

**University of Pennsylvania**  
**Department of Electrical and System Engineering**  
**System-on-a-Chip Architecture**

ESE5320, Fall 2023 Intermediate Throughput Milestone Wednesday, November 15

---

**Due:** Friday, December 1, 5:00PM

**Group:** Achieve target speed and functionality and writeup progress

1. Accelerate portions of your deduplication and compression task; running on real-time network input. From last week, you should have network I/O in place with a fully functional design. The goal here is to achieve high throughput on some portion of your pipeline (e.g.  $> 100$  Mb/s on functional CDC, SHA, deduplication pipeline with a simplified placeholder for LZW so that it does not limit throughput). This should demonstrate you are able to add functionality to the provided I/O setup and maintain high throughput.
  - (a) Describe the portion of the pipeline you accelerated. What parts are fully functional? What parts are simplified placeholders?
  - (b) Report throughput achieved. Include details on the throughput supported by each major operation as well as the overall throughput.
  - (c) Report current compression rate achieved.
  - (d) Describe all validation performed on your accelerated implementation.
  - (e) Describe where each component runs and the resources it uses.
  - (f) Identify where this design is in your design space. Explain additional design-space axes beyond your previous milestone as necessary.
  - (g) Describe your next steps to accelerate fully functional versions of the remaining portions of the pipeline and further accelerate your components.
  - (h) Support your description with a performance model.
  - (i) Describe who did what.
2. Turn in a tar file for your code above to the designated assignment component in canvas (one per group).
3. Turn in a tar or zip file with binaries to support execution of your code to the designated assignment component in canvas (one per group).
  - (a) `encoder.xclbin`, `BOOT.bin`, `boot.scr`, `image.ub` for FPGA kernel
  - (b) `encoder` for OpenCL host code executable

- (c) `decoder` executable configured to work with your encoded file and that can be run on the Ultra96. (Most likely, this is just a compilation of the `Decoder.cpp` we supplied; however, if you chose a different maximum block size, you may need to change `CODE_LENGTH`; so give us back one with that change made.)  
Make sure to compile it with the `aarch64-linux-gnu-g++` compiler and test it on the Ultra96. While you could run the decoder on your host machine (which could be Linux/Mac OS/Windows), we will run your decoder on the Ultra96.
- (d) `client.sh` shell script to invoke client with suitable `-s` parameter to demonstrate your guaranteed data transfer performance.