Parallelizing H.264 for the Cell Broadband Engine is an ongoing work of the TU-Berlin. We have successfully parallelized the macroblock processing part with great performance and scalability results. One of the next steps is to move the entropy decoding part of H.264 to the Synergistic Processing Element (SPE) in order to parallelize it further.

Cell Broadband Engine

The Cell Broadband Engine [5] is a heterogeneous multi-core consisting of one PowerPC Element (PPE) and eight SPEs. The PPE is a dual-threaded general purpose PowerPC core with 512 kB L2 cache. Its envisioned purpose is to act as the control/OS processor, while the eight SPEs provide the computational power. Figure 2 shows a schematic overview of the Cell processor. The processing elements, memory controller, and external bus are connected to an Element Interconnect Bus (EIB). The EIB is a bi-directional ring interconnect with a peak bandwidth of 204.8 GB/s [2]. The XDR memory can deliver a sustained bandwidth of 25.6 GB/s.

What makes the Cell such an innovative design is not its heterogeneity, but its scalable memory hierarchy. In conventional homogeneous multi-core processors, each core has several layers of cache. The cache provides ample speedup, because it reduces the average latency and bandwidth usage of the external (off-chip) memory. With multiple cores there are multiple caches and coherency actions are required. The cache coherency actions grow with a complexity of $O(n^2)$ with increasing core count, which quickly become unpractical in many-core architectures. In the Cell
architecture, the SPEs do not feature a cache and rely on a local store and DMA unit instead for access to the memory. Each SPE has a local store of size 256 kB. The SPEs can only work on data in the local store. The programmer is responsible for the data transfers using explicit DMA operations. The programming style is that of the shopping list model. Instead of loading every data item separately at the time it is needed (as is the case with cache based systems), all data required for a task is brought in at once and before execution of the task. Moreover, loading the data of one task should be done concurrently with the execution of another task in order to fully hide the memory latency.

**H.264 Decoding**

Currently H.264 [1, 6] is the best video coding standard in terms of compression rate and quality [4]. Also, it is the most widespread standard for digital video. It is used in Blu-ray, digital television broadcast, online digital content distribution, mobile video players, etc. The compression rate is over two times higher compared to previous standards, such as MPEG-4 ASP, H.262/MPEG-2, etc. H.264 uses the YCbCr color space with mainly a 4:2:0 subsampling scheme. In this subsampling scheme the luma component (Y) has the same resolution as the frame, while the chroma components (Cb and Cr) are at a quarter resolution. Throughout the paper this subsampling scheme is assumed.

In this bachelor assignment(s) we will focus on the decoding part of H.264, of which the block diagram is depicted in Figure 3. In the entropy decoding the data of the MBs is extracted from the H.264 stream. The macroblock kernels use the extracted data to decode the MB. At the moment the macroblock kernels have already been ported successfully to the Cell architecture using the widespread FFmpeg open source codec as a base. What remains is porting and parallelizing the entropy decoding.

**Requirements**

What is required for this assignment is a good understanding of C and experience in going to large pieces of code. Also some fondness towards programming is highly recommended. Furthermore, any experience in programming the Cell processor is useful, but not required.

You will be working on the FFmpeg [3] code that has already been parallelized to use the SPEs for
the macroblock kernels. We are able to play a H.264 movie on our Playstation 3 while using the SPEs. The Playstation 3 will also be your development environment. An introduction to the FFmpeg code including the compile, run, debug and play procedure is provided, so this part you do not have to figure out yourself. Any questions about code specific to the Cell we can also explain. This also holds to a certain degree for the FFmpeg code, however, no guarantees can be made in this regard as the code is pretty vast.

**Porting the entropy decoding to the SPE**

The assignment in this bachelor thesis is to port the entropy decoding kernel to the SPE. In order to run the entropy decoding on more than one core on the Cell processor, the SPEs have to be used. However “normal” code that runs on a cache-based processor core, does not run on a LS-based SPE. You will face the challenge of programming with a new programming paradigm in which you also have to take the data flow into account.

**Literatur**


