Prof. G. Zachmann
Christoph Schröder (schroeder.c@cs.uni-bremen.de)

University of Bremen
School of Computer Science
CGVR Group
September 14, 2020

Summer Semester 2020

# Assignment on Massively Parallel Algorithms - Sheet 2

Due Date

## Exercise 1 (Moore's Law and Power consumption, *Credits*)

a) Consider two approaches of doubling the number of transistors: halving the size of a single transistor while maintaining constant die area (Moore's Law) versus maintaining the size of a single transistor while doubling the die area. List at least three reasons why the first approach is superior to the second approach.

b) The idealized formula for energy consumption by a processor core is

$$E = ctfV^2$$

where $c$ is a CPU-dependent constant, $t$ is total execution time, $f$ is the processor's clock frequency, and $V$ is the supply voltage. The frequency and voltage are correlated as follows

$$f = \alpha V$$

with $\alpha = 0.2 \cdot 10^9$ $\texttt{HzV}^{-1}$. Suppose our algorithm must complete in $t = 10$ seconds and needs a total of $10^{10}$ clock cycles to execute. What is the CPU energy consumption $E$ for one task that completes in $t = 10$ seconds? What is the energy saving ratio when we run this perfectly parallelizable algorithm in two tasks on two CPU cores in parallel, assuming each task takes $t = 10$ seconds to complete?

## Exercise 2 (Amdahl's law, *Credits*)

Given a single core processor A and a multi-core processor B with N cores. Additionally, assume that all cores of A and B are identical.

a) Given a program that runs 1.7 times faster on processor B than on processor A. Compute the parallel portion of the program i.e. $f = P/(P + S)$ with $P$ = execution time of parallizable part on single processor and $S$ = execution time of inherently serial part on single processor (see Slide on Amdahl's Law (the "Pessimist") in the Introduction Chapter).

b) Suppose parallel portion $f$ is 0.5, how many processor cores are needed to achieve an overall speed up of 1.6?

## Exercise 3 (CUDA basics: Launching kernels, *Credits*)

Starting from the framework `myFirstKernel`

a) Allocate device memory for array `d_a` to hold the results of the kernel.
Overall `numBlocks`×`numThreadsPerBlock` threads will be launched, and each thread writes to one array element.

b) Configure and launch the kernel `myFirstKernel(int *d_a)` using a 1D grid of 1D thread blocks.

c) Have each thread set an element of `d_a` as follows:
```
idx = blockIdx.x*blockDim.x + threadIdx.x
d_a[idx] = (blockIdx.x - 6) * (100 - threadIdx.x)
```

d) Copy the result in `d_a` back to the host memory to array `h_a`.

e) Free the device array `d_a`

f) Cuda kernels cannot return a value. What could be the reason for this?

g) Measure the kernel execution time using cuda events. Consider the following types and functions which are described further in the CUDA docs:
```
cudaEvent_t event;              float milliseconds = 0;
cudaEventCreate(&event);        cudaEventElapsedTime(&milliseconds, start, stop);
cudaEventRecord(event);         printf("CUDA time: %f\n", milliseconds);
cudaEventSynchronize(event);
```