18-643 Recitation 3: Lab 1 Lookback and Project

Shashank Obla
PhD Candidate, Department of ECE
Carnegie Mellon University
What’s today about?

• Your goal today: understand “valid” approaches to designing experiments for Lab 1 and discuss the course project handout to get clarifications

• Pointers on Lab 2:
  – Keep in mind you’re designing hardware
  – Loop reordering, loop optimization, dependencies are bad, memory (design your own cache)
  – CNN and HLS Video on Canvas
  – We’ll deal with external memory in Lab 3

• This is not a lecture! You need to be interactive, lots of discussions, questions and answers
Lab 1: What **was** it about?

- After Lab 0 you know you can use Vitis
- Get used to using Vitis and designing on it
  - Understand different parts of the process
  - Learn to negotiate with it
- Learn basics of benchmarking designs
  - Ops/Second in MMM
- Get your hands dirty on an actual example
  - Do you know what the metrics mean?
  - Designing experiments to measure it
  - Validating experiments: measuring what you want
Some Specs on Ultra96v2

- ARM Core Frequency: 1.2 GHz (don’t worry about the max, that’s not what you’re running at!)
- Default Vitis Fabric Frequency (Target): 150 MHz
  - Achieved is what HLS could do but target is what the fabric is set too (think positive timing slack)
- DRAM: $533 \text{ MHz} \times 32 \text{ bits} \times 2 \text{ (DDR)} = 4.16 \text{ GBps}$
  - Where’s the 512 bits?
  - Constant Bandwidth $\Rightarrow F_1 \times Bits_1 = F_2 \times Bits_2$
  - Burst from DRAM and move wide data at low freq.
- DRAM Latency: Measured from ARM Core $\sim 65\text{ns}$
Part 3: MMM

• Do you know what ops/second mean? $2 \times N^3$
• Are you able to time your code? Many ways...
  – gettimeofday from sys/time.h
  – highresolutionclock from chrono
  – Profiling from OpenCL events (not any time)
  – clock from time.h (can be platform dependent)
• Cannot measure between enqueues directly
• Memory transfer overhead isn’t significant, why?
  1. There is no actual transfer – only one DRAM
  2. Other Overheads – Cache flush + enqueue/finish
What about performance?

```c
for (int row = 0; row < dim; row++)
    for (int k = 0; k < dim; k++)
        for (int col = 0; col < dim; col++)
            C[row][col] += A[row][k] * B[k][col];
```

- This can get you to \(~200\) Mops/sec at 150 MHz
  - How many MACs per cycle is that?
  - How much memory bandwidth is that?
- So much parallelism – where’s the performance
  - Vast fabric – use more MACs in parallel
  - How to feed data into all the MACs?
- You’ll explore this Lab 2 but let’s see an example
Blocked Matrix-Matrix Mult.

for (kk = 0; kk < n; kk += bsize)
  for (jj = 0; jj < n; jj += bsize)
    for (i = 0; i < n; i++)
      for (j = jj; j < jj + bsize; j++) {
        sum = C[i][j];
        for (k = kk; k < kk + bsize; k++)
          sum += A[i][k]*B[k][j];
        C[i][j] = sum;
      }

- What did we do here and why does it work?
- Would it work on the FPGA?
Improving Arithmetic Intensity

for (kk = 0; kk < n; kk += bsize)
   for (jj = 0; jj < n; jj += bsize)
      // Copy block of B into BRAM
      for (i = 0; i < n; i++)
         for (j = jj; j < jj + bsize; j++) {
            sum = C[i][j];
            for (k = kk; k < kk + bsize; k++)
               sum += A[i][k]*B[k][j];
            C[i][j] = sum;
         }

• Explicitly build the memory system to suite needs
• CNN has much better reuse than MMM
Let’s Review: Bandwidth & Latency

- What is throughput and latency?
- Why is throughput $\neq 1/\text{latency}$?
- How can we measure these quantities? What the best/worst case for each?
Measuring Bandwidth

• Know what bandwidth you’re measuring!
  – Generating transaction – hardware ports - DRAM

• Given you don’t change the hardware module interfaces and settings can you reach 4 GBps?
  – 150 MHz × 32 bits = 600 MBps

• Reading Column-by-Column
  – If you do not buffer, must be slower! Why?
  – No burst reads (wasting 60 bytes) + skipping row buffers (DRAM is not true random access)
  – ~ 70 MBps under the given constraints
What the code might look like

```c
int val = 0;  // Initialization
for (int row = 0; row < size; row++) {
    for (int col = 0; col < size; col++) {
        val |= in1[row * size + col];
    }
}
out_r[0] = val;
```

- Force meaningful computation
- If you didn’t initialize, what did you compute?
- Flip the loops to get column-by-column access
  - You’re not measuring the right thing if you get the same bandwidth number
Beware: Loop Flattening

for (int row = 0; row < size; row++) {
    for (int col = 0; col < size; col++) {
        for (int rc = 0; rc < size * size; rc++) {
            int row = rc / size;
            int col = rc % size;

            // Code here
        }
    }
}

• Helps reduce loop enter overhead and pipelining
• Can lose loop ordering (associative operations)
• Only when you really want to measure something
  – #pragma HLS loop_flatten off
What about Write Bandwidth?

```java
for (int row = 0; row < size; row++) {
    for (int col = 0; col < size; col++) {
        out_r[row * size + col] = size;
    }
}
```

- Much easier – no sneaky optimizations
- Numbers actually look very similar (600 & 70)
  - Bound not by DRAM bandwidth
  - Port Bandwidth and Access Pattern
The trickier one: Latency

- Which one(s) do you think are actual latency numbers?
- Latency ≠ 1/Throughput
- What did we want to measure?
  - The best latency for one int read
  - Read same memory location
  - From a kernel on the FPGA
- It’s an FPGA (HP Port)
  - No cache, no pages, no anything
  - Plain and simple memory

<table>
<thead>
<tr>
<th>Self-Reported Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.043</td>
</tr>
<tr>
<td>7.026</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>35.76</td>
</tr>
<tr>
<td>68.7</td>
</tr>
<tr>
<td>89.124</td>
</tr>
<tr>
<td>102</td>
</tr>
<tr>
<td>126</td>
</tr>
<tr>
<td>480</td>
</tr>
<tr>
<td>554</td>
</tr>
<tr>
<td>1,972.75</td>
</tr>
<tr>
<td>28,900</td>
</tr>
<tr>
<td>5,600,000</td>
</tr>
</tbody>
</table>
What are we measuring?

- HLS Kernel
- AXI SmartConnect
- SoC System
- Controller
- ARM Cores
- Fabric Clock (Def: 150 MHz)
- Core Clock (1.2 GHz)
- DRAM Clock (533 MHz)

Around 40 cycles!
How can we measure it?

```c
int val = 0;
for (int i = 0; i < count; i++) {
    val = A[val];
    // Need to wait for value A to come in!
}
out_r[0] = val;
```

- Pointer chasing! A[] loaded with 0s
  - Not so unnatural – think linked-list/graph traversal
- Depends on fabric clock frequency
  - 150 MHz – 480ns; 400 MHz – 320 ns...
- What if we wanted to measure from the CPU?
DRAM Latency: ARM Core (~65ns)

```c
int val = 0;
for (int i = 0; i < count; i++) {
    val = A[val]; // A = {16, 32, 48, 64, ... 
    // Need to wait for value A to come in!
}
out_r[0] = val;
```

- Few ns latency measured, what!?  
  - Enter cache; Need to avoid hitting the cache  
  - Strided access with stride = cacheline size (64B)
- Not an FPGA! OS, Virtualization, Pages... sigh...  
  - Incur TLB misses along with row buffer misses  
  - But 4 KB and 8 KB in size... 1 in 64 times...
Parting Thoughts on Lab 1

• Keep in mind the pitfalls for low-level measurements
  – Know what you’re measuring
  – And are you measuring it right?
• FPGA is slow (lower frequency) but wide data
  – Poor latency but comparable bandwidth
• Keep in mind these numbers going forward
  – Know what’s the best you can do – the limit
  – Sanity check measurements against these
• Onto Lab 2 then!
PROJECT PROPOSAL: MUCH CLOSER THAN IT APPEARS
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
Examples of Applications

• Irregular Computations include
  – Sparse Graph/Matrix (SpMV, graphNN, traversal (bfs...), counting, signal processing, ...)
  – String Matching (genome sequence, regex,...)

• Regular Computation but not throughput
  – Low-latency/Power/Precision (Lot of ML...)
  – Large Matrix Multiplication (imp. design space)
  – Stencil Computation (Compute vs Communication)

• Streaming Applications
  – Video processing (real-time video proc., rendering)
  – Networking Apps (compression, security...)

Examples of Metrics

• What you use to quantify your output

• Throughput, Latency, Power, Resource Utilization
  – Most basic applicable across application domains
  – Could take various forms: Ops/sec, Frame Rate, Memory Bandwidth, MTEPS, etc...
  – Composite Metrics (J/op, Perf/W, EDP, FoM...)

• Other metrics niche and domain specifics
  – Accuracy of results (ML/CNN)
  – QoR Metrics: Image/Video Resolution
  – You probably know your domain better than us
Examples of Design Spaces

• What you have control over. Some examples:
  • Generic Design Spaces
    – Performance (any form) – Resource Tradeoffs
  • Genome Sequence Alignment
    – NoC and NoC Topology to send data to PEs
    – Number of PEs and hence NoC endpoints
  • Stencil Computation (Compute vs Comm.)
    – Size of stencil vs size of overlap
  • Input Formats: Sparse Storage formats, Sparsity, Input Resolution, Matrix size, ...
Parting Thoughts

• As much/more about the proposal vs execution
  – Hypothesize – Execution is only to prove/disprove
  – Not working ≠ Failure, be able to answer: why?
• Coming out of the project – Be able to reason about the design space backed by results
• It takes time to iron out the specifics even if you have your application and design space in mind
• Fill the weekly questionnaire with any thoughts
  – Talk to us and get feedback early
  – You know what you’re going to be assessed on