ECE 6775
High-Level Digital Design Automation
Fall 2023

Final Project

Cornell University

CSL

Announcements

- Lab 4 due tomorrow
  - Useful tips on array partition & reshape posted on Ed
  - Focus on reducing HLS-estimated latency before generating bitstream
  - Designs with high resource utilization may cause routing failures

Project Logistics

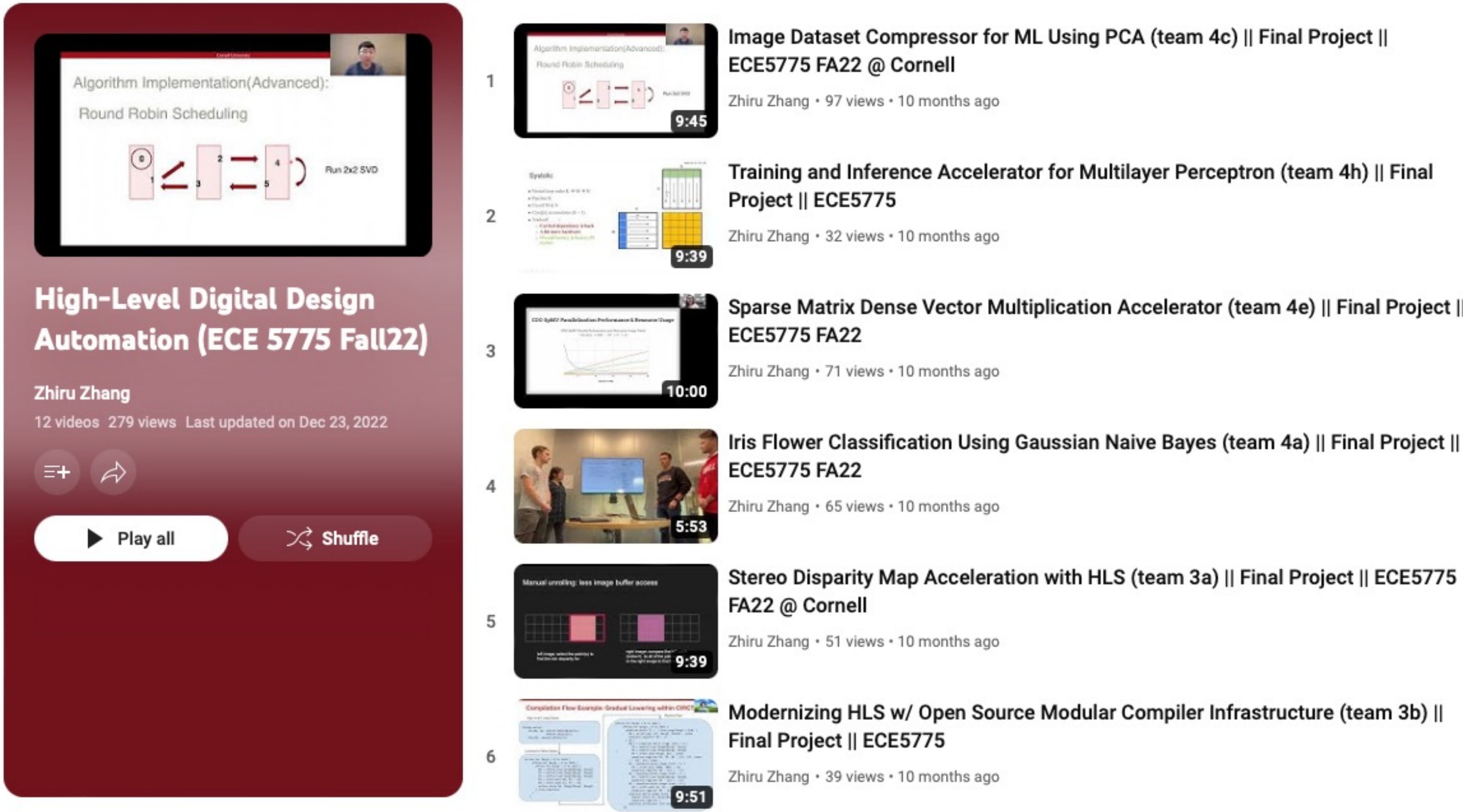
- Fill out project teaming sheet today!
  - 3-4 students per teams (12 teams expected)

- Project abstract due Wed 11/8 (no extension)

- Project meetings start on Thursday at **Rhodes 471**

- Due dates
  - Project abstract due Wed 11/8 (**no extension**)
  - Demo due Mon 12/11 in a **recorded video**
  - Final report due Thu 12/14

Project Presentations from 5775 FA22



https://www.youtube.com/playlist?list=PLRvJfry30-22OqHmVYfruWs8Hv-Fnwr57

Project Abstract (due Wed 11/8)

- Two project themes
  - **App**: Accelerator design for compute/data-intensive applications
  - **Tool**: Compilation/synthesis for accelerator design/programming

- 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

Theme 1 (App)
Application-Specific Accelerator Design

- Utilize HLS to create FPGA-based hardware accelerators for compute-intensive applications
  - Explore hardware customization techniques in emerging application domains, e.g., computer vision, genomics, machine learning, confidential computing.

- Design languages: C++ or DSL

- HLS tools: Xilinx Vivado/Vtis HLS

- FPGA platforms
  - ZedBoard: a small device; relatively short compile time
  - Alveo (datacenter): a much larger device equipped with high-bandwidth memory (HBM); long compile time (~hours)

Theme 1: How to Choose an Application?

Ideal application characteristics for hardware acceleration
(a) Abundant parallelism
(b) Custom (low-bitwidth) numeric types
(c) Distributed memory accesses

Lab 1 CORDIC: **(b)**

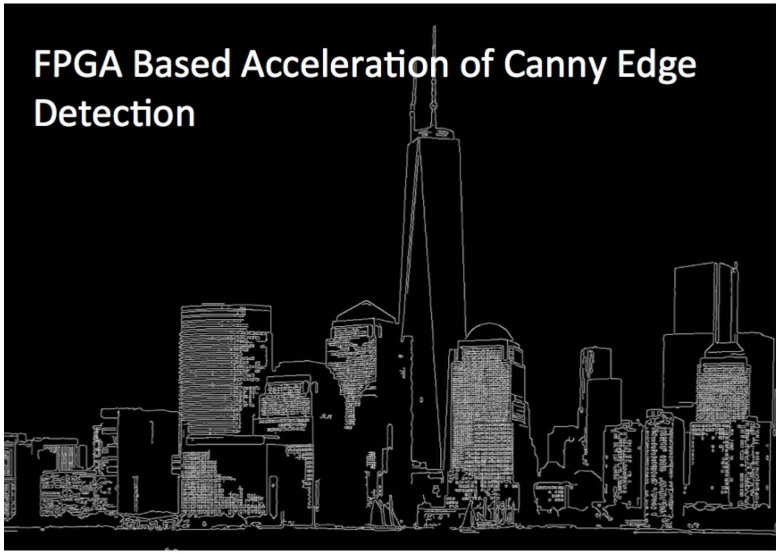
Lab 2/3 K-Nearest Neighbors: **(a) (c)**

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

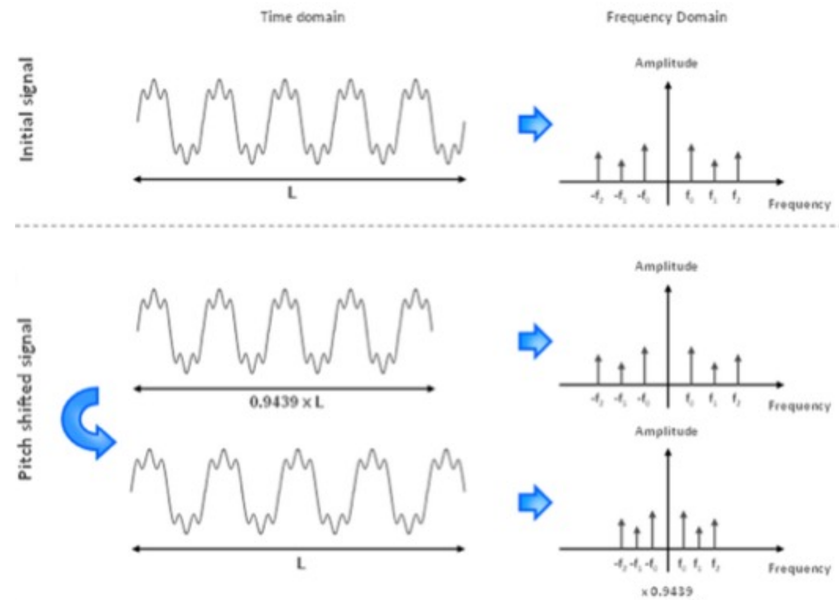
Theme 1: Topics to Consider

- Compute customization
  - Systolic array design for dense linear algebra, e.g., MV, MM, Conv
    - Evaluate the design using a simple ML model such as an attention layer in Transformer
- Data type customization
  - A parameterized HLS library for low-precision floating point arithmetic (e.g., E2M4, E3M1)
    - Evaluate the library using lab designs such as CORDIC or BNN
  - Large-bitwidth integer arithmetic on FPGAs (e.g., 512b multiply)
    - Evaluate the design using a crypto application
- Memory customization (explored in past projects)
  - Reuse buffer for stencil-based image/video processing
  - Custom memory layout for sparse linear algebra

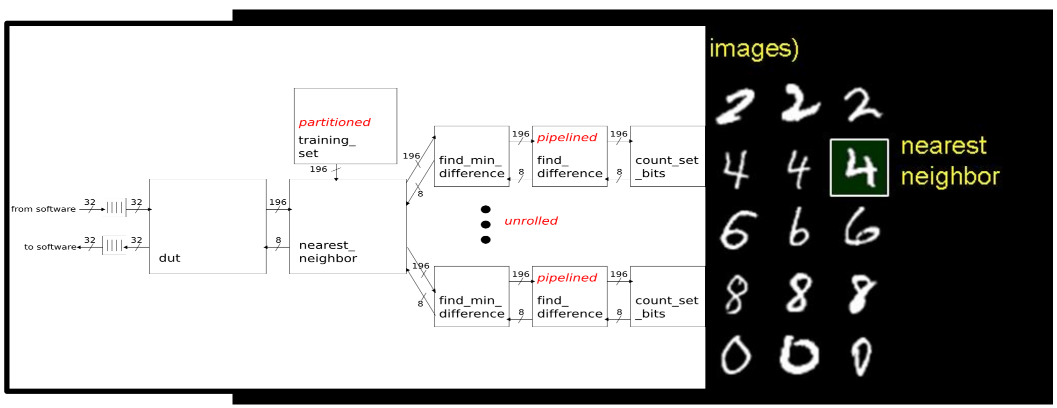
Theme 1: Sample Projects from Previous Years



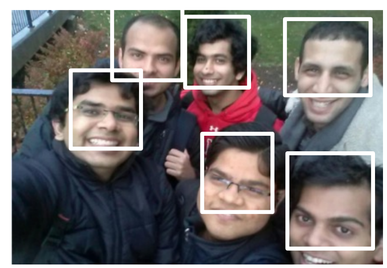
Canny Edge Detection



Real-Time Vocal Processor of Pop Music



Digit Recognition



Face Detection

Theme 1: Do’s and Don’ts

- Choose an application you are familiar with (or one that’s easy to grasp)

- Minimize “setup” time for the baseline implementation (under 1 week)

- Analyze parallelism and operational intensity (OI) before HLS coding

- Focus on HLS-level performance optimization and minimize runs of bitstream generation

Anticipated Project Schedule

- Week 1: Brainstorm and project abstract
- Week 2: Complete baseline design with proper testing
- Week 3: Perform design optimizations at HLS level
- Week 4: Implement design on board
- Week 5: Continue optimization and work on the project report

Theme 2 (Tool)
Accelerator-Centric Compilation/Synthesis

- Develop new compilation and/or HLS 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: LLVM, MLIR, HeteroCL
  - Commercial HLS as a back end: Vivado/Vitis HLS

Theme 2: Topics to Consider

- New customization primitives in HeteroCL
  - App-specific data reuse schemes
- Automated OI analysis
  - First-order data reuse analysis for DNNs
- Automated memory customization
  - Customized cache generation for irregular data access patterns

Theme 2 Sample Project: Trace-Based Array Partitioning

Cycles	RD0		RD1		RD2		RD3	
	i	j	i	j	i	j	i	j
1	0000	0000	0000	0010	0010	0000	0010	0010
2	0000	0001	0000	0011	0010	0001	0010	0011
…	…	…	…	…	…	…	…	…
10	0000	1001	0000	1011	0010	1001	0010	1011
…	…	…	…	…	…	…	…	…

```
int A[Rows][Cols];
int sum;

for ( int i = 1; i < Rows – 1; i ++ )
   for ( int j = 1; j < Cols – 1; j ++ )
        sum =  A[i-1][j-1] + A[i-1][j+1]
                  + A[i+1][j-1] + A[i+1][j+1];
```

Important bits (mask)

i	…	i_3	i_2	i_1	i_0
j	…	j_3	j_2	j_1	j_0

Hash function

Mask	Bank
00	0
01	1
10	2
11	3

i/j	0	1	2	3	4	5	6	7
0	0	0	1	1	0	0	1	1
1	0	0	1	1	0	0	1	1
2	2	2	3	3	2	2	3	3
3	2	2	3	3	2	2	3	3
4	0	0	1	1	0	0	1	1
5	0	0	1	1	0	0	1	1
6	2	2	3	3	2	2	3	3
7	2	2	3	3	2	2	3	3

Partitioning array A

Theme 2: Do's and Don'ts

- Leverage open-source compiler infrastructures (e.g., LLVM, MLIR)
  - Avoid building a new IR from scratch

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

Recap: About This Course
Hardware/Software Co-Design

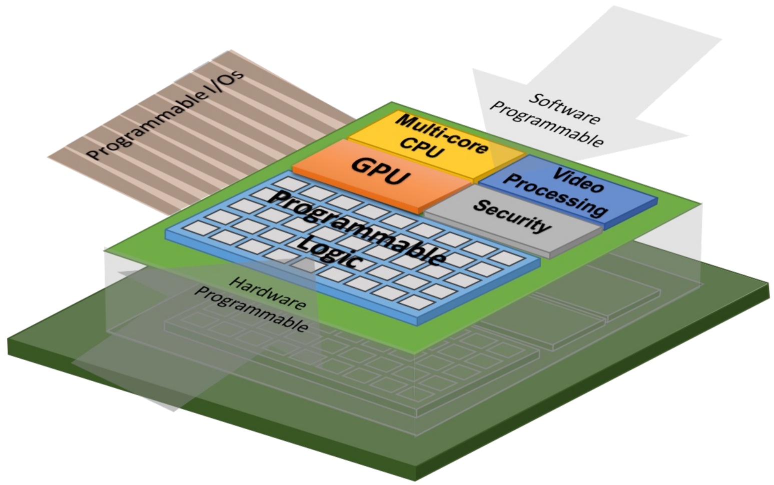
- Specify applications/algorithms in software programs

- Synthesize software descriptions into special-purpose hardware architectures, namely, *accelerators*
  - Explore performance-cost trade-offs
  - Exploit automatic compilation & synthesis optimizations

- Realize the synthesized accelerators on FPGAs

Co-Design Revisited

	Mapping
"HARD"	**Software**
	to
"SOFT"	**Hardware**



"SOFT": FPGA is a reprogrammable fabric

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

Blurred Line Between Hardware and Software Design

- Hardware

- Software

```c
int max;

int findmax(int a[10]) {
  unsigned i;
  max = a[0];
  for (i=1; i<10; i++)
    if (a[i] > max) max = a[i];
}
```

```asm
            .text
findmax:    ldr       r2, .L10
.L10        ldr       r3, [r0, #0]
            str       r3, [r2, #0]
            mov       ip, #1
.L7:        ldr       r1, [r0, ip, asl #2]
            ldr       r3, [r2, #0]
            add       ip, ip, #1
            cmp       r1, r3
            strgt     r1, [r2, #0]
            cmp       ip, #9
            movhi     pc, lr
            b         .L7
.L11:       .align    2.
L10:        .word     max
```

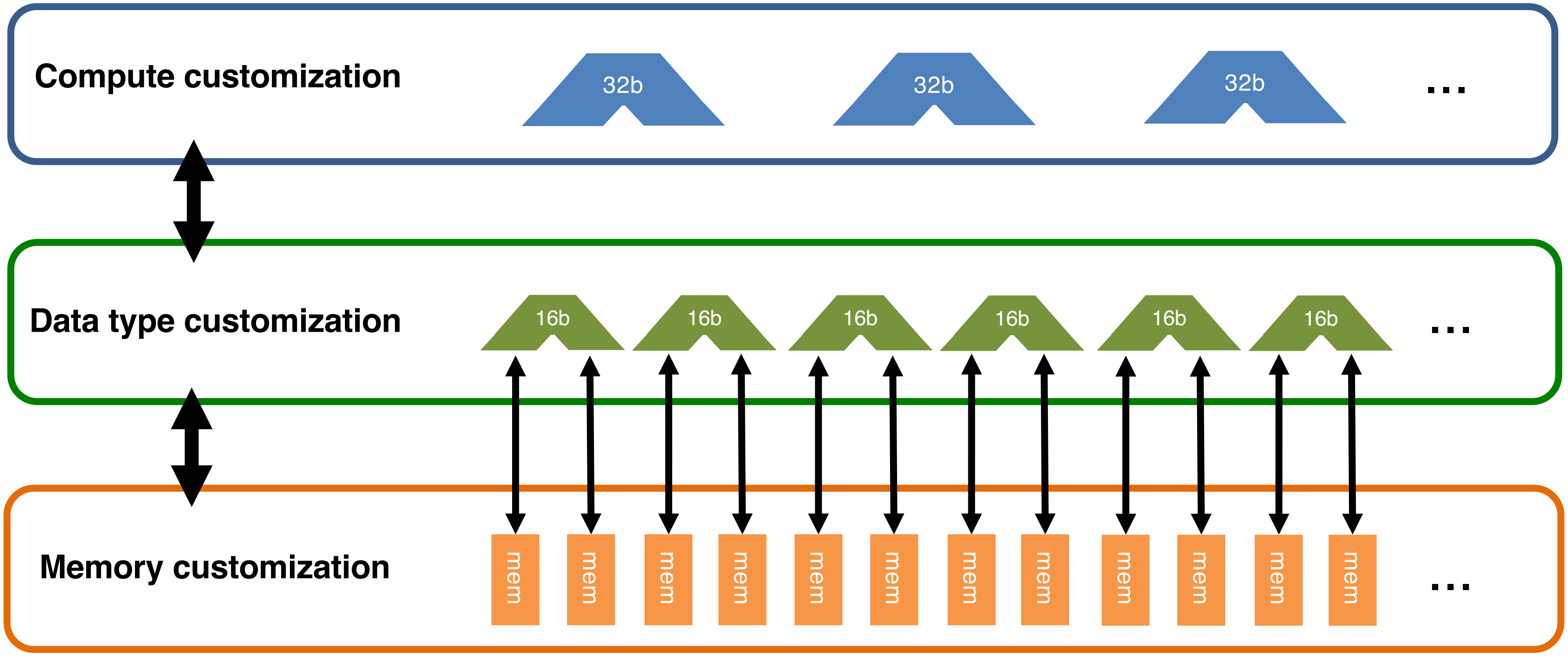
Recap: Learning Outcomes (The Intangibles)

- Develop a principled approach to analyzing accelerator design process and essential design factors

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

Achieve these objectives through a blend of theoretical foundation and practical implementation

Essentials of Hardware Specialization



Parallelism
(dependence)

Resources
(logic, memory, I/O)

Data
(access pattern)

Pipelining   Parallel processing   Bandwidth   Reuse   Precision   Layout