SEVENTH FRAMEWORK PROGRAMME
THEME ICT-2009- Call 4- 3.6
Computing Systems

Project acronym:

2PARMA

Project full title:

PARallel PAradigms and Run-time MAnagement techniques for Many-core Architectures

Deliverable D3.2.1: Initial report on the efficiency analysis of parallel programming models for many core architectures

Version [1.1]

Delivery due date: M18 (June 2011)
Actual submission date: July 25, 2011
Lead beneficiary: HHI

Dissemination Level of Deliverable		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	
Nature of Deliverable		
R	Report	X
P	Prototype	
D	Demonstrator	
O	Other	

Author(s):	*Jens Brandenburg (HHI)*		
Reviewer(s):	*Alex Bartzas (ICCS), Giovanni Agosta (POLIMI)*		
WP/Task No:	WP3/ Task 3.2	**Number of pages:**	28
Identifier:	D3.2.1_V1.1	**Dissemination level:**	PU
Issue Date:	July 25, 2011		

Keywords: Efficiency analysis, Profiling, Tracing, NoCTrace

Abstract: This deliverable reports on the activities carried out in **Task 3.2** on the efficiency analysis of parallel programming models for many core architectures. For this analysis, the tools and methodologies developed in 2PARMA **Task 3.1** have been used, in particular the profiling and tracing tool called NoCTrace and released as **D3.1.1.** Although the approach is quite general, the use case we analyzed in this report is a Scalable Video Coding SVC decoder running on STM's platform P2012 using a component based programming paradigm. In this report we analyzed different features of the SVC decoder and present different metrics with respect to the programming model to provide optimizations guidelines for future versions of the application.

A final report on the efficiency analysis of parallel programming models for many core architectures will be released as **D3.2.2** at M30 at the end of Task 3.2.

Approved by Project Technical Manager [illegible] **Approved by the Project Coordinator:** C Silvano	**Date: July 25, 2011**

Project Technical Manager: Prof. Dr. William FORNACIARI – Politecnico di Milano
e-mail: fornacia@elet.polimi.it - **Phone:** +39-02-2399-3504- **Fax:** +39-02-2399-3411

Project Coordinator: Prof. Dr. Cristina SILVANO – Politecnico di Milano
e-mail: silvano@elet.polimi.it - **Phone:** +39-02-2399-3692- **Fax:** +39-02-2399-3411

Table of contents

I. Executive summary

This deliverable contains a report on the efficiency analysis of parallel programming models for many core architectures. As a use case to perform this analysis we run a parallel version of a Scalable Video Coding SVC [1] decoder on STM platform P2012 [2]. The implementation of the SVC decoder is based on a CBSE tool-chain called MIND [3]. The analysis is performed with the help of the tools and technologies developed in 2PARMA Task 3.1, in particular with the profiling and tracing tool described in 2PARMA deliverable D3.1.1 [4].

This deliverable is organized as follows:

- In Section II the profiling methodology is described, that has been used to provide the profiling and trace data presented in this deliverable.
- In Section III we describe the use case for this deliverable, which is based on the SVC decoder running on platform P2012. The SVC algorithm is presented and how the sequential SVC decoder implementation has been parallelized and mapped onto P2012, with the help of the MIND CBSE tool-chain.
- In Section IV the different profiling types are described, which can be created with profiling tool described in section II.
- In Section V the analysis of the use case presented in section III is performed. In this section the profiling and tracing data gathered during the platform simulation runs are shown and discussed. This section is structured into different sub-sections to discuss different aspects of the SVC algorithm and present different application metrics.

II. Profiling methodology

The NoCTrace tracing and profiling tool has been designed to support the development, optimization and architectural co-exploration of parallel applications running on many core architectures. The functionality of the tool can be split into two disjoint tasks, namely the collection of trace data and the processing of this trace data to create exploitable results. To reflect this separation of concerns and to enable a flexible solution, the tool has been split into a backend part for trace data collection and a frontend part for trace data processing. One of the main advantages here is that the frontend can process the trace data from different trace sources as depicted in Figure 1.

The NoCTrace backend has been required to collect the trace data in a non-intrusive way from a SystemC platform simulation model run. Therefore the trace data collection module, e.g. the NoCTrace backend, has been integrated with the SystemC library by embedding the data collection probes into the SystemC kernel itself. A more detailed description of the tools architecture and the backend implementation can be found in deliverable D3.1.1 [4].

In the current version of the P2012 SDK (version 2011.1), used for the development of platform P2012 applications, the SystemC/TLM platform simulation model can be only accessed as a binary executable. A source code version of the platform P2012 simulation model will be available in the next version of the platform SDK coming later. Therefore the NoCTrace backend implemented in deliverable D3.1.1 cannot be used to collect the necessary trace data. Instead the P2012 TraceLib has been used to create the results reported in this deliverable (see Figure 1).

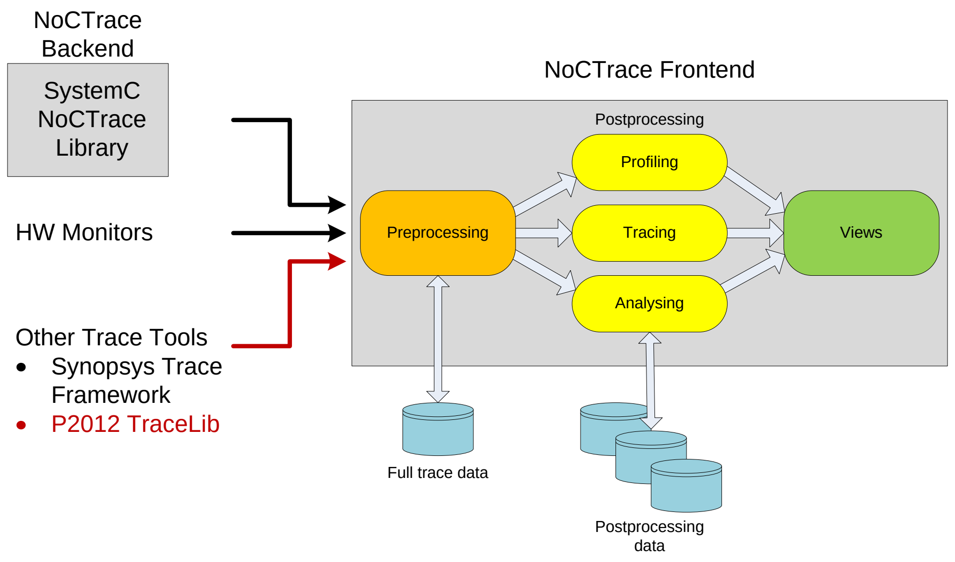


Figure 1: NoCTrace architecture overview

The P2012 TraceLib is based on an intrusive approach where the software developer instruments the application in order to collect the trace data. One of the main advantages of this approach is that the P2012 TraceLib can be used at different target systems, e.g. the TLM platform, the H/W emulator and the FPGA board, to collect the trace data. A disadvantage is of course that the trace data collection influences the application execution and that for each tracing event an additional macro has to be added to the source code of the application.

III. Use Case: Scalable Video Coding

III.1. SVC Algorithm

Scalable Video Coding SVC [1][5] is an amendment to the existing H.264/MPEG-4 AVC video coding standard. The AVC standard itself is based on the principle of block based hybrid video coding. Therefore the picture will be partitioned into block shaped units referred as macroblocks. Onto each macroblock a hybrid algorithm of motion compensation and transform coding is applied, to exploit temporal and spatial statistical dependencies.

Figure 2 shows the block diagram of an H.264/MPEG-4 AVC video decoder. The input bit-stream data, a sequence of Network Abstraction Layer Units NALU, will be entropy decoded to extract the transform coefficients of the residual signal and additional side information like macroblock modes, motion vector data or intra prediction modes. Onto the transform coefficients the inverse quantization and transformation functions are applied, to obtain the residual signal. Additionally a prediction signal is computed, either from a list of previously reconstructed pictures by the motion compensation functions or from the current picture by the intra prediction functions. Finally the loop-filter functions are applied to the reconstructed signal, to remove visible artifacts at block edges, often found in block based video coding.

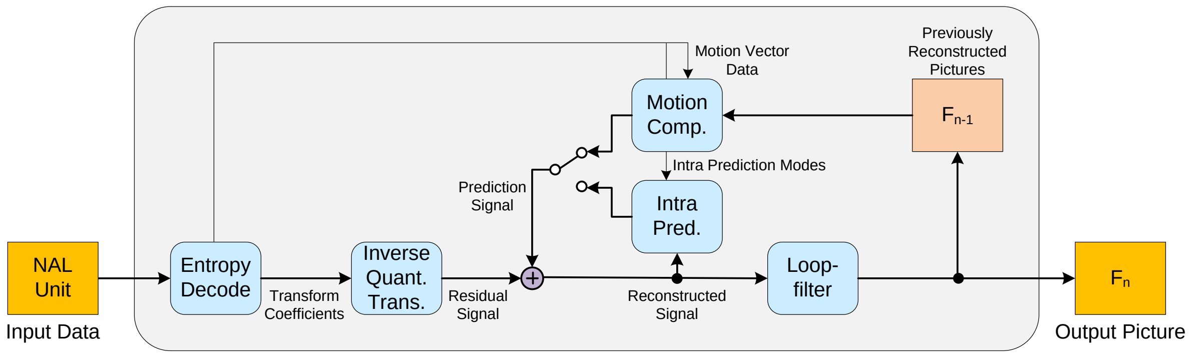


Figure 2: H.264/MPEG-4 AVC video decoder block diagram

SVC extends the AVC standard by a layered approach, to transmit various representations of the same video sequence over a single data link. In order to reduce the overall bit-rate of the bit-stream, the SVC standard introduces inter-layer prediction mechanisms, to use information of one SVC layer in another one. SVC distinguishes between three different modes of scalability, temporal scalability, quality scalability and spatial scalability. In case of temporal scalability different layers of the bit-stream represent different frame rates of the same video sequence. In case of quality scalability different layers of the bit-stream can be decoded with a different fidelity, which is typically measured in Peak Signal to Noise Ratio PSNR. Finally in case of spatial scalability different layers of the bit-stream represent different picture resolutions of the same video sequence.

Temporal scalability in SVC is realized by restricting the motion compensated prediction to reference frames that have a temporal layer index, which is equal or less than the predicted frame. In the example of Figure 3 four different temporal layers can be decoded with frame rates from 7.5 Hz up to 60 Hz. On the lowest temporal layer the motion compensated prediction is restricted to all frames with temporal layer index equal to 0. On the highest temporal layer all frames with temporal layer index 0 to 3 can be used for the motion compensated prediction.

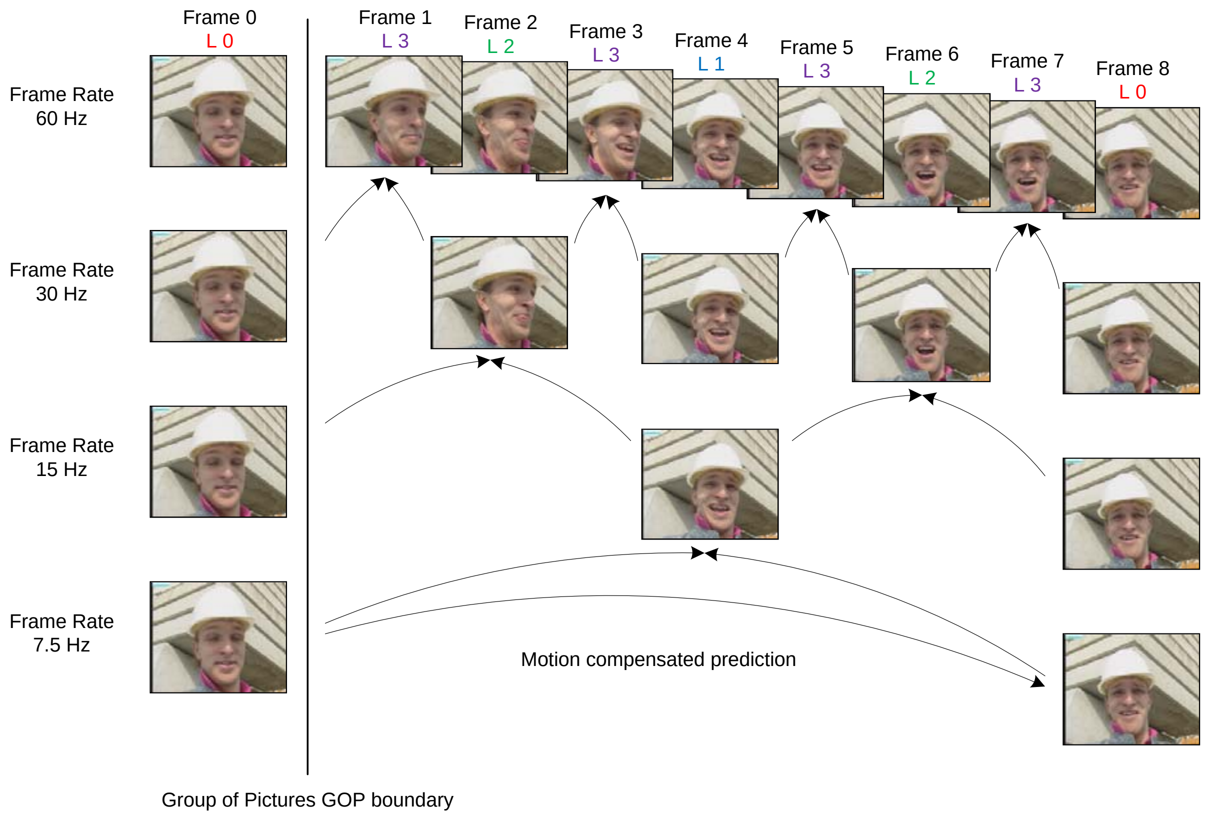


Figure 3: Temporal scalability, dyadic motion compensation

Whilst temporal scalability can be realized with tools already available in the AVC standard, for quality scalability new inter layer prediction tools have been introduced. According to an adaptive signal on macroblock basis, the residual signal of inter predicted macroblocks, the reconstructed signal of intra predicted macroblocks, the macroblock mode and the motion vector data can be reused in the SVC enhancement layer (see Figure 4).

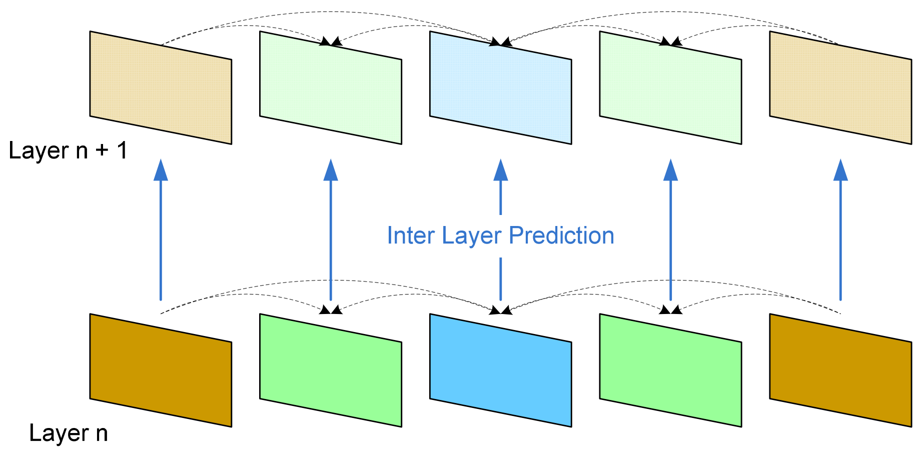


Figure 4: Inter layer prediction for quality scalability

A special feature of the quality scalability from SVC is that the combination of residual and intra inter-layer prediction signal can be performed in the transform domain (before applying the inverse transformation) reducing the overall necessary computational steps.

To use the same inter layer prediction tools introduced by quality scalability within spatial scalability, the base layer has to be up-sampled to the same resolution like the enhancement layer (see Figure 5). Of course now the combination of residual and intra inter-layer prediction signal has to be performed in the picture domain. Additionally the intra inter-layer prediction signal has to be loop-filtered, before it can be used in the up-sampling stage.

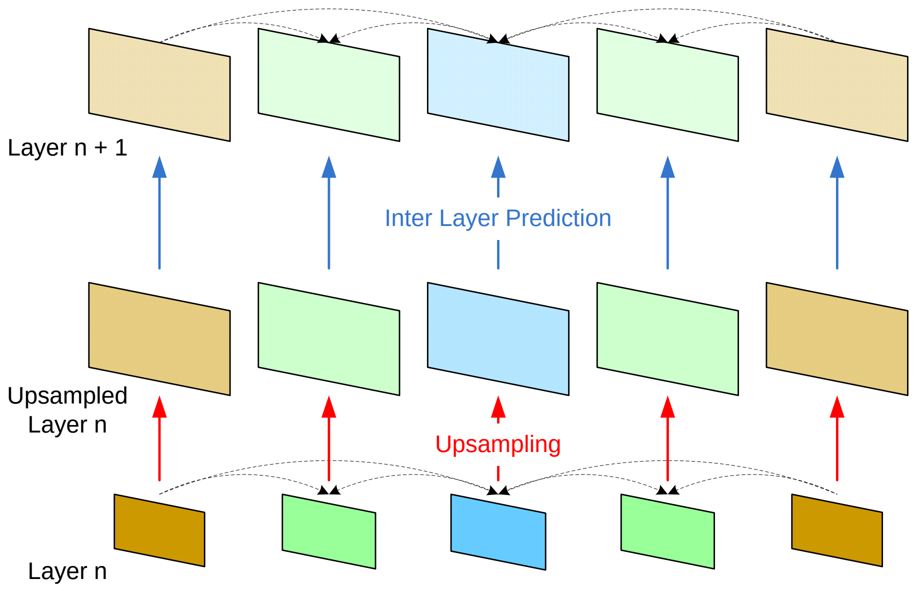


Figure 5: Inter layer prediction for spatial scalability

In order to reduce the additional computational effort to decode a quality or a spatial enhancement layer, the very resource demanding motion compensated prediction and loop-filtering has to be performed only once at the reconstruction layer. Nevertheless, it can be expected, that spatial scalability is the most resource demanding type of scalability in SVC.

III.2. SVC Decoder Application on Platform P2012

To port the SVC decoder application on platform P2012 the CBSE tool-chain based on the MIND component framework has been used. In combination with the Parallel Programming Pattern PPP component repository this framework provides a powerful tool to parallelize the application and map the components onto different processing elements of platform P2012.

The porting and parallelization has been implemented step by step, to simplify debugging and functional verification of the SVC decoder application. In a first variant the whole SVC decoder has been treated as one single component running on one processing element. In a second variant we have modified the internal loop structure, to prepare a split and parallelization of the SVC decoder. Finally in a third variant the SVC decoder has been mapped onto three different processing elements as explained below.

III.2.1. SVC Decoder Variant 1

In this variant the core SVC decoder is running on one processing element. Additionally two more components have been implemented, to function as the bit-stream data source and the decoded YUV frame sink. In the current version of the TLM model all processing has to be done on the P2012 fabric itself. Therefore no communication with the host and no file I/O can be used, to load the bit-stream data or store the decoded YUV frames. In this way the current source and sink component of the SVC decoder application are like stubs providing the required interfaces. Internally both components accessing constant memory array's, containing the input bit-stream data and a CRC checksum for each decoded YUV frame. This enables the source component to read the input bit-stream data directly from memory and pass the NALU's to the core SVC decoder. In the sink component a CRC checksum of the decoded frame is computed and compared with the pre-computed value, to check the functional correctness of the core SVC decoder. These three components have been linked together with FIFO components to enable a parallel processing in a pipelined approach.

III.2.2. SVC Decoder Variant 2

In this variant the same mapping has been used like in variant 1. The bit-stream source component has been mapped onto processing element 1, the core SVC decoder onto processing element 2 and the decoded YUV frames sink component onto processing element 3. In order to prepare a split and parallelization of the core SVC decoder component we have identified 3 main loops, consisting of parsing, reconstruction and loop-filtering, in the SVC decoder algorithm. These loops are processed over all macroblocks of all NALU's of one Access Unit AU. These loops, as implemented in variant 1, cannot be parallelized effectively, because the output of one loop is the input of the next one. But this dependency is only blocking on a macroblock level. This means after macroblock 1 has been parsed, the macroblock can be reconstructed. In the meantime macroblock 2 can be parsed and so on. Therefore we had to transform the original loops into a macroblock level interleaved loop as depicted in Figure 6. So this variant is an intermediate implementation to prepare the split and parallelization of the core SVC decoder algorithm.

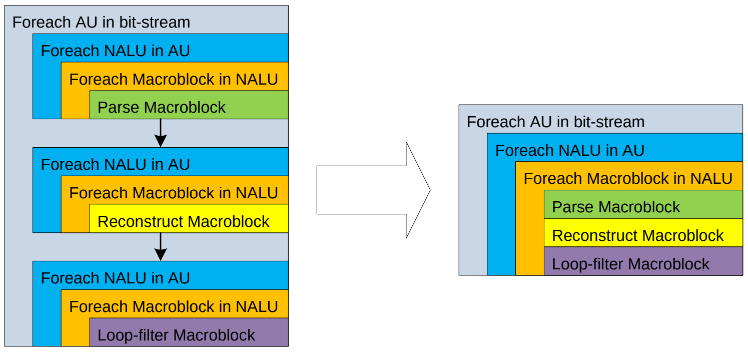


Figure 6: SVC decoder variant 2 loop transformation

III.2.3. SVC Decoder Variant 3

In this variant we have split the core SVC decoder algorithm from variant 2 into three sub-components that can be processed in parallel in a pipelined approach. The overall SVC decoder application consists in this variant of five components connected by each other with additional FIFO components (see Figure 7). Additionally components for booting, scheduling and synchronization have been used from the PPP repository to assemble the final application. The synchronization component at the bottom of Figure 7 is necessary to signal the parser, that he can continue the decoding of the next NALU.

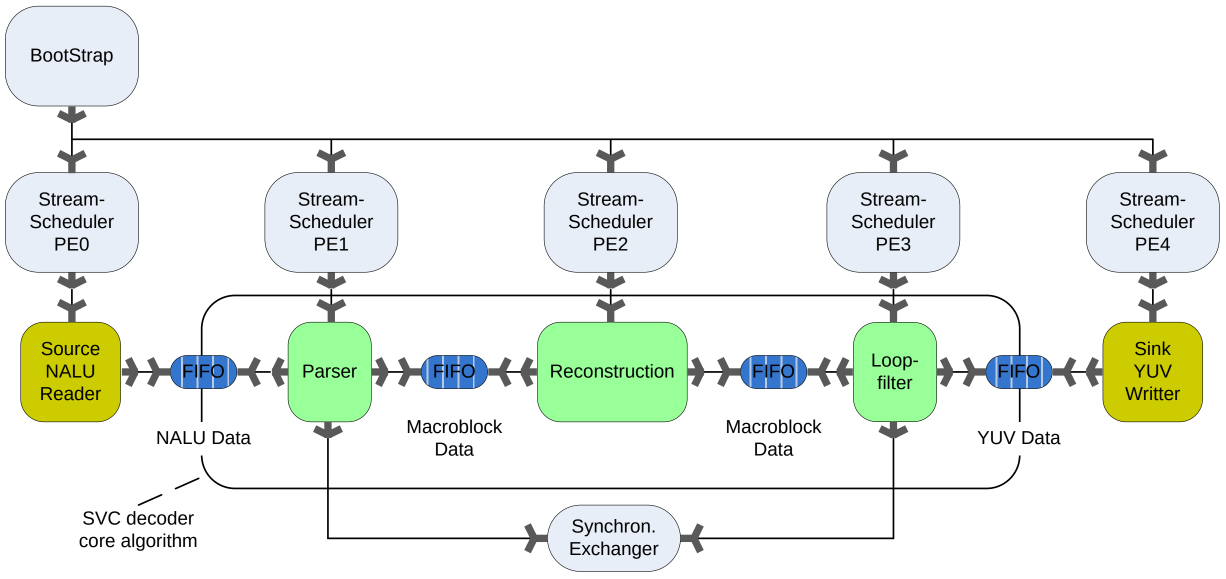


Figure 7: Component based SVC decoder

IV. Profile Types

In the following section the different profiles create by NoCTrace should be discussed. For this purpose the trace data generate with the help of the P2012 TraceLib has been processed with NoCTrace. The resulting post-processed data have been stored into profile data files and have been used to generate the presented graphs with the help of Gnuplot [6]. To generate the trace data for this section the following input bit-stream configuration has been used.

N	Name	Size and Rate	Frames	QP	Bit-rate	PSNR	Simul. Time	Real Time
1	BUS	176x144@15	16	34	152kBit/s	30.4	23.38s	5:03h

IV.1. Function profiles

The P2012 TraceLib is based on intrusive software traces, which have to be inserted into the application source code by the application developer. For this purpose, the developer can use low-level basic trace functions or he can define high-level custom trace functions, to be used within the application. A detailed description how to define and use customized trace functions can be found in chapter 4 of the P2012 tools book [7]. In summary the tracing framework provides four different types of customizable traces: complex traces, string traces, function traces and section traces. From these different trace types the function traces can be processed by NoCTrace, to generate a function profile for each instrumented function and processing element as can be seen in Figure 8.

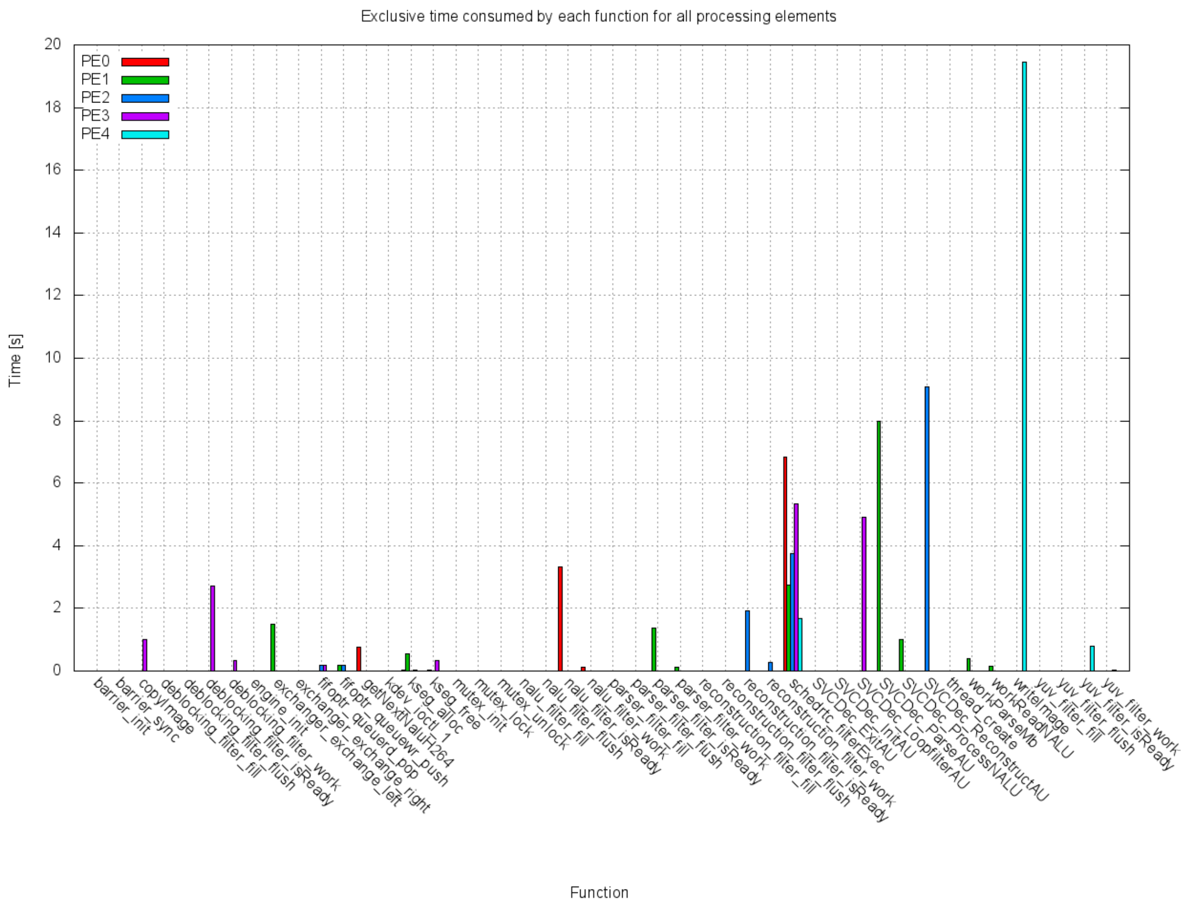


Figure 8: Exclusive time consumed by each function for all processing elements

The metric used to generate this function profile was an exclusive metric, meaning the profile shows the time spend in each function exclusively, without the time spend in the sub-functions. The same function profile can be also generated with other metrics presenting the inclusive time spend in each function (see Figure 9) or the number of calls for each functions.

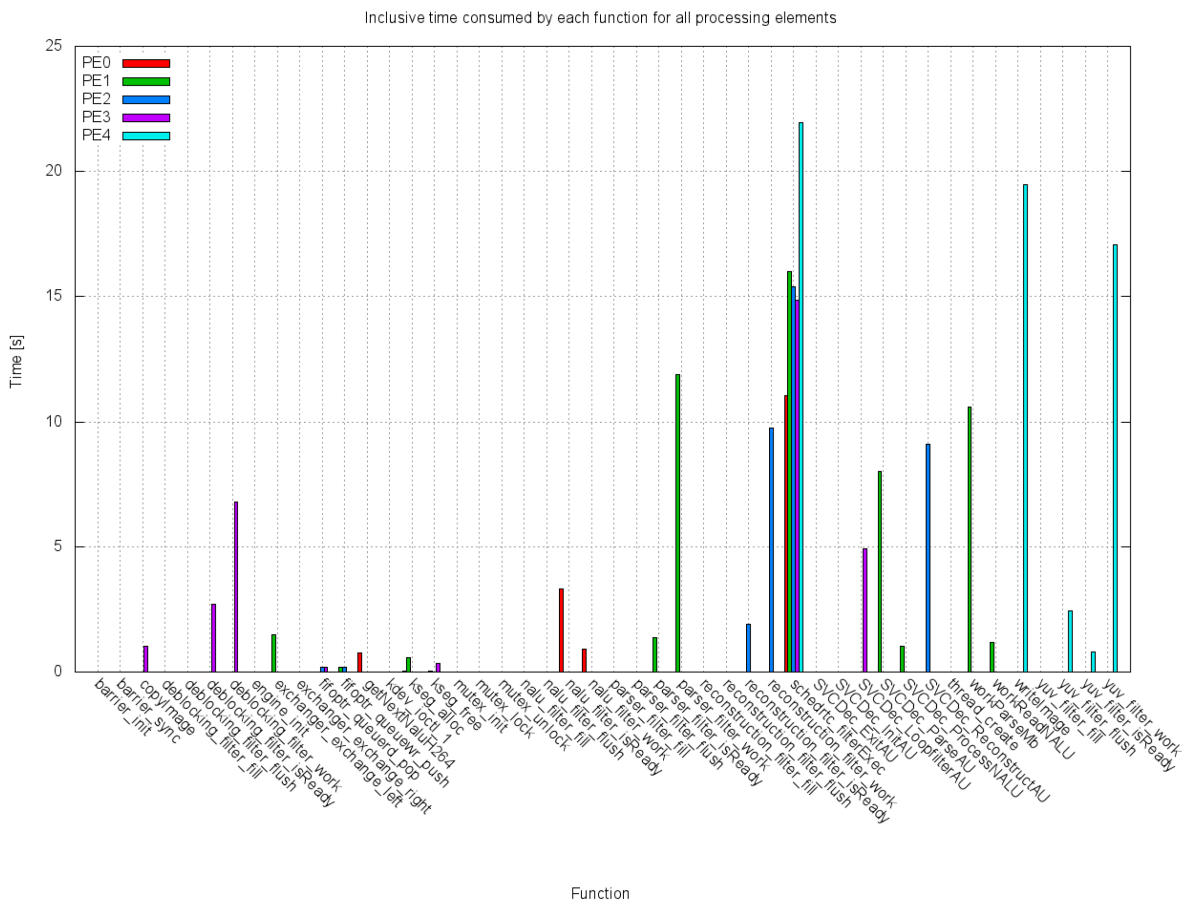


Figure 9: Inclusive time consumed by each function for all processing elements

Of course not all functions of the SVC decoder application have been instrumented. We have only instrumented the top-level functions of each component, to reduce the impact of the trace functions on the simulation execution. Additionally a set of PPP functions and the underlying NPL runtime functions have been pre-instrumented in the current SDK.

Such a total function profile can be used to perform a first hotspot analysis. The most time consuming function is `writeImage` on PE4 with an overall execution time of 19.46s, followed by `SVCDec_ReconstructAU` on PE2 with 9.09s, `SVCDec_ParseAU` on PE1 with 7.99s and `schedrtc_filterExec` on PE0 with 6.81s.

The functions `SVCDec_Reconstruct_AU` and `SVCDec_ParseAU` make use of various sub-functions which have not been instrumented. Therefore the exclusive time for these both functions has been spend mostly in the sub-function calls, which is hidden in the current profile, due to the pruning of the function tree. In case these data are of relevance the number of instrumented functions has to be increased, by the sacrifice of simulation speed and accuracy. Especially for an in-depth function profile a different approach using hardware data collection probes is preferable.

The same function profile data is shown in Figure 10 for each processing element separately. To provide a fast overview about the most time consuming functions, all functions have been order by their exclusive time value. Such a separated view may be preferable in case the total number of functions increases in the function profile and the overall view becomes too confusing.

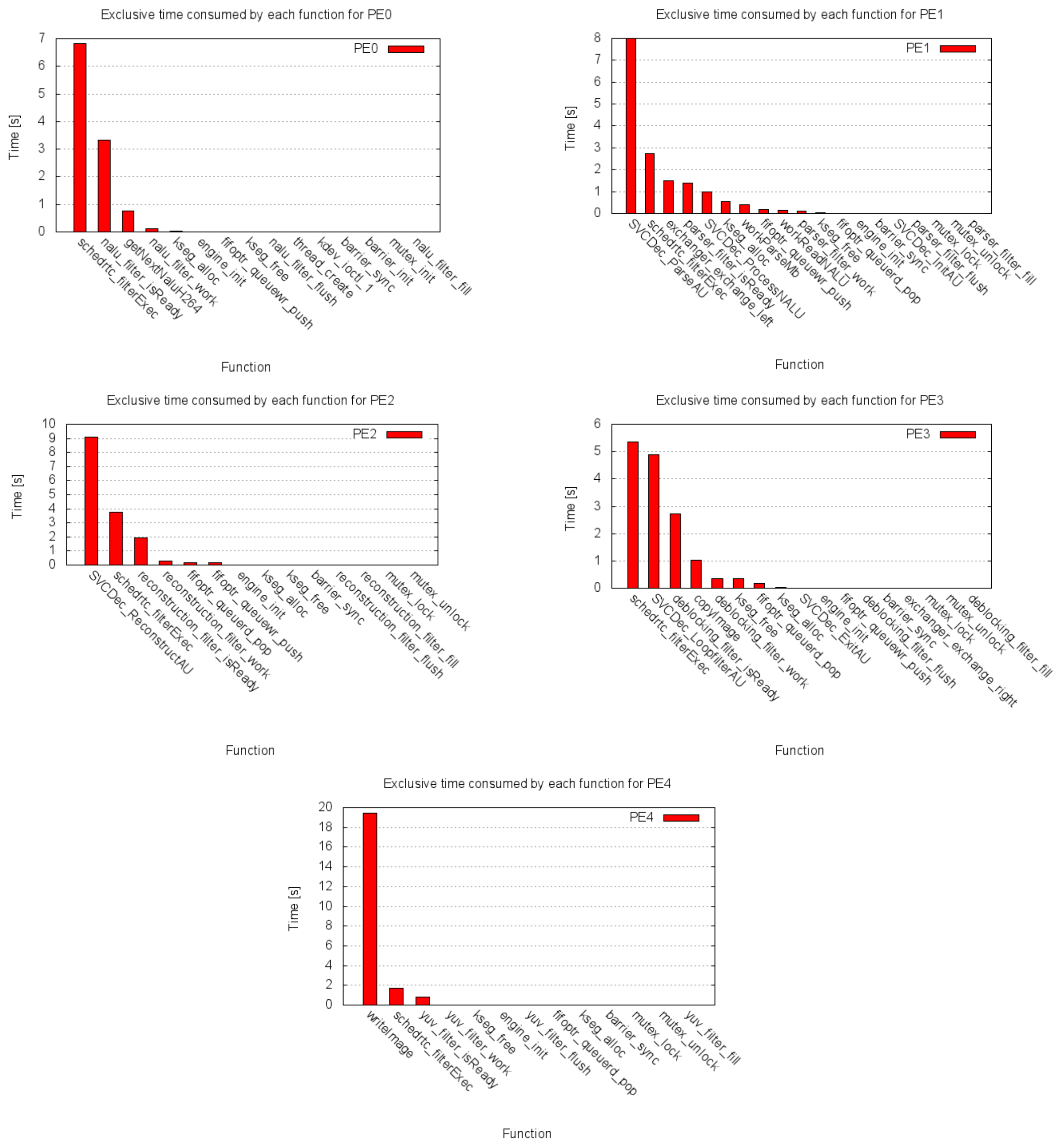


Figure 10 (a) - (d): Exclusive time consumed by each function for PE0 (a) to PE4 (d)

IV.2. Mean function profile

A different approach to analyse the relevance of a certain function for possible optimizations, is to determine the mean time each function has spend exclusive over the set of all processing elements as depicted in Figure 11.

Due to the fact that the function `schedrtc_filterExec`, a function part of the StreamScheduler component of the PPP repository, is performed on each processing element, this function is the most time consuming function over all processing element. The next four functions are performed on a single processing element only and represent the main data processing functions for each of these processing elements.

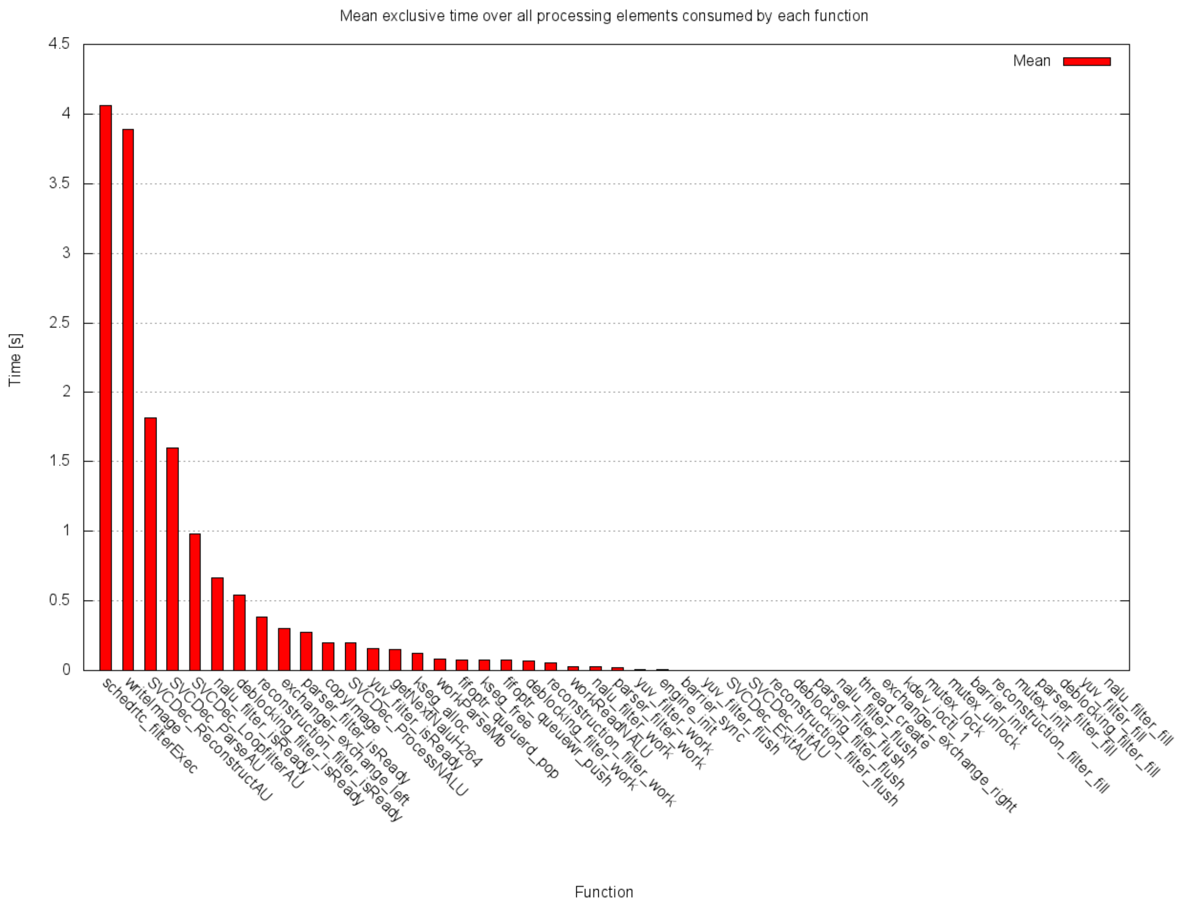


Figure 11: Mean exclusive time over all processing elements consumed by each function

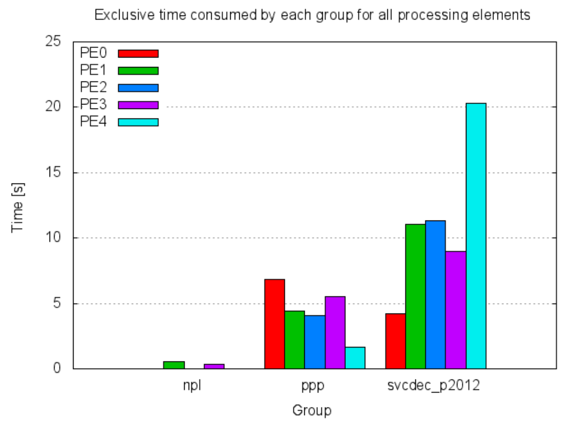
IV.3. Group profile

A special feature of the P2012 TraceLib is that the trace functions can be grouped into disjoint function groups. In the current application three different groups have been defined to separate between functions belonging to the underlying NPL runtime, the PPP component repository and the SVC decoder routines.

Such a group profile can be useful to analyse the workload distribution and the overall overhead imposed by the programming model. In Figure 12 (a) it can be seen, that the core

SVC decoder algorithm running on PE1, PE2 and PE3 has been split fairly equal. Contrary to this result, the workload of the sink component on PE4 is currently limiting the overall performance of the application.

In Figure 12 (b) it can be seen that the overhead imposed by the component framework cannot be neglected. Especially on PE0 the application has spend more time in the PPP components than in the data processing routines itself.



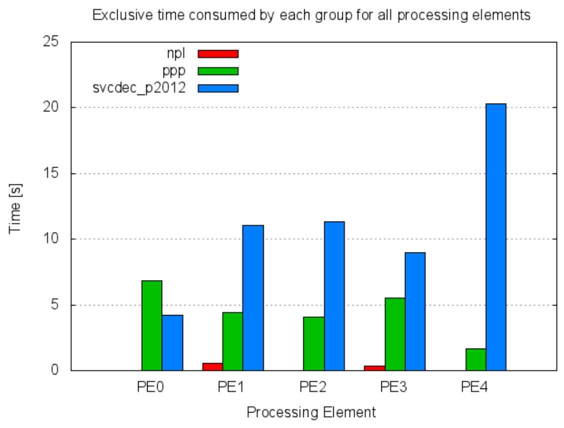


Figure 12 (a) (b): Exclusive time consumed by each group for all processing elements depicted (a) over the different groups or (b) over the processing elements

IV.4. Interval analysis

Also of importance is an analysis of the timing behaviour of the application. For this purpose NoCTrace can determine the execution time of the core data processing routines over a predefined interval from the P2012 TraceLib function trace data. In general for video decoding a constant playback frame rate is important. Therefore the analysis interval is adjusted adaptively to the number of the decode frames. The result can be seen in Figure 13, which shows the execution time of the core data processing routines for each frame.

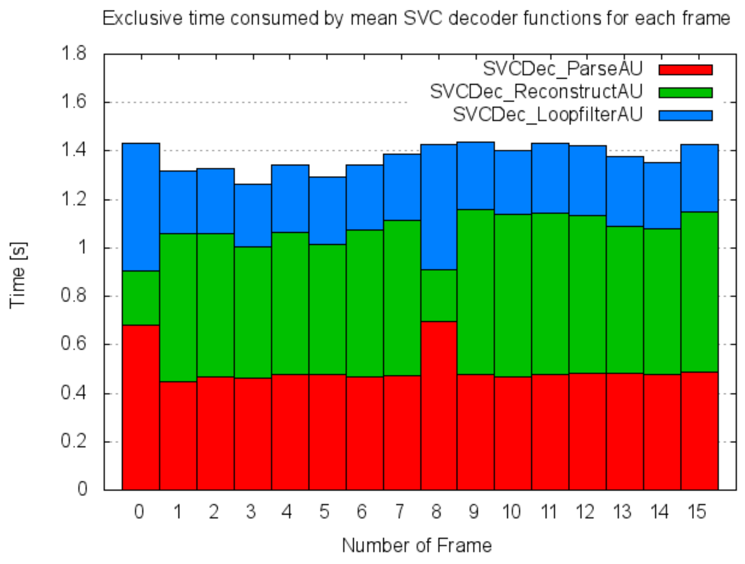


Figure 13: Exclusive time consumed by mean SVC decoder functions for each frame

The strongest deviation from the mean value for each function can be found for frame 0 and 8. In the current bit-stream these both frames are the intra only encoded frames (I-frames). All other frames use motion compensated prediction (P-frames). So in case of I-frames parsing and loop-filtering are much more processed than the reconstruction function.

IV.5. Memory profile

Unfortunately a tracing and profiling of all memory accesses is difficult to perform with a data collection module relying on software instrumentation. Instead NoCTrace can determine from the provided trace data the amount of memory allocated over the time and its maximum value. This is possible due to the fact that the memory allocation and free functions have been instrumented to report the size and the address of the allocated buffers.

In Figure 14 (a) and b) the memory profiles for the current example simulation run can be seen. The allocated memory can be distinguished between L1 cluster shared memory with a low latency and L3 global memory with a high latency. In summary the application allocates 1588 kB memory, from which the main part 1547 kB is stored in the L3 global memory. Of course the usage of the L1 cluster shared memory is preferable over the L3 global memory, due to the high latency of the L3 global memory. But the L1 cluster shared memory is limited to 256 kB which does not fit to the application needs, even at this small picture resolution. In order to optimize the application, it is crucial to determine the parts of the allocated memory which should be best placed into the L1 cluster shared memory. For such a purpose a memory access trace analysis is very useful.

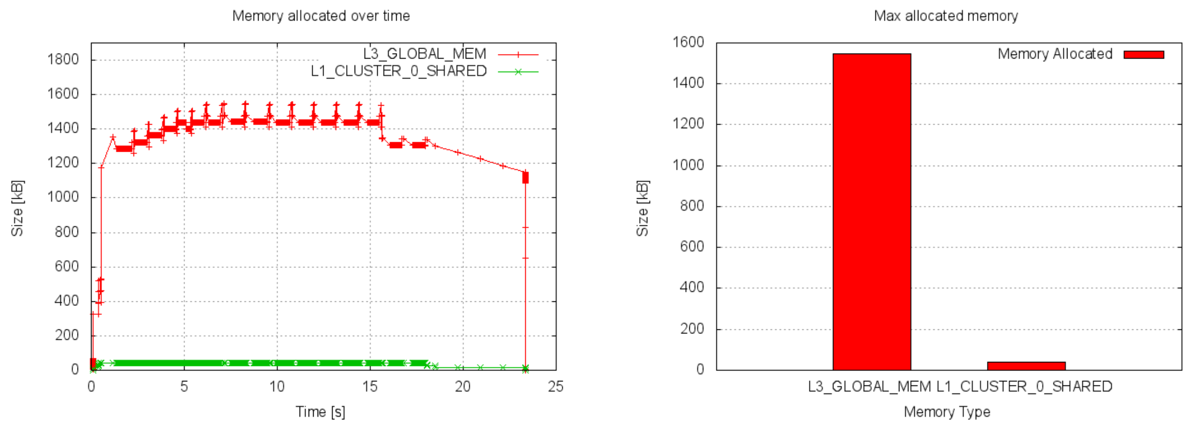


Figure 14 (a) (b): Memory allocated over time (a) and maximum allocated memory (b)

V. Analysis

In this section we analyze the execution of the SVC decoder application on platform P2012. The platform simulation model can be configured with different time accuracy parameters. Whilst in mode 0 the platform simulation model is untimed and the instruction set simulators are instruction accurate, in mode 1 the simulation model adds cycle approximate delays and delays on memories. For these analysis we configured the platform simulation model to run in mode 1 (`-time-accuracy=1`).

To encode the test bit-streams we choose the JSVM reference model software encoder [8] and the ITU test sequence BUS in different resolutions [9]. In order to reduce the real simulation time only 16 frames have been encoded in an I8P configuration. Therefore frame 0 and 8 are intra only encoded frames I-frames, whilst all the other frames are motion compensated frames P-frames.

All simulation runs have been performed on a DELL T5400 Precision workstation with 2 Intel Xeon X5450 CPU's with 3.00 GHz clock speed and 8GB main memory.

V.1. Bit-rate

In this test the influence of the bit-rate onto the SVC decoder application execution will be analyzed. In video decoding typically the bit-rate is linked with the quality of the decoded video sequence measurable in Peak Signal to Noise Ratio PSNR. The higher the bit-rate available to encode the test sequence, the higher the quality and the PSNR of the decoded video sequence.

For this test we used 6 configurations as shown in the table below; 3 test bit-streams have been encoded in QCIF resolution (176x144) with different quantization parameter QP. Additionally 3 test bit-streams have been encoded in CIF resolution (352x288) also with different QP values.

N	Name	Size and Rate	Frames	QP	Bit-rate	PSNR	Simul. Time	Real Time
1	BUS	176x144@15	16	42	49kBit/s	24.8	22.88s	5:00h
2	BUS	176x144@15	16	34	152kBit/s	30.4	23.38s	5:03h
3	BUS	176x144@15	16	27	350kBit/s	36.1	24.18s	5:09h
4	BUS	382x288@15	16	42	160kBit/s	25.7	89.82s	19:54h
5	BUS	382x288@15	16	34	474kBit/s	31.2	91.55s	20:05h
6	BUS	382x288@15	16	27	1143kBit/s	36.7	94.26s	20:37h

From quantization parameter 42 to 34 the bit-rate increases by factor 3 (QCIF 3.1, CIF 3.0) approximately. From quantization parameter 34 to 27 the bit-rate increases by factor 2.4 (QCIF 2.3, CIF 2.4) approximately. Accordingly the overall simulation time is increased by 2.1% (QCIF 2.2%, CIF 1.9%) and 1.0% respectively (QCIF 1.0%, 1.0%).

A closer view provides the mean exclusive time consumed by each group for the different quality parameters as depicted in Figure 15 (a) (b). The biggest influence can be found in the svcdec_p2012 group with an increase of the execution time of about 7.0 % (QCIF 7.6% CIF 6.4%) and 9.6% respectively (QCIF 10.0% CIF 9.2%).

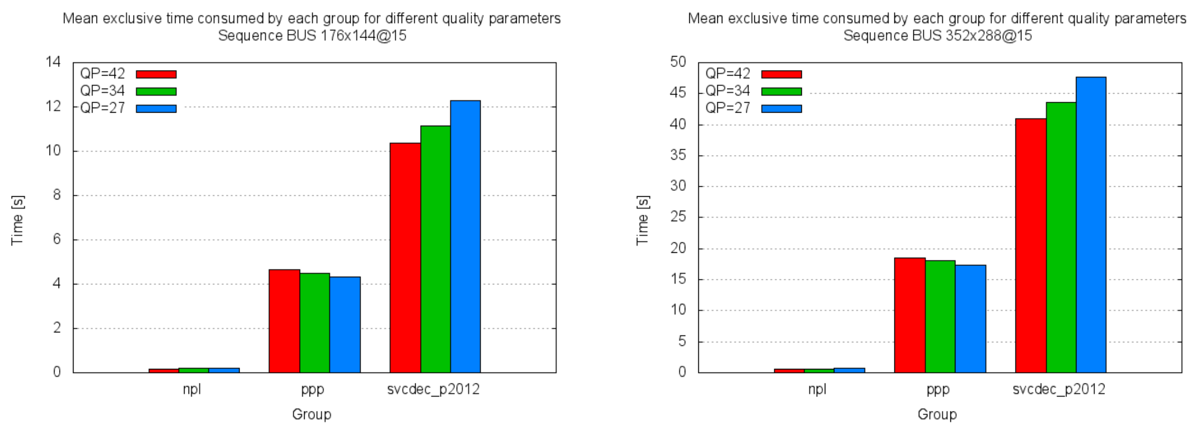


Figure 15 (a) (b): Mean exclusive time consumed by each group for different quality parameters (a) QCIF resolution (b) CIF resolution

In Figure 16 (a) (b) the execution time of the svcdec_p2012 group functions is shown for each processing element. On PE4 the CRC checksum of the decoded picture is computed, which is relatively independent of the picture quality. Contrastingly on PE0 and PE1 the bit-stream is processed and parsed. Therefore the execution time depends on the bit-rate.

At least on PE2 and PE3 the reconstruction and the loop-filtering are performed. Also here the execution time depends on the bit-rate, which is not obvious at a first glance. This dependency can be explained with the fact, that a bit-stream with a higher bit-rate contains more residual information which have to be reconstructed and also the loop-filtering becomes more complex in such a case.

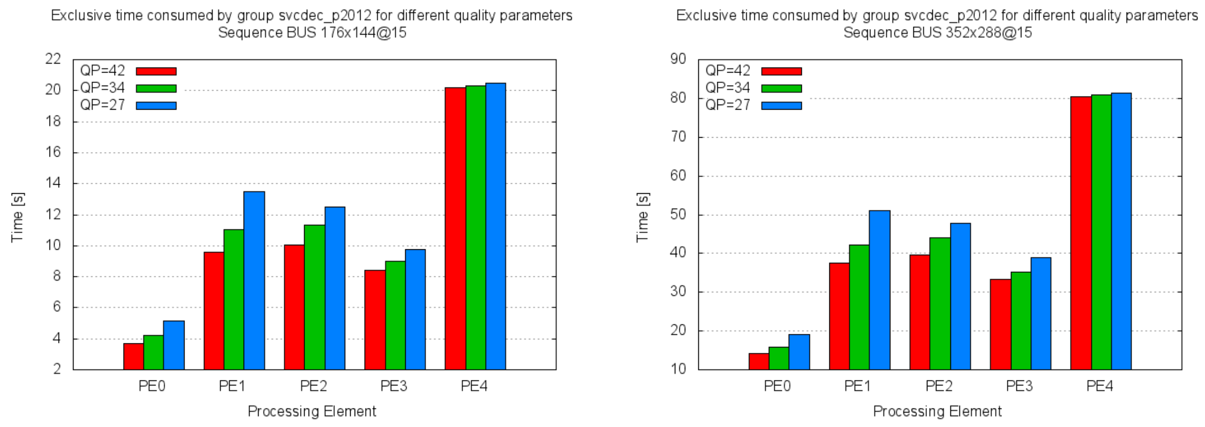


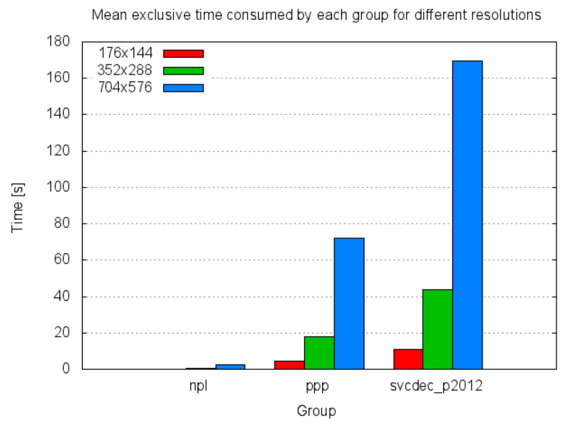
Figure 16 (a) (b): Exclusive time consumed by group svcdec_p2012 for different quality parameters (a) QCIF resolution (b) CIF resolution

V.2. Picture Resolution

In this test the influence of different picture resolutions onto the SVC decoder application execution will be analyzed. For this test we use 3 configurations as shown in the table below with 3 different resolutions QCIF (176x144), CIF (352x288) and 4CIF (704x576). All bit-streams have been encoded with the same quantization parameter. From one resolution to the next one the number of macroblock’s, which have to be processed by the decoder, is increased by 4 (400%).

N	Name	Size and Rate	Frames	QP	Bit-rate	PSNR	Simul. Time	Real Time
1	BUS	176x144@15	16	34	152kBit/s	30.4	23.38s	5:03h
2	BUS	352x288@15	16	34	474kBit/s	31.2	91.55s	20:05h
3	BUS	704x576@15	16	34	1089kBit/s	33.7	360.39s	78:54h

Locking at Figure 17 (a) it can be seen that the mean exclusive time consumed by each function group is increased similar to the picture resolution (NPL: 325%, 360%; PPP: 401% 400%; svcdec_p2012: 391%, 389%). In Figure 17 (b) the execution time of the svcdec_p2012 group functions is shown for each processing element. Also in this figure it can be seen that the execution times are increased similar to the picture resolution (PE0: 376%, 371%; PE1: 382%, 374%; PE2: 390%, 381%; PE3: 392%, 405%; PE4: 398% 399%). So in case of different picture resolution the exclusive timed consumed by the different function groups is scaled nearly proportional to the picture resolution.



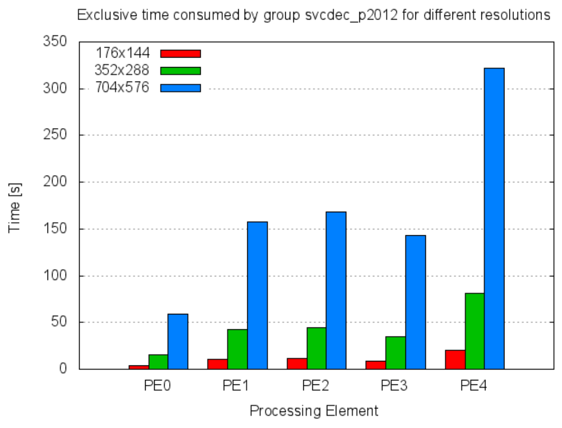


Figure 17 (a) (b): (a) Mean exclusive time consumed by each group for different resolutions; (b) Exclusive time consumed by group svcdec_p2012 for different resolutions

V.3. Scalability

In these tests the influence of the new algorithms introduced by the SVC standard on the SVC decoder application execution will be analyzed. For quality scalability and spatial scalability a separated test is performed. Temporal scalability is not analyzed due to the fact, that temporal scalability introduces no additional algorithm or complexity in comparison to AVC.

V.3.1. Quality Scalability

In this test the influence of the quality scalability inter-layer prediction on the SVC decoder application execution will be analyzed. For this test we use 6 configurations as shown in the table below; 3 different test bit-streams have been encoded at QCIF (176x144) resolution with 0, 1 and 2 quality layers. Additionally 3 test bit-streams have been encoded at CIF (352x288) resolution also with 0, 1 and 2 quality layers. The quantization parameter for each of the different quality layers has been chosen in a way, such that the target bit-rate is similar to the bit-rates in section V.1 (the bit-rate influence analysis). In this way the additional overhead imposed by the SVC inter-layer prediction can be quantified.

N	Name	Size and Rate	Frames	QL	Bit-rate	PSNR	Simul. Time	Real Time
1	BUS	176x144@15	16	0	49kBit/s	24.8	22.88s	5:00h
2	BUS	176x144@15	16	1	147kBit/s	28.8	25.70s	5:38h
3	BUS	176x144@15	16	2	352kBit/s	33.5	38.11s	8:22h
4	BUS	382x288@15	16	0	160kBit/s	25.7	89.82s	19:54h
5	BUS	382x288@15	16	1	464kBit/s	29.6	96.77s	21:12h
6	BUS	382x288@15	16	2	1182kBit/s	34.2	144.79s	32:00h

As explained in section V.1 the bit-rate increases by factor 3 between base layer and first enhancement layer and by factor 2.4 between first enhancement layer and second enhancement layer. The overall execution time is increased by 10.0% (QCIF 12.3%, CIF 7.7%) and 49.0% (QCIF 48.3%, CIF 49.6%) respectively. In comparison with the execution time increase from section V.1 of 2.1% and 1.0% the additional inter-layer prediction has a much higher influence on the SVC decoder application execution.

In Figure 18 (a) and (b) the mean exclusive time consumed by each function group for different quality layers can be seen. Of course the highest influence can be found again at the function group svcdec_p2012, but also for the group PPP a strong increase can be found between one quality enhancement layer and two quality enhancement layers.

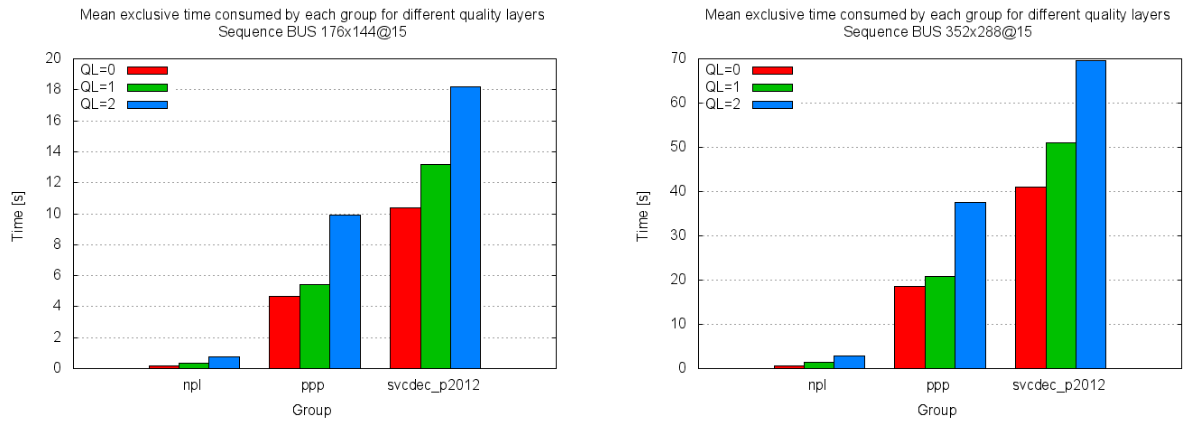
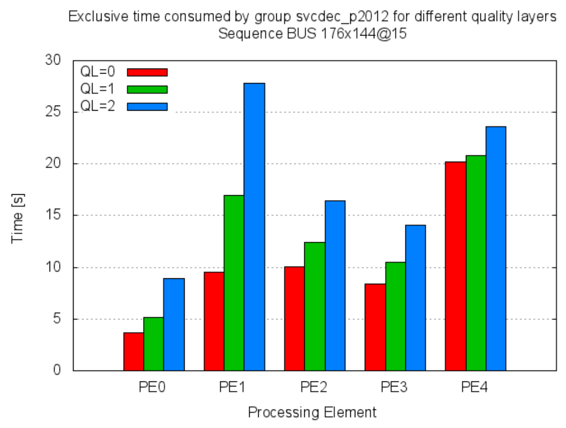


Figure 18 (a) (b) Mean exclusive time consumed by each group for different quality layers (a) for QCIF resolution and (b) for CIF resolution

To analyze this behavior Figure 19 (a) (b) and Figure 20 (a) (b) show the exclusive time consumed by the function groups svcdec_p2012 and PPP respectively.



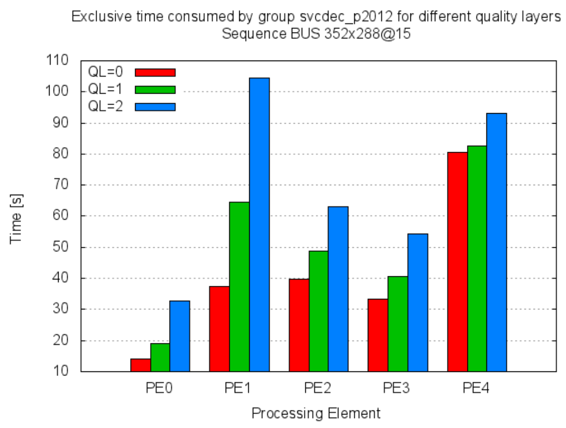
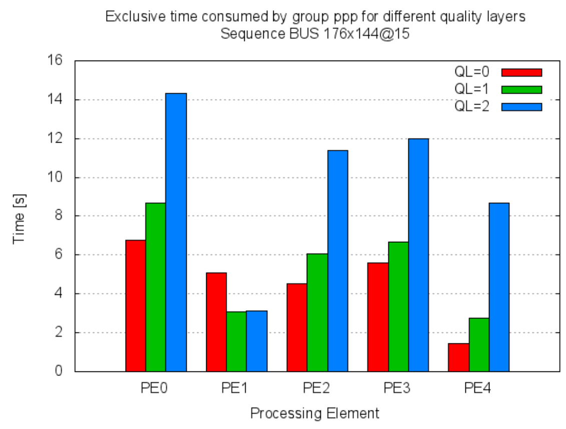


Figure 19: (a) (b) Exclusive time consumed by group svcdec_p2012 for different quality layers (a) for QCIF resolution and (b) for CIF resolution



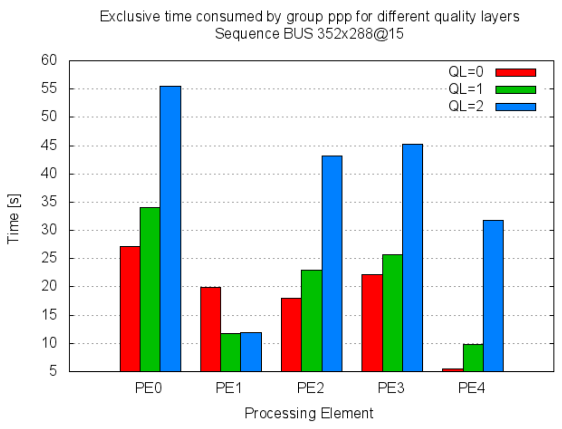


Figure 20 (a) (b): Exclusive time consumed by group PPP for different quality layers (a) for QCIF resolution and (b) for CIF resolution

As can be seen in Figure 19 (a) (b) the bit-stream parsing performed on PE1 increases most of all with each additional quality layer. At the end the parsing becomes the most executed function and even outperforms the CRC computation on PE4 for the configuration with two quality enhancement layers. In contrast to this effect the execution time of the PPP group decreases on PE1, whilst on all the other processing elements a strong increase can be found.

An explanation for this behavior can be found in the implementation of the Stream Scheduler component, which is part of the PPP component repository. The Stream Scheduler triggers the data processing component, to check whether the next portion of data can be processed. Due to the fact that the parsing component becomes the bottleneck, when the number of quality layers increases, the components on the other processing elements became more and more busy to check for the next available portion of data. Therefore the execution time of the PPP function group increases and even the execution time of the CRC processing on PE4 increases, albeit these functions should be independent from the number of the quality layers or the bit-rate of the input data.

V.3.2. Spatial Scalability

In this test the influence of the spatial scalability inter- layer prediction onto the SVC decoder application execution will be analyzed. For this test we use 3 configurations as shown in the table below; one test bit-stream contains only the spatial base layer with a QCIF (176x144) resolution. For comparison reasons a second bit-stream with CIF (352x288) resolution without using spatial scalability is included. In the third bit-stream spatial scalability inter-layer prediction from QCIF to CIF resolution has been enabled.

N	Name	Size and Rate	Frames	SL	Bit-rate	PSNR	Simul. Time	Real Time
1	BUS	176x144@15	16	0	152kBit/s	30.4	23.38s	5:03h
2	BUS	352x288@15	16	0	474kBit/s	31.2	91.55s	20:05h
3	BUS	352x288@15	16	1	604kBit/s	31.1	292.52s	63:56h

Figure 21 (a) shows the mean exclusive time consumed by each group for the different bit-streams; Figure 21 (b) and (c) show the exclusive time consumed by the svcdec_p2012 and PPP group functions respectively.

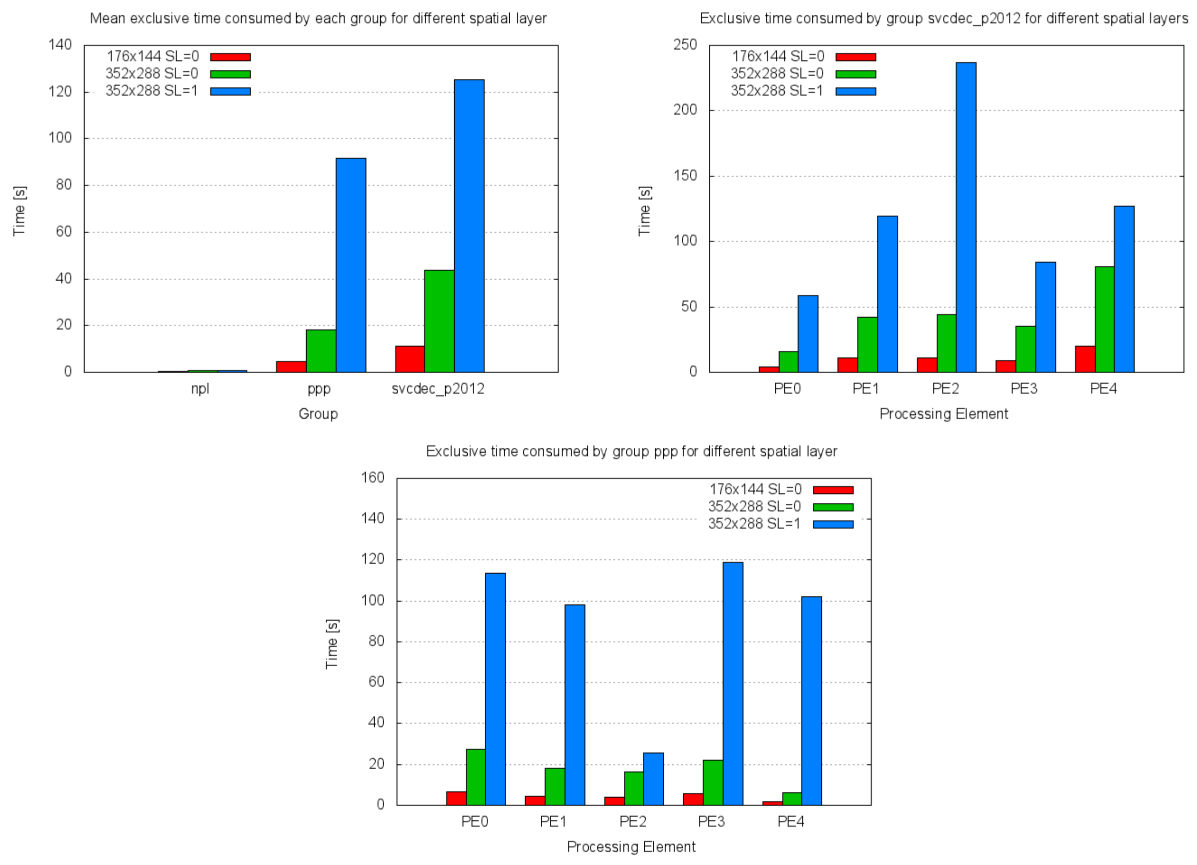


Figure 21 (a) (c): (a) Mean exclusive time consumed by each group for different spatial layer; (b) (c) Exclusive time consumed by group svcdec_p2012 (b) and PPP (c) for different spatial layers

In Figure 21 (a) it can be seen that the execution time of the PPP and svcdec_p2012 group functions for the spatial scalable bit-stream increases by 508% and 287% additionally in comparison to the single layer bit-stream with the same resolution. The highest increase can be found for PE2 (see Figure 21 (b)) where the reconstruction is performed. Part of the reconstruction process is the additional up-sampling introduced by SVC spatial scalability.

This up-sampling performed on PE2 can be expected to be the most important factor in the execution time increase. In contrast with this effect the PPP group execution time is increased very strong on all other processing elements PE0, PE1, PE3 and PE4 (see Figure 21 (c)). An explanation for this behavior has been given in section V.3.1 above.

V.4. Workload Distribution

A metric interesting for the analysis of the parallel execution of the SVC decoder application is the workload distribution. Due to the fact that this workload distribution might be very strong depend on the current input bit-stream data, we analyzed this metric for all 12 configurations tested in the previous sections. The analysis is focused on the data processing functions of the core SVC decoder. Therefore we exclude the input bit-stream processing on PE0 and the YUV frame output on PE4.

For this test we compute the mean execution time over all core data processing functions on PE1, PE2 and PE3 and the standard deviation from this mean value in order to get a metric for the workload distribution. The following table contains the results for each of the 12 configurations used in the previous sections.

Resolution	QP	QL	SL	Mean [ns]	SD	SD in %
176x144	42	0	0	9339502283	703570036	7.53
176x144	34	0	0	10437464945	1042342121	9.99
176x144	27	0	0	11913272861	1594024259	13.38
176x144	36	1	0	13291160614	2711899654	20.40
176x144	30	2	0	19435637095	6015430916	30.95
352x288	42	0	0	36826793173	2607188455	7.08
352x288	34	0	0	40472528955	3825139374	9.45
352x288	27	0	0	45922213157	5173023044	11.26
352x288	36	1	0	51331171159	9960871013	19.41
352x288	30	2	0	73986683559	21947323482	29.66
704x576	34	0	0	1.56084E+11	10373863226	6.65
352x288	34	0	1	1.47088E+11	65251144532	44.36

In Figure 22 (a) (b) and (c) these results are shown. In summary it can be noted, that in the AVC case with only one layer the workload distribution is good balanced between parsing, reconstruction and loop-filtering, independently of the bit-rate and the picture resolution. Adding SVC quality or spatial enhancement layer results in a more unbalanced workload distribution due to the fact, that either the parsing in case of quality scalability or the reconstruction in case of spatial scalability needs much more computational resources.

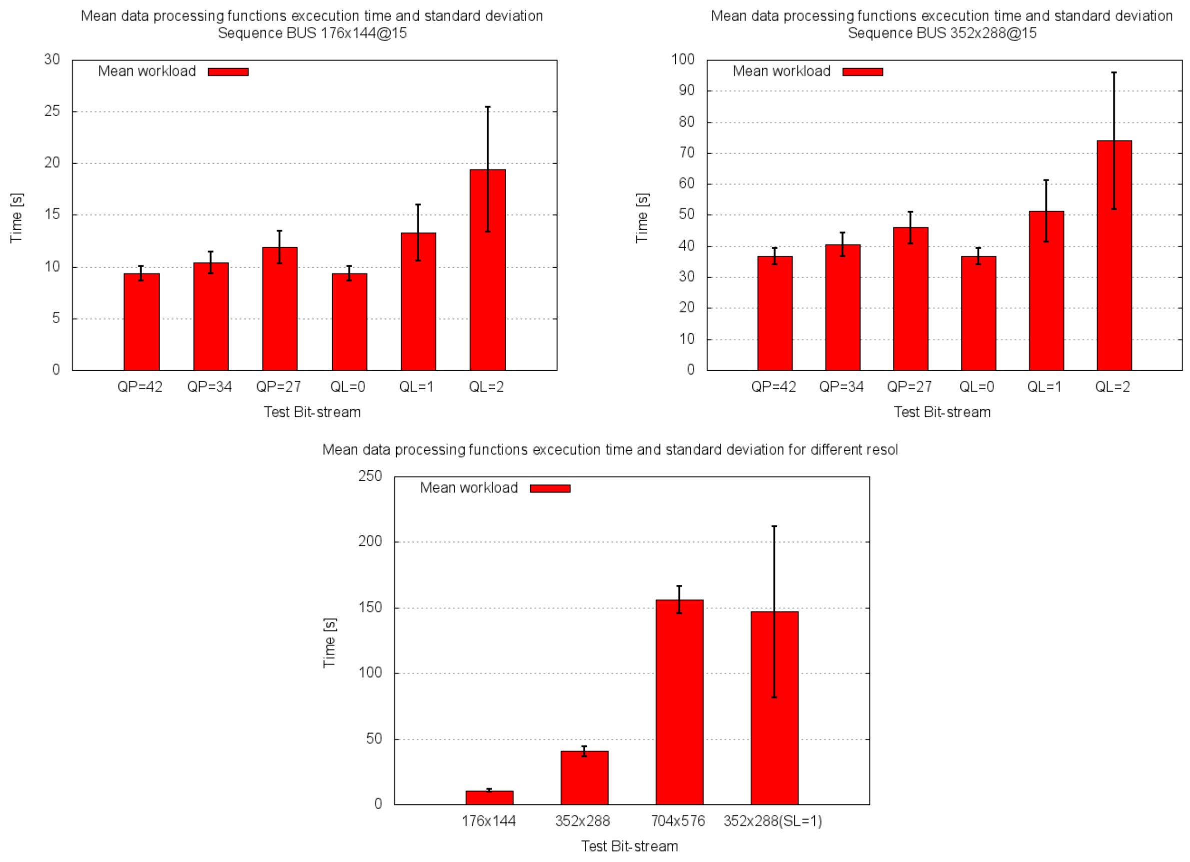


Figure 22 (a) - (c): Mean data processing functions execution time and standard deviation (a) for QCIF resolution (b) for CIF resolution and (c) for different resolutions

V.5. Memory

In this analysis we provide information on the amount of allocated memory during the execution of the SVC decoder application. In the current version of the SVC decoder nearly all memory buffers are allocated in the L3 global memory region. Only a small amount of memory, which does not vary over the different configurations, is allocated in the L1 cluster shared memory. Therefore we show only the data for the L3 global memory region, for all of the 12 configurations described in the sections above.

In Figure 23 (a) and (b) it can be seen that the maximum amount of allocated memory changes only by a small amount for different bit-rates. The amount of input bit-stream data is relatively small compared with the data necessary to store the reconstructed pictures. Compared with this, increasing the number of quality layers in the bit-stream increases also the amount of allocated memory, due to the fact, that some intermediated computational results have to be stored in addition to enable the quality inter-layer prediction.

Increasing the picture resolution results in a similar increase of the allocated memory as can be seen in Figure 23 (c). Also in this case it can be noted, that for spatial scalability an additional amount of memory has to be allocated in comparison with the single layer case.

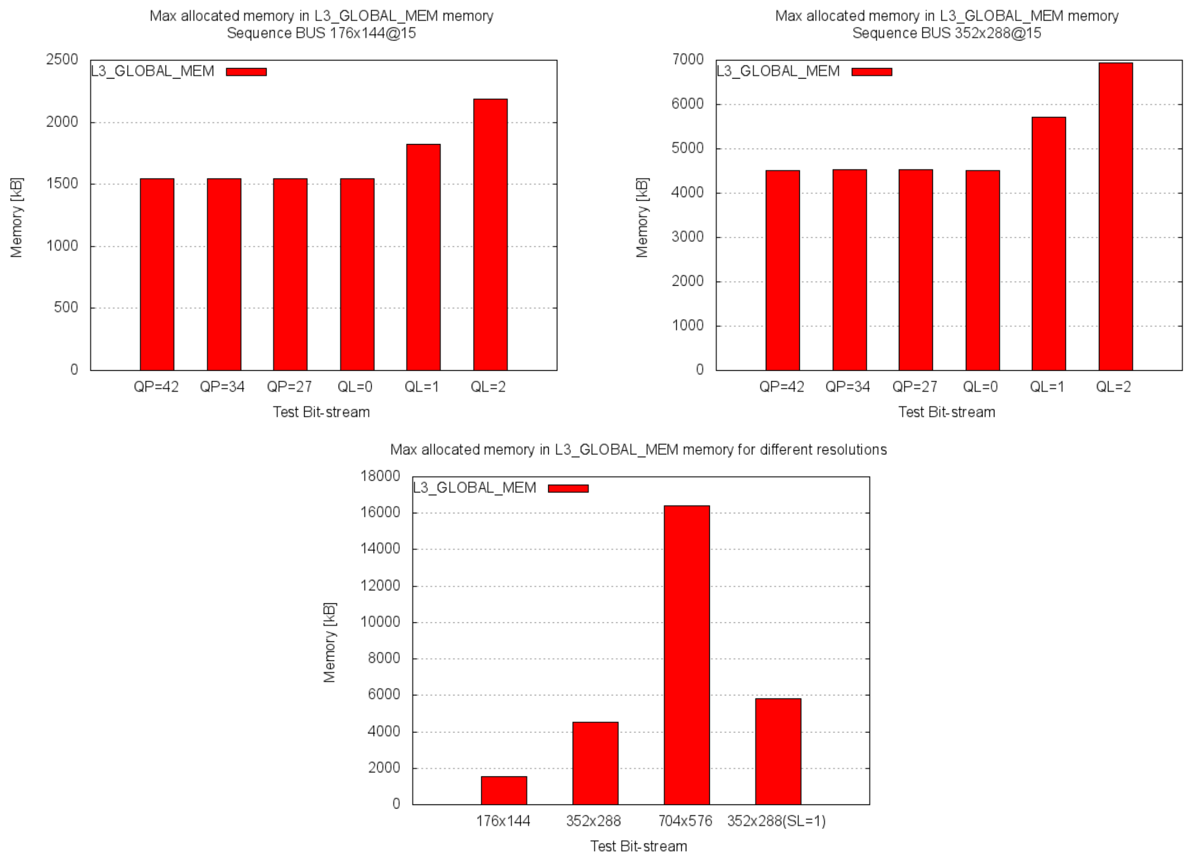


Figure 23 (a) - (c): Max allocated memory in L3_GLOBAL_MEM memory (a) for QCIF resolution (b) for CIF resolution and (c) for different resolutions

V.6. Summary

In summary the following observations can be made from the analysis presented in sections V.1 to V.5:

- Both quality scalability and spatial scalability impose a noticeable effect on the execution time and the amount of allocated memory of the SVC decoder (see section V.3 and V.5). Therefore a resource manager might choose the video representation appropriate for a given resource constraint, trying to optimize the visual quality of the decoded bit-stream. Moreover such a modification might be realized at runtime, making the SVC decoder application highly adaptive to the current resources available.
- In the current implementation the workload distribution becomes more and more imbalanced in case additional SVC enhancement layers have to be decoded (see section V.4). Additional optimizations are necessary to deal with this effect.
- The current implementation of the scheduling of the components on the processing elements, leads to an increased overhead in particular on the components which tend to have less work to do (see section V.3). The implementation needs to be modified especially in case the power consumption of the platform should be minimized.
- In general the current implementation is not capable to decode even a small resolution like QCIF in realtime. To optimize the whole application with respect to realtime issues the different memory latencies have to be taken into account and probably the degree of parallelism has to be improved.

VI. Conclusion

In this deliverable we presented an efficiency analysis of a parallel SVC decoder running on platform P2012 using a CBSE based parallel programming model. The analysis has been performed on various input bit-stream data to provide a robust view on the application dependencies with respect to different workload. Different metrics like execution time, workload distribution and allocate memory have been presented in the previous sections.

This deliverable is the first report on the efficiency analysis as part of the research performed in 2PARMA task 3.2. A second and final report is to be expected at M30. This enables us to optimize the application and the programming model accordingly to the observations presented in this deliverable. The results of this deliverable will also influence the development of the profiling tool in T3.1, as has been shown, that especially a memory access analysis will become important to exploit the full potential of P2012.

VII. References

[1] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Rec. H.264 and ISO/IEC 14496-10:2010 (Advanced Video Coding), ITU-T and ISO/IEC JTC 1/SC 29, Mar. 2010.
[2] Platform 2012: A Many-core programmable accelerator for Ultra-Efficient Embedded Computing in Nanometer Technology. STMicroelectronics and CEA, Nov. 2010
[3] MIND Homepage: http://mind.ow2.org/index.html
[4] Initial implementation of the profiling tool for parallel computing platforms. 2PARMA deliverable D3.1.1, June 2010.
[5] Heiko Schwarz, Detlev Marpe, Thomas Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9), pp. 1103-1120, September 2007.
[6] Gnuplot Homepage: http://www.gnuplot.info/
[7] P2012 Tools Documentation - Programmer Manual: version 1.1. STMicroelectronics, 2011.
[8] SVC reference software: http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm
[9] SVC test sequences: ftp://ftp.tnt.uni-hannover.de/pub/svc/testsequences/