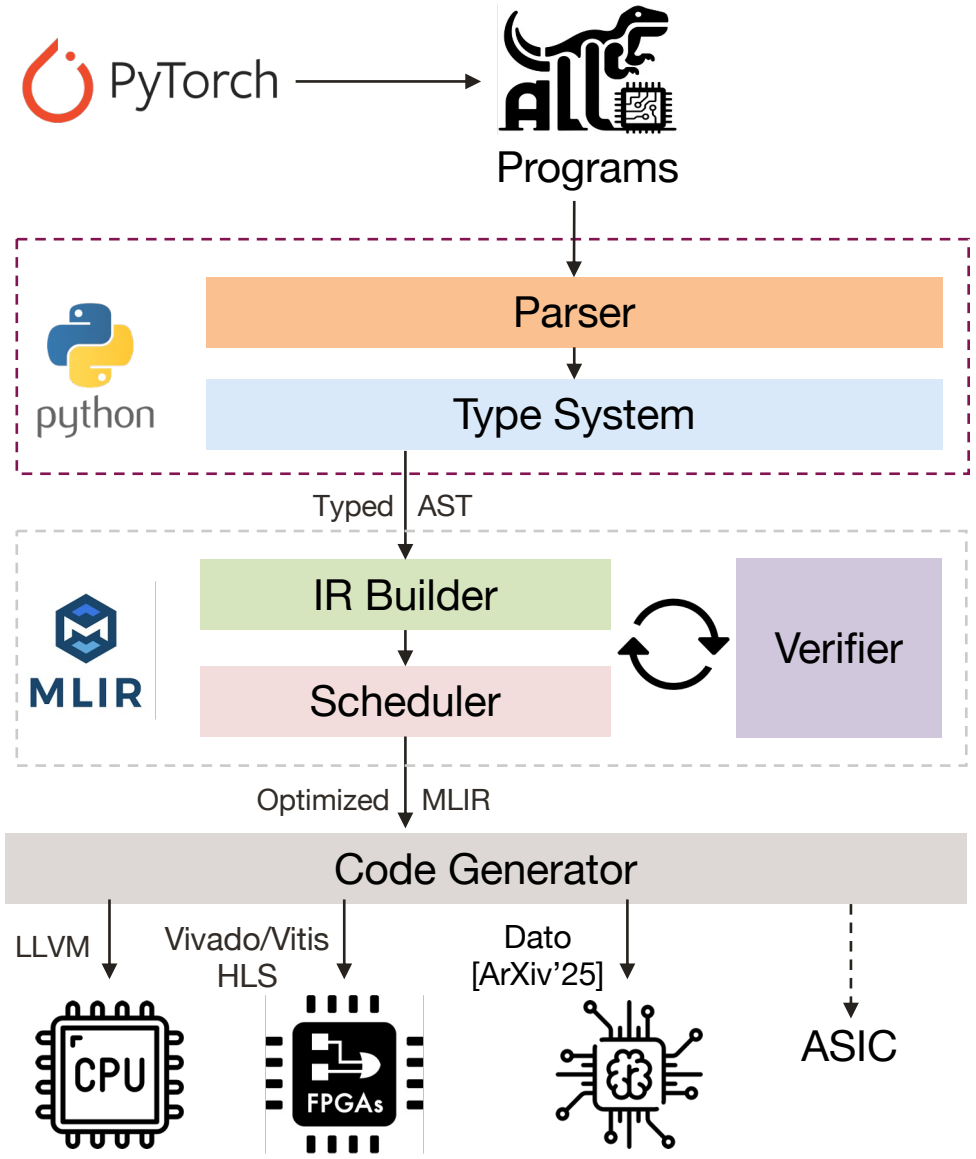
- Software frameworks
  - Open-source compiler infrastructure: Allo, LLVM, MLIR
  - Commercial HLS as a back end: Vivado/Vitis HLS, Siemens Catapult HLS, Google XLS

Theme 2: Topics to Consider

- Machine learning for HLS
  - New scheduling algorithms leveraging ML techniques
  - LLM agents for design space exploration

- Extending Allo accelerator design language
  - Automated generation of hardware customization primitives
  - PyTorch to HLS lowering through Allo
  - Target ASIC through Catapult HLS or Google XLS

Theme 2: A Closer Look at Allo




```
@df.kernel(mapping=[P0, P1])
 def matmul(A: int8[M, K],
            B: int8[K, N],
            C: int16[M, N]):
  i, j = df.get_pid()

  with allo.meta_elif(i == M+1 and j > 0):
   for k in range(K):  # drain
    b: int8 = pipe_B[i, j].get()
  with allo.meta_elif(j == N + 1 and i > 0):
   for k in range(K):
    a: int8 = pipe_A[i, j].get()
  with allo.meta_else():  # main processing
   c: int16 = 0
   for k in range(K):
    a: int8 = pipe_A[i, j].get()
    b: int8 = pipe_B[i, j].get()
    c += a * b
    C[i - 1, j - 1] = c
    pipe_A[i, j + 1].put(a)
    pipe_B[i + 1, j].put(b)
```

https://github.com/cornell-zhang/allo

Theme 2: Do's and Don'ts

- Leverage open-source compiler infrastructures and programming frameworks
  - Avoid building a new IR from scratch

- Formulate the problem in an exact way before implementing any heuristic algorithms

Theme 2: Anticipated Project Schedule

- Week 1 (Nov 3): Discuss project ideas and write the project abstract
- Week 2 (Nov 10): Set up the baseline toolflow
- Week 3 (Nov 17): Implement new optimizations and/or programming constructs
- Week 4 (Nov 24): Continue tool development
- Week 5 (Dec 1): Evaluate the tool on real designs and work on the project report

Recap: What this Course is About

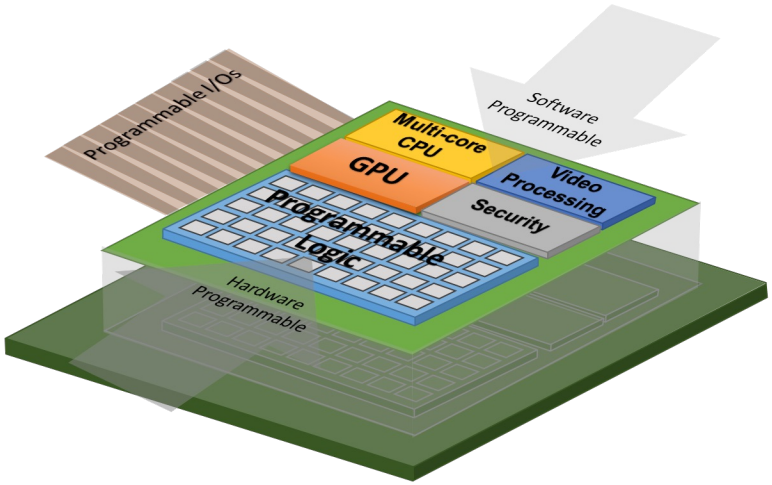
Hardware/Software ***Co-Design***

- Specify applications/algorithms in software programs
- Synthesize software descriptions into special-purpose hardware components, namely, *accelerators*
  - Perform manual source-level code optimizations
  - Utilize automatic compilation & synthesis optimizations
  - Explore performance-cost trade-offs
- Realize the synthesized accelerators on FPGAs

Co-Design Revisited

"HARD" **Mapping Software**

"SOFT" **to Hardware**



"SOFT": FPGA is a reconfigurable fabric

"HARD": Performance-oriented programming is challenging, esp. for FPGAs

Recap: Learning Outcomes (The Intangibles)

1. Develop a principled approach to analyzing accelerator design process and essential design factors (e.g., parallelism, resources, precision)

2. Gain comprehensive insights into accelerator design from the perspective of an HLS compiler

We achieve these objectives through **a balanced mix of theoretical foundations** (lectures & homework) **and practical applications** (labs & project)

“Recipe” for Accelerator Design with HLS

- To achieve high performance:

  Run as many processing elements (PEs) in parallel as possible

  → *unrolling* (more PEs), *pipelining* (higher PE utilization)

  Challenge: Keep the PEs continuously fed with data

- Common bottlenecks:

  Limited memory bandwidth, on-chip or off-chip, e.g., each BRAM has only two ports

  → *array partitioning* (more ports), *array reshaping* (wider access per port), *data reuse* (fewer off-chip accesses), *custom data types**

  Limited on-chip capacity (memory or compute):

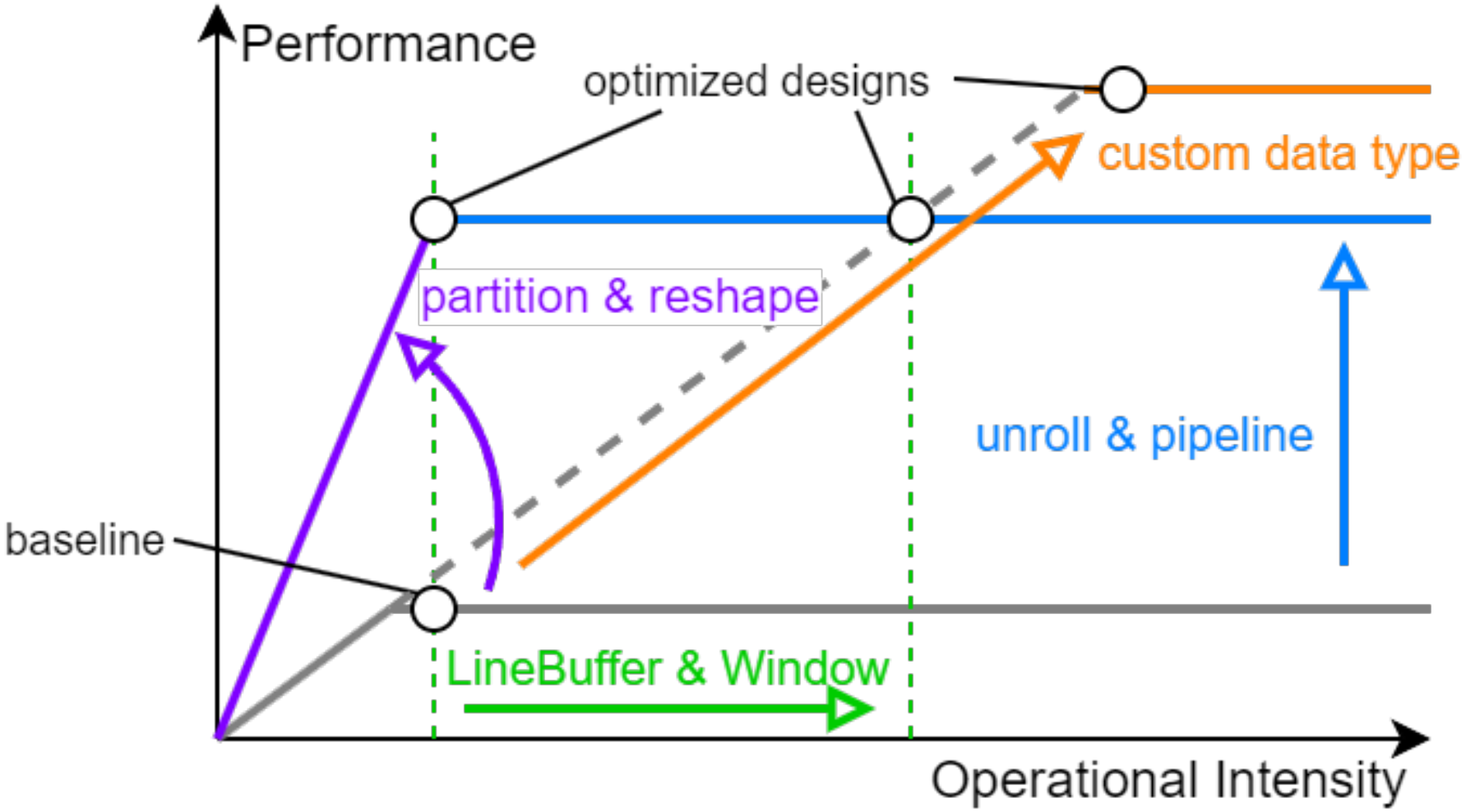
  → *tiling*, applies to loop and data; *custom data types*

  Carried dependence, i.e., recurrence:

  → *reordering* of compute (e.g., loop reorder) can help in some cases

* Custom numerics may alter design’s behavior and require careful algorithm-hardware co-design

Roofline Model, Another Look



Next Time

- Project meetings (Rhodes 471)