1. Problem Statement

The context of this thesis is a private mobile radio infrastructure. It is compliant with the controversial ETSI specification EN 300 392, defining the widely employed Terrestrial Trunked Radio System (TETRA). Targeted at public service institutions, such as police, fire departments or ambulance, the system is designed to provide reliable speech transmission for its subscribers. As a modern standard, its speech transmission method is digital for reasons of bandwidth economy and uses the according TETRA voice codec. The search for an economic unit capable of processing large numbers of speech channels led to the idea to run the TETRA codec on a current streaming platform (GPU). The central question therefore is:

How many channels of the TETRA speech codec can be encoded in realtime on current standard graphic cards?

2. Speech Coder

Processing intensive ACELP (Algebraic Code Excited Linear Prediction) encoding method.

3. Nvidia GeForce 200 GPU

GeForce GT200 series with unified shader model and CUDA API for general purpose applications.

3.1 Architecture

- Containing multi-processors (MPs) executing shader programs, accessing device memory.
- Multiprocessors: extremely lightweight threads collaborating as vector unit. Thread groups execute shader programs with the same operation on multiple operands. Operands must be loaded to fast shared memory for efficient processing.

3.2 GPU vs. CPU

GPU outmatches CPU in terms of FLOPS only if certain requirements are met:
- memory access patterns, alignment
- enough load for thread scheduler
- branching in shader programs

4. Method

A single GPU thread cannot process in realtime => Problem-inherent data parallelism must be exposed.

1. Which algorithm stages are the most runtime-intensive?
2. Measure runtime proportions.
3. Optimize most intensive stages.

Chart shows runtime distribution among the ten most complex stages.

Most stages are of similar problems.

Recursive filters of the form

\[ y[n] = \sum_{i=1}^{N} a_i y[n-i] \]

are unsuited for parallel processing but can be substituted by convolution with their impulse response.

Convolutions can then be decomposed into equally-sized subconvolutions well suited for parallel processing.

5. Results

Optimized parallel implementation of the given algorithm stages allows real-time encoding of at least 3 channels per MP.

With 30 MPs (e.g. GT275 card), this is 90 channels per GPU.

6. Conclusion

The potential of GPU use in the context of speech coding has been evaluated. A common CELP algorithm is analysed for its suitability to parallel processing.

While some aspects of the coder can be exploited, generally, the algorithm’s structure makes the GPU adapt a hard task.

However, data parallelism arising from the processing of many channels at once, allow the GPU to deal with a substantial number of channels.