Hardware für Eingebettete Systeme

Lab 10: GPU Programming with CUDA (2/2)

Prof. Dr. Mario Porrmann
M.Sc. Marc Rothmann
B.Sc. Philipp Gehricke

1 Matrix Multiplication in CUDA

Once again, you are going to program using CUDA. In the last lab exercises you programmed a vector addition where you used basic CUDA extensions. Now you should use the same basic instructions or even more advanced techniques to program a Matrix Multiplication for the embedded GPU on the Jetson Nano. But this time you won't be provided with any basic frame for your code. This gives you maximum freedom for your implementation. Again, this guide provides reliable information on this topic and there are also many more sources for matrix multiplication using the CUDA API.

However there are a few constraints and tasks you should fulfill:

- Your program should randomly initialize square matrices of any desired size
- Multiply the matrices on the CPU
- Multiply the matrices on the embedded GPU
- Implement some kind of time measurement. (e.g. like the one in the previous lab)
- Verify your results: Does the GPU implementation generate the same results as the CPU implementation?
- Your code (on the GPU) must not be slower than 100 ms for an input size of 1000 colums and 1000 rows! (Or 10 seconds for an input size of 10000 * 10000)
- Test your implementation for different matrix sizes
- Take note of your observations and use the tools introduced in the last lab exercise to document your results. (e.g. runtime, energy consumption, memory management) Compare the measurements between your CPU and your GPU implementation. You can safely overclock the Jetson Nano by using '`jetson_clocks`' within '`jtop`'. This would improve the runtime. Is overclocking worth it with respect to performance per Watt?