18-643 Recitation 4: More on the Project

Shashank Obla
PhD Candidate, Department of ECE
Carnegie Mellon University

What's today about?

- Your goal today: discuss more about the project specifications and get a feel for what good proposal could look like
- Housekeeping:
  - Lab 1 grades are up on Canvas
  - Review comments and feel free to get back to us
  - Recitation 3 covered the concepts behind Lab 1
- Only 3 – 4 weeks left until proposal!
- This is not a lecture! You need to be interactive, lots of discussions, questions and answers

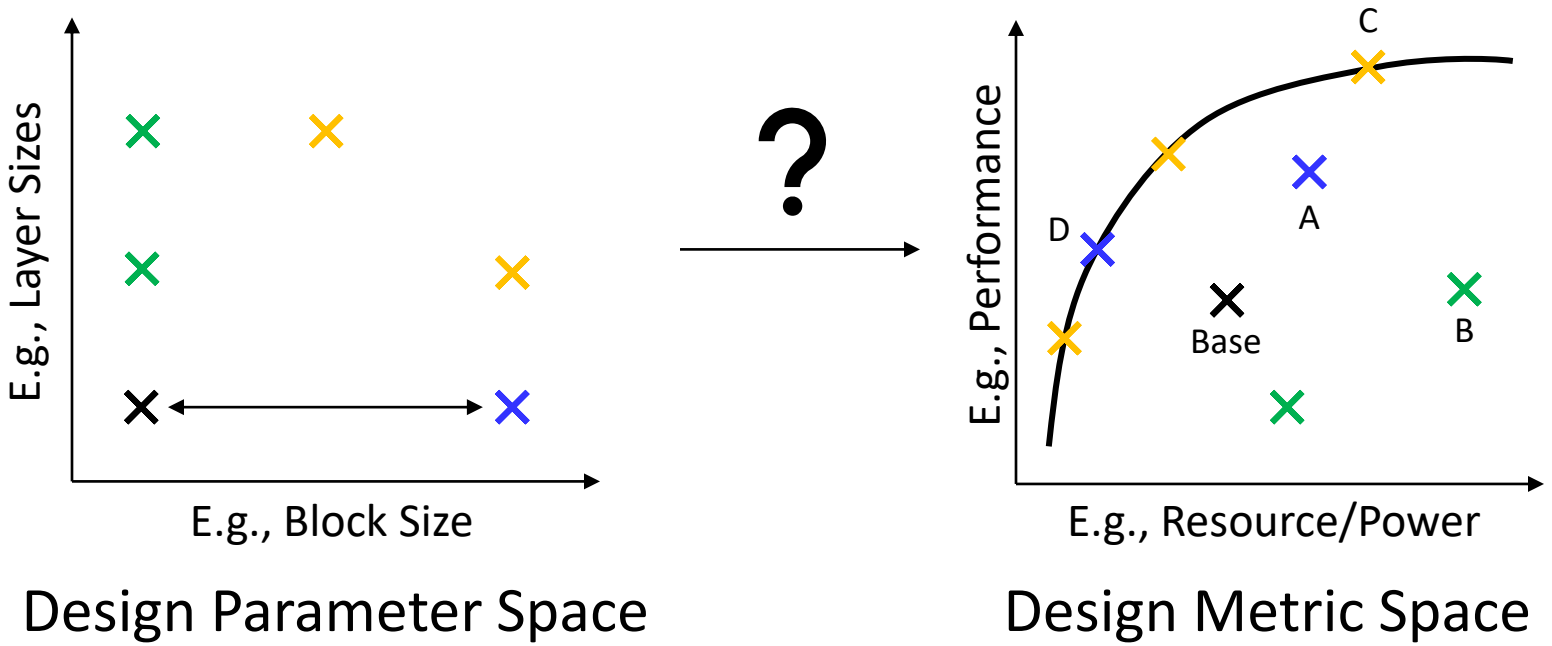
643 Project: What's it about?

- A beginner's guide to research!
  - Not "a" design but study of a design space
- Proposal: Coming up with a question
  - Should be interesting for you
  - Should not be entirely new – you're trying to push the boundary and not learn what's out there
  - Where you are, where you want to be, how to get there, and think about contingencies.
- Doing the project – Execution/Engineering
  - What you've already been training for
  - Success means you're able to answer the question

What makes a good project?

1. Application: What's an FPGA good for?
   - Compute Applications w/ off-chip data IO
   - Irregular computation, Regular computation but not throughput, Streaming applications
2. Metric(s): What to optimize for?
   - Constraints (validity) vs Optimization metrics
3. Design Space: Not one instance but a space
   - Metrics interact with implementation choices
   - Can you extrapolate to other parts in the space?
4. Tools and Platforms: What you need to get going

The Question: Getting from the Design Space to the Metric Space



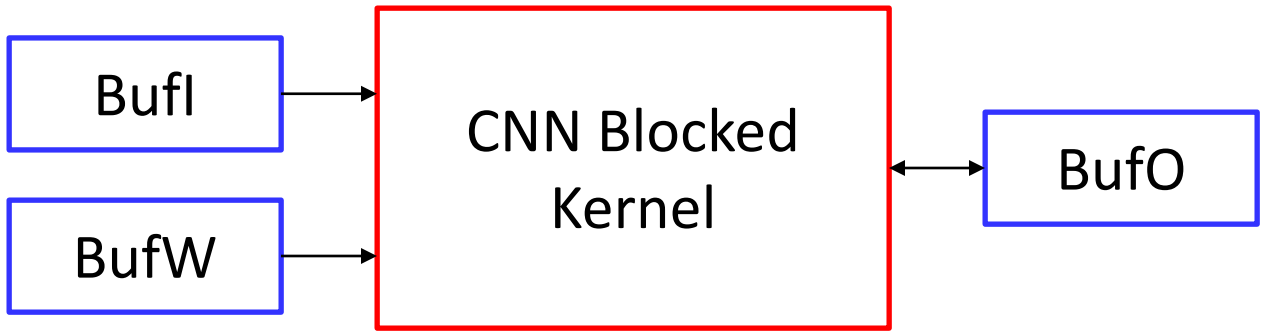
Design Parameter Space

Design Metric Space

- Not throwing darts: Informed decisions
- Hypothesize the relationship and prove/disprove

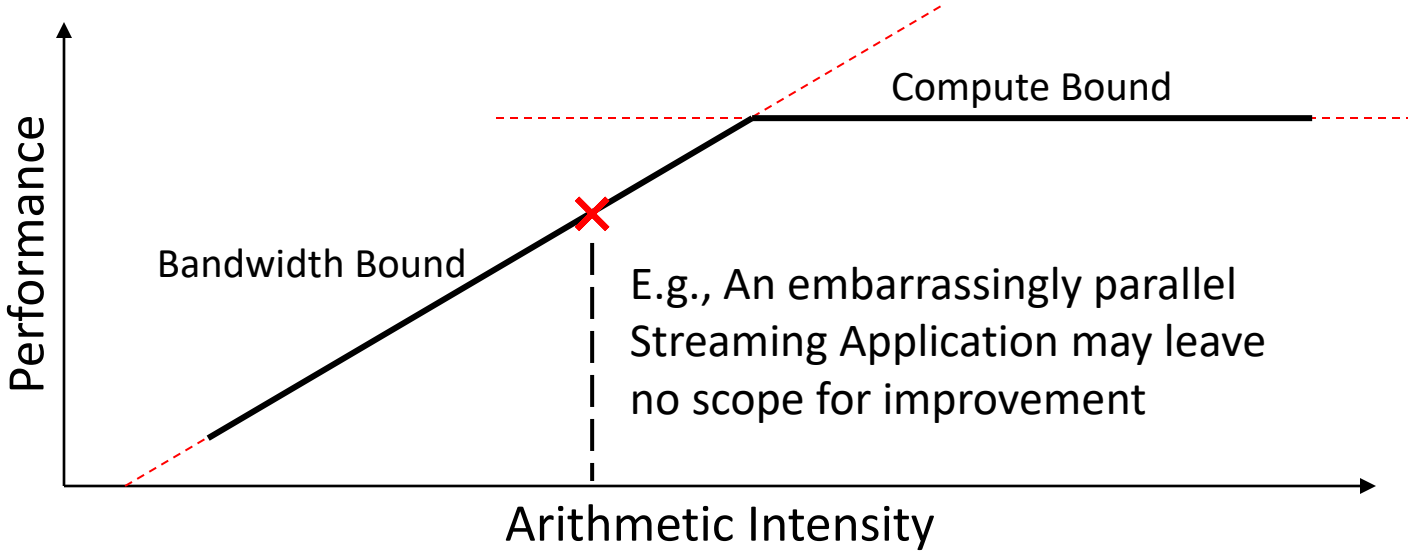
What's the IO requirement?

- Consider Lab 2 CNN Kernel



- This is a "kernel"
  - No off-chip IO ⇒ the problem is not interesting
- Off-chip IO involved talking to the memory or any external source/sink (could be emulated)
- Is that enough?

It has IO, so why is it not OK?



- Ultra96 DRAM/IO system is not great (~ 4 GBps)
  - Can be memory bound always – not interesting
- Simple designs might be memory bound
  - Need to work hard to improve performance!
  - Co-design memory and compute to improve AI

Using Resources

- It's OK to start with something already out there
  - Do you understand it well enough?
  - Is there something you want to study about it?
  - Can you modify the design to get what you need?
- You need to pick a paper
  - It could describe an implementation
  - Could be an ASIC or a software version – how would you do it differently for an FPGA?
- Platforms/Tools – HLS or something even higher like Vitis AI or a Domain-specific tool/compiler

Bottom line it should 5-weeks' worth of effort

PRIOR STUDENT PROPOSALS

Disclaimer:

- Keep an open mind, try not to get fixated on these ideas
- This is not an endorsement; the choice of the proposals doesn't mean that they are perfect (in fact the "presentations" could be better)
- These are only the presentations and not the entire, more exhaustive, proposal documents which is also graded

What are we looking for?

- Title – in the form of a question
- Application and Optimization Metric
  - Why this choice? What's interesting and why FPGA
- Background – Brief Research Paper summary
- Preliminary – Existing working implementations
- Overview of Approach: Design Space
- Overview of Expected Results – Hypothesis
- The 5-Week plan
- Risk Assessment and Contingencies

Maybe not all in the proposal presentations

How does the size of tiling of a matrix affect the speed of next-state generation in a Conway Game of Life simulation?

Introduction - Application

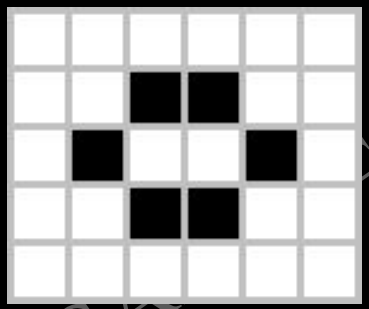
- Conway’s Game of Life, also known as Life, is a zero-player game, where the next-state generation is solely dependent on the current-state
- Each cell interacts solely with its 8 surrounding neighbors
- Using base rules, applied iteratively as many cycles as desired

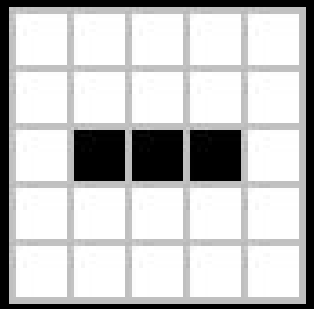
Introduction - Setup and Optimization Metric

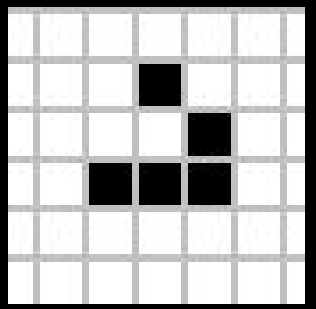
- Given: A random starting state for Conway's Game of Life
- Given: A set number of cycles
- No wraparound
- Desired Output: Figure out final (1024th) state as quickly as possible (latency)

Project Motivation

- Life is probably the most often programmed computer game in existence
- Life is fascinating because of the many patterns that emerge
- Highly parallelizable computation





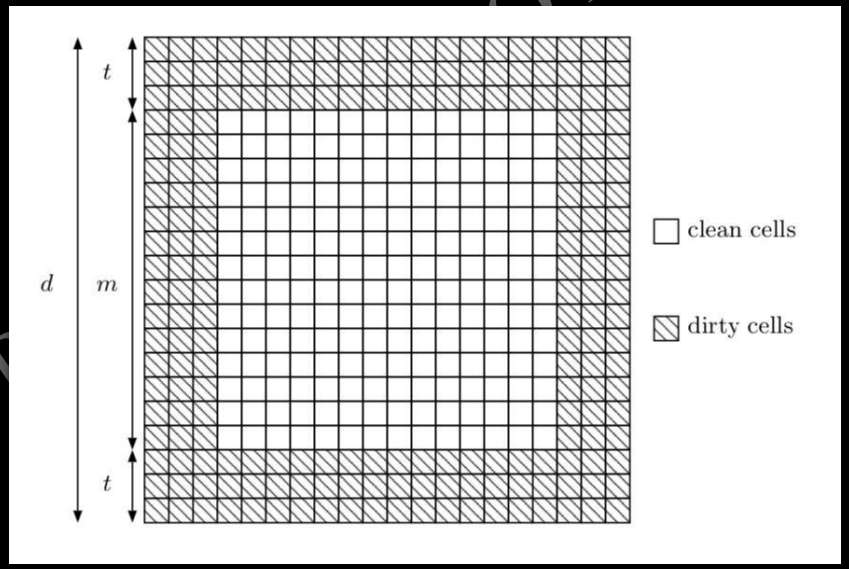


Terminology

- A cycle for the rest of the presentation is defined as the next state or clock tick in a game of life simulation
- An iteration is defined as the next state computed by the program, which may or may not be the next cycle, as it might compute multiple cycles at a time
- A block is the total size of the grid being loaded, and includes clean and dirty cells
- A patch is the total size of the clean cells only
- D-M=2t, where D is the length of the block, M is the length of the patch, and t is the number of cycles computed in 1 iteration

Background

- The paper raised the possibility of computing multiple cycles in each iteration
- After t cycles, the outside t layers of grid will be dirty

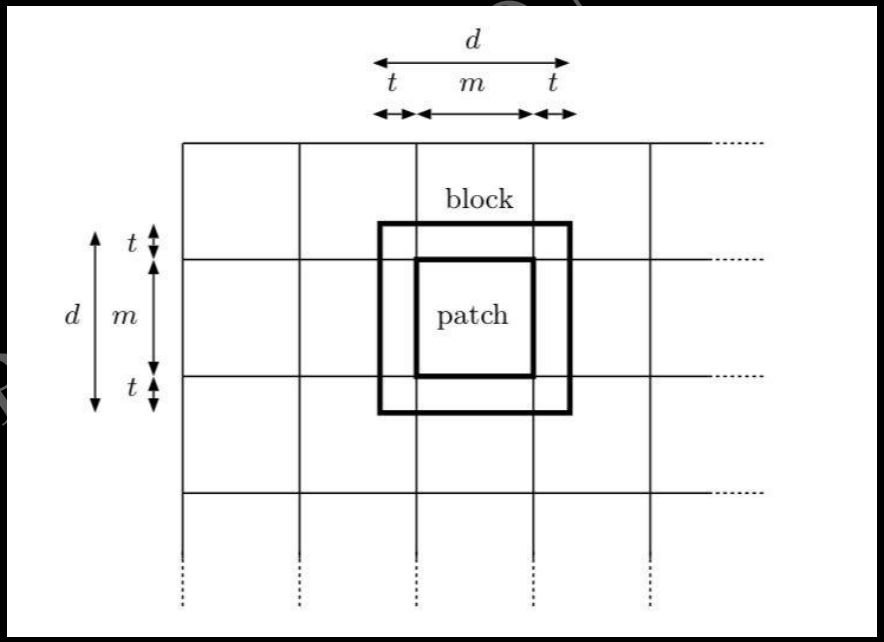


Preliminary

- We have a working C implementation that generates t th cycles
- Given number of cycles and input, it can tell us the output
- Naive implementation of simply updating cell by cell

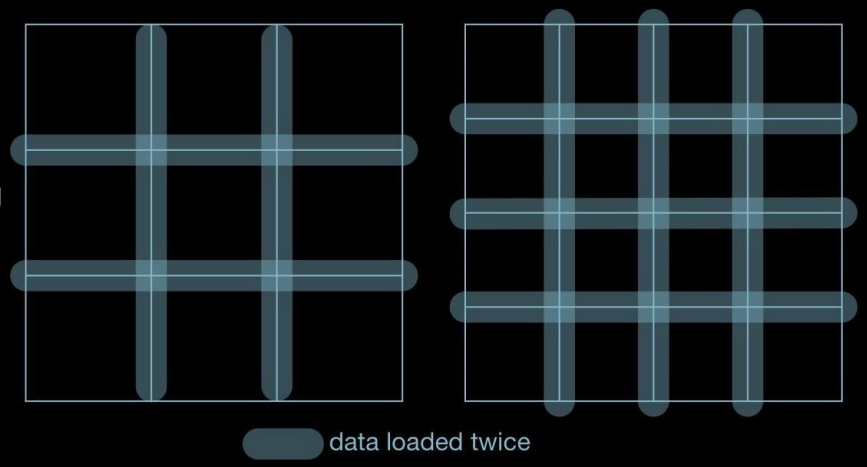
Overview of Approach - Design Space

- Approach 1: Increase size of block, see how performance changes
- Approach 2: Increase size of patch, (size_block - size_patch)/2 is the number of iterations handled between memory loads ((d-m)/2)



Overview of Approach - Trade Offs

- Increasing size of block helps so long as it is within the in-fabric BRAM capacity, and within the the off-chip DRAM bandwidth, as it decreases the number of iterations that is needed to compute the final output, as it is possible to skip cycles



Overview of Approach - Trade Offs

- Decreasing size of tiling becomes problematic if the value of t increases by too much and too many cycles are being skipped in one iteration, and the amount of clean cells that you get in the patch becomes inefficient relative to the size of the block, and is not offset by the decrease in number of iterations required

Overview of Approach - Design Space and Tools

- Size of block in terms of an DxD matrix to generate the next cycle
- Size of patch in terms of MxM matrix to generate the t th cycle (t=(D-M)/2)
- Use Vivado and Xilinx SDK, on the Ultra96 board

Overview of Approach - Investigating Design Space

- Keep 1 constant while changing the other.
- First start by increasing D, while holding M=D-2 constant
- Then decrease M by 2 each time, while holding D constant
- Converge to optimal

Overview of Expected Results - Expect to Learn

- Investigate how sweeping D and M affect each other and the overall latency of the system
- Expect to learn about how to investigate a design space effectively
- Expect to gain a deeper insight into Game of Life and its intricacies

Overview of Expected Results - Best Value

- Assume 4GB/s DRAM bandwidth, 1Mbyte BRAM capacity, each cell is stored as a single bit, each byte has 8 bits, computing 1024 cycles of 16384*16384 bit matrix. Assume kernel same 10GFLOPS at 3ns as Lab 3. Differences due to number of FLOPS required to be performed.
- Assume compute bound: Computation in Fabric is bottleneck
  - FLOPS = (256*40)*(8*10^6*8) = 6.55* 10^11 FLOPS
  - FLOPS/FLOPS_kernel = 6.55*10^11 /(10GFLOPS/s) = 65.5 s
- Assume memory bound: DRAM bandwidth is bottleneck
  - Data/Bandwidth: (16384*16384*1024)/(4*10^9*8) = 8.59s
- Assume no other delays
- Choose the higher bound as the best case, hence we expect to be compute bound and expect to take 65.5s

Overview of Expected Results - Target Value

- Assume 4GB/s DRAM bandwidth, 1Mbyte BRAM capacity, each cell is stored as a single bit, each byte has 8 bits, computing 1024 cycles of 16384*16384 bit matrix. Assume kernel same 10GFLOPS at 3ns as Lab 3. Differences due to number of FLOPS required to be performed.
- Assume compute bound: Computation in Fabric is bottleneck
  - FLOPS = (256*60)*(8*10^6*8) = 9.83* 10^11 FLOPS
  - FLOPS/FLOPS_kernel = 9.83*10^11 /(10GFLOPS/s) = 98.3 s
- Assume memory bound: DRAM bandwidth is bottleneck
  - Data/Bandwidth: (16384*16384*1024)/(4*10^9*8) = 8.59s
- Assume no other delays
- Choose the higher bound as the best case, hence we expect to be compute bound and expect to take 98.3s

Weekly Testable Milestones

- Week 11
  - Valid working Vivado Implementation
- Week 12
  - Sweep block matrix DxD in terms of D, holding M fixed
- Week 13
  - Sweep block matrix MxM, while holding D fixed

Weekly Testable Milestones

- Week 14
  - Vary values of D and M and figure out how the 2 interact with each other
- Week 15
  - Prepare final report and presentation

Risk Assessment

- Baseline might not build in a week, if we struggle with customizing our own IP and getting it set up in Vivado
- Unsure where convergence will be reached, specifically if local minimum for D will result in the best value for latency when sweeping M
- Expected time for report writing and final presentation is a bit short given the rush expected near the end of the semester

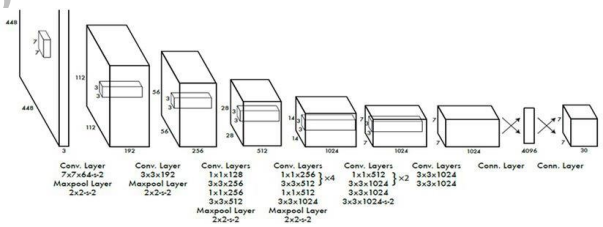
Let's Recap

- Title
  - How the size of tiling affects the compute portion of the Game of Life
- Application and Optimization Metric
  - Stencil Computation in Game of Life for latency
- Preliminary – Basic C implementation as baseline
- Design Space – Compute/Memory Load
- Overview of Expected Results – Best vs Target Value from optimization
- The 5-Week plan, Risks, etc.

Is real time object detection possible?

Introduction

- YOLO (you only look once) CNN algorithm for Real Time Object Detection
  - Predict object bounding boxes and classify each detected object in an image
- The CNN compute problem requires many external memory accesses and numerical computations for each layer
- Real time detection requires high throughput, but still want good accuracy
- Goal: Design a hardware accelerator on FPGA for the pre-trained YOLO CNN to perform fast object detection



Optimization Metrics

- Frame rate:
  - # frames processed / total runtime
- Accuracy: Were all objects in the image correctly detected and identified?
  - Evaluate mean average precision (mAP) for a set of test images
  - AP considers false positive, false negative, and true positive classifications.

Procedure

1. Save a fixed set of 500 test images from 2007 Pascal VOC test set to an SD card
2. Time from when 1st image is read from SD card to when the last labeled image is written back to SD card
3. Take labeled images from SD card and compute the mAP

Motivation

- Recent CNN architectures have evolved to provide high accuracy, but has led to a rapid increase in computation
- CNN algorithms have high computational complexity and require frequent memory accesses so general purpose CPUs have difficulty producing results in real time
- GPUs do not provide as much flexibility for optimizing data access and memory structure as FPGAs
- Programmability, power efficiency, and parallel nature of FPGAs make it a good hardware platform for CNNs

Approach Overview

Design Space

How the resources utilized on the FPGA affects the speedup (in terms of achievable frame rate) and resulting accuracy of the CNN

- Reduce the precision of fixed point representations for the layer weights and activations. This will reduce the size of the computation units, potentially increasing the peak performance we can achieve.
- Memory interleaving, differing memory bandwidths and different hardware structures (loop unrolling and pipelining)
- Adjust utilization of DSPs, LUTs, etc. to optimize speed of the convolution/pooling/batch norm kernels

Approach Overview

- Design Points:
  - Floating point weights
  - Fixed point 8 bit weights
  - Binary weights
- For each number format, design hardware for convolutional, max pooling, and batch norm
  - Explore tile sizing, BRAM shape/size/interleaving, number of kernels

Tool and Platform

- Vivado HLS to generate hardware for the Ultra96v2
- Tiny YOLOv4 (https://github.com/AlexeyAB/darknet)

Overview of Expected Results

- What do we expect to learn?
  - Using FPGAs can improve performance, even over very powerful CPUs
  - Reducing weight precision to increase throughput will lead to a small decrease in mAP
- What is the expected target metric value?
  - YOLOv4-tiny w COCO test-dev: mAP of 33.1
  - Computation for the YOLO CNN is still dominated by the convolutional layers
    - Increased throughput will come primarily from convolution kernel/memory access
  - Floating Point Design:
    - 6.9 billion FLOP per input image
    - From Lab 2, a 10 GFLOPs convolution kernel occupies about 30% of area and can achieve ~6 GFLOPs

Expected Results

- Assuming a majority of computation is for the convolution and intermediate data between convolutions stays on the fabric and using FLOPS from the convolve layer in lab 3:
  - (6 GFLOPs) / (6.9 billion FLOP/image) ≈ 0.87 fps
  - Using buffers for each kernel, double buffer input layer -> 7 fps
  - Larger tiling: Increase AI of kernel -> increase FLOPS for each layer
- Fixed point/binary weights:
  - Holding all/most weights in BRAM will significantly reduce time spent accessing DRAM
  - Fixed point/binary weights will reduce number of DSPs/LUTs needed for each convolution
  - From results in Nguyen '19, these techniques yielded 66 fps on Virtex-7 FPGA VC707
    - Our Ultra96v2 has less than half the resources -> 30 fps

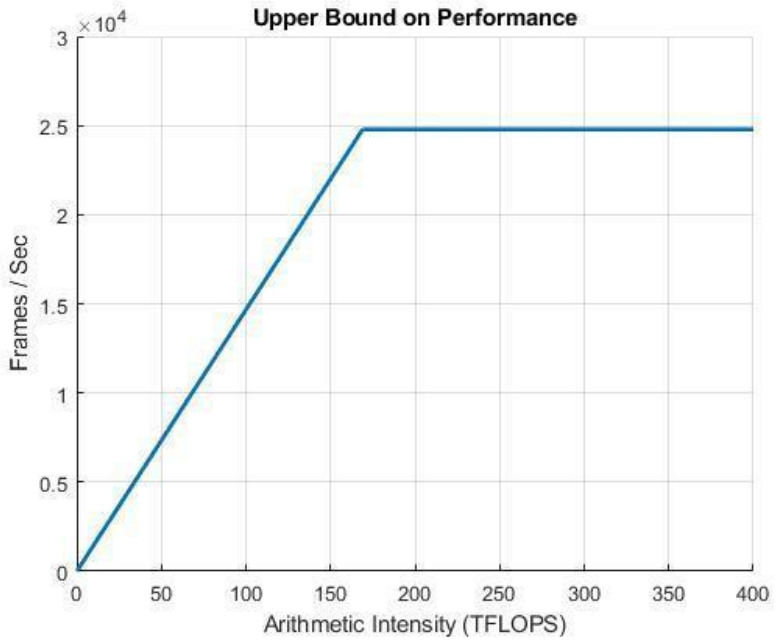
Overview of Expected Results

Upper Bound

- DRAM bandwidth = 4 GB/s
- Image size = 161 kB

DRAM bound: max frame rate = 12,389 fps

- Requires reading or writing 4GB of images every second



Weekly Milestones

Week 10	• Set up Vivado & HLS environment and testbench for a set of test images • Work on loading images from SD card using file system library • Be able to run YOLO CNN inference on the ARM core with loaded images and write results back to SD card • Evaluate performance metrics for baseline results

Week 11	• Design floating point kernels using Vivado HLS for the following layers:
	  • Convolutional
	  • Batch normalization
	  • Max pooling
	• Carefully consider and explore design space.

Weekly Milestones

Week 12	• Integrate 3 kernels with the Arm core and DRAM • Optimize usage of kernels by amortizing data access time • Evaluate performance metrics for floating point designs
Week 13	• Retrain YOLO CNN to use very low bit precision for network weights • Redesign kernels using binary/8 bit weights • Try to store all/most weights in BRAM on fabric • Explore design space with reduced precision kernels

Weekly Milestones

Week 14	• Continue working on low precision/binary weight designs • Integrate these designs w/ the rest of the system • Explore using partial reconfiguration with kernels
Week 15	• Compile results and create graphs demonstrating results • Create final presentation w/ deep dive analyzing results from this study

Let's Recap

- Title
  - Is real time object detection possible?
- Application and Optimization Metric
  - YOLO CNN - throughput (frame rate) and accuracy
- Preliminary – pre-trained CNN implementation
- Design Space – Precision and Block Size/kernels
- Overview of Expected Results – Calculation based on baseline comparison and roofline modeling
- The 5-Week plan, Risks, etc.

Parting Thoughts

- Coming out of the project – Be able to reason about the design space backed by results
- Fill the weekly questionnaire with any thoughts
  - Helps to write things down, clears your ideas
  - Talk to us, come to OH and get feedback early
  - You know what you're going to be assessed on
- Have fun with it!
- Next couple of recitations: Go over some examples of toolchains not covered in the course, possibly Vitis AI and PR (or DFX as Xilinx calls it)