To measure the reference OpenCV execution speed, I used the *TickMeter* class because it is part of the OpenCV core. I figured it would be better to use standardized methods than to roll my own. By using the TAPI introduced in OpenCV 3, I only had to make one change to my existing host code to run it on my iGPU: change the *Mat* data type to *UMat*. This data type is a generic one that can hold the data in the VRAM behind abstraction. I ran the reference programs on my laptop with a Ryzen 5 2500U with integrated graphics.

For running the OpenCV reference program, I isolated four CPU cores from the OS scheduler and dedicated them to the program. I also measured the performance of the test implementation (aka ”Lane Program”, I didn't have much inspiration for its name) that I made during the research phase of the project. This program is executed on the CPU and is only capable of using a single thread. For running the Lane Program, I dedicated one of the isolated cores to it. Except for the raw FPGA implementation, for which we can calculate the latency exactly, I ran all tests 500 times and calculated the average run time and standard deviation between run times.

Platform	Average (ms)	Standard Deviation (ms)
FPGA (raw)	11,2	-
FPGA (co-processing)	26,6	0,443
Reference OpenCV (CPU)	72,2	1,801
Reference OpenCV (GPU)	14,2	0,794
Reference Lane Program	382,7	13,81

Table 9: Performance comparison per platform

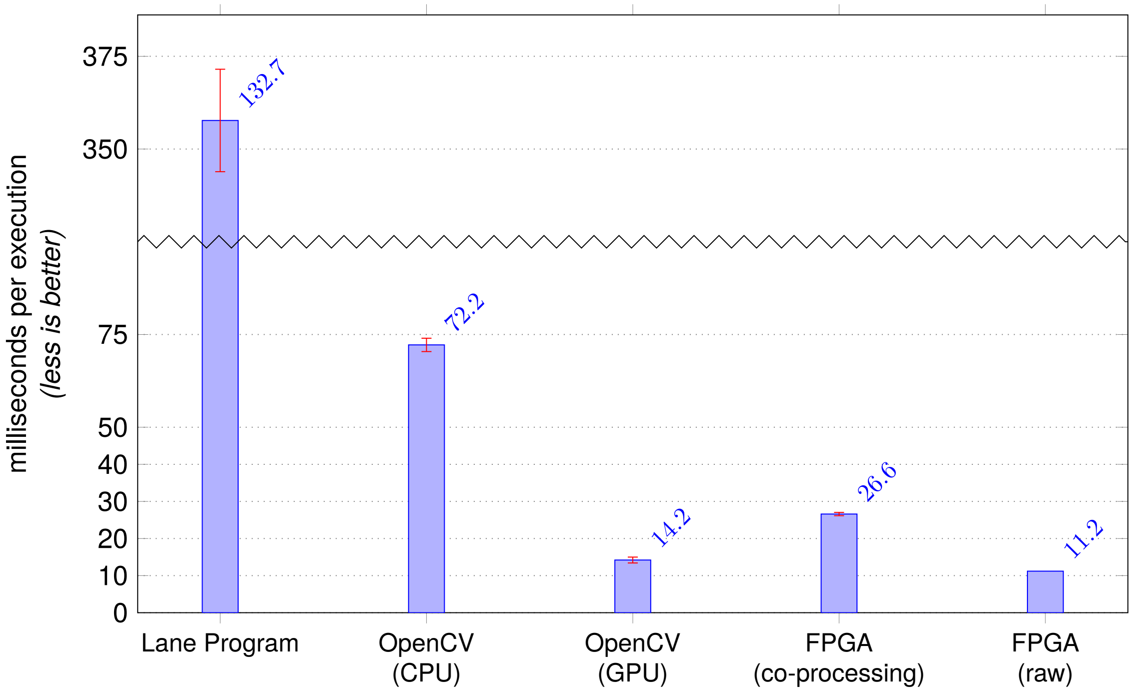

Figure 26: Bar chart visualization of Table 9