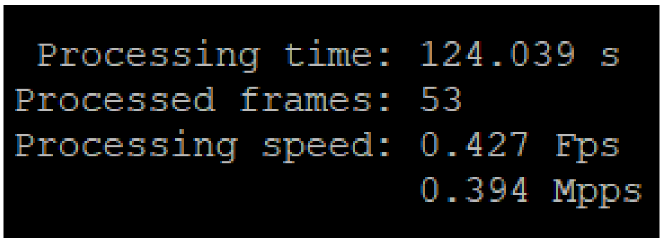
Hardware/Software Codesign – Project part 1

João Pedro Reis Teixeira

The project started by setting up the board according to the given guide. The notebook *hdmi_introduction* on the board was used as a reference to port the code to PYNQ's PS. The *BaseOverlay* class initialized the base overlay at the start. HDMI input replaced the camera input using *config* and *start*, with video parameters set via the *mode* attribute. HDMI output was configured using the *VideoMode* object for 1280x720 resolution, 24-bit BGR format, and 60 FPS, then added with *config* and *start*. The *readframe* and *writeframe* functions handled HDMI input and output. To handle frame format changes, the *newframe* function created an empty frame, then the old contents were copied to it, and written to output. Finally, the *close* function shuts down both HDMI connections. After this, the performance of PS was measured and profiled. The measurements, executed with profiling disabled, can be seen in Figure 1. The profiling results are shown in Figures 2, 3 and 4.



```
 Processing time: 124.039 s
Processed frames: 53
Processing speed: 0.427 Fps
                  0.394 Mpps
```

Figure 1: Measurements of PS version without profiling

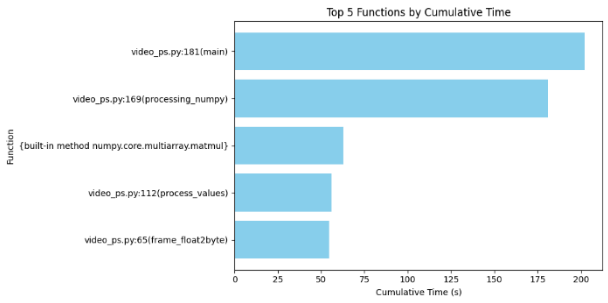


Figure 2: Profiling top 5 functions by cumulative time

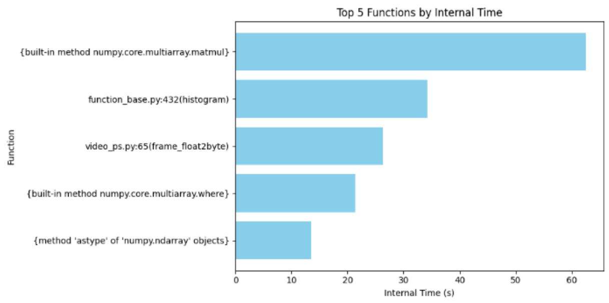


Figure 3: Profiling top 5 functions by internal time

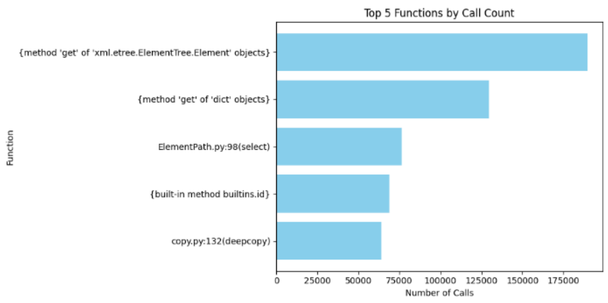


Figure 4: Profiling top 5 functions by call count

As shown by Figure 1, the frames per second are extremely low, at 0.427 *Fps*. This implementation is then far from being suitable for real-time playback and enhancement. The best option to accelerate the program's execution is to make a good split between PS and PL. FPGAs (PL) are very efficient in highly parallelizable and data-intensive tasks, while processors (PS) are better at handling complex control logic and iterative processing. This means that we can execute in PL functions like *frame_convert_BGR2YCrCb* and *frame_convert_YCrCb2BGR* (and the corresponding *pixel* versions) because, being matrix multiplications, they are highly parallelizable and data-intensive. This would greatly reduce the execution time, considering *numpy.matmul*, responsible for the matrix multiplication for both functions, has the highest internal execution time and the third highest cumulative time. The function *frame_float2byte* involves simple arithmetic operations applied to each pixel. The uniformity and simplicity of these computations makes this function suitable for PL. Its internal time is also the third highest, meaning a reduction of it would significantly improve performance. Two other functions to execute in PL would be *process_values* and *transform_update*. The transformation of Y channel values involves calculating and applying histograms, which is a data-parallel task that can benefit significantly from the FPGA's ability to process multiple pixels simultaneously. The remaining functions should not benefit from execution in the FPGA and, if they do, it would be a small increase in performance which should not be very relevant, so their execution can remain in PS.