Profiling tool for the Transaction Data of Scalable Video Decoding

Abelardo Baez Quevedo, Gustavo M. Callicó, Sebastian López
Research Institute for Applied Microelectronics (IUMA), University of Las Palmas de Gran Canaria (ULPGC)
Campus Universitario de Tafira, Las Palmas (Spain)
{ abaez, gustavo, seblopez }@iuma.ulpgc.es

***Abstract*— Is highly desirable to known in advance the transaction data in the design of an electronic embedded system. Especially for data-intensive applications, such as complex video system, when the options available in the video decoder change and/or the features of the input video sequences are different. This paper exposes the development of a profiling script intended to help in the transaction data of decoding video sequences using the H.264 Scalable Video Coding (SVC) standard. The tool incorporates some Python scripts that allows the system designer analyze the transaction data of the decoder, with several bit streams automating profiling and decoding tasks. Using the profiling scripts, the results show how the transaction data load changed based on the intrinsic characteristics of the video sequence and in the scalable options selected. Due to the huge number of functions that form part of the SVC video decoder, a set of modules that better describe the internal structure of the decoder has been defined. The assignation of functions to modules is open to the designer and can be changed at any time to accommodate changes in the system. This tool make the created profiling environment a helpful tool for the system designer to make better decisions about the transaction data load distribution, based on the modules defined inside the tool.**

Keywords — Scalable Video Coding, profiling, performance analysis, transaction data.

I. INTRODUCTION

The Scalable Video Coding (SVC) is an Annex of the H.264/AVC standard [1]-[2], which extends the original standard with new tools designed to efficiently support temporal, spatial and quality (SNR) scalability developed by the Joint Video Team (JVT). The JVT is a group of experts from ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Pictures Experts Group (MPEG) created to develop the H.264/AVC standard.

The Open SVC Decoder (OSD) [3] is an open source library created at the IETR/INSA of Rennes that implements a SVC decoder written in C language. It has been developed and tested over different platforms: x86 processors, PDAs and DSPs, making it interesting for embedded systems.

A profiling tool could be basically defined as an analysis environment that measures the performance of a piece of code. This performance is measured in several different ways, in this case we focus on transaction data load and the memory usage. Profiling tools give detailed information about where the bottlenecks are located in a code or which functions are using larger amounts of memory.

In this work, the Valgrind have been used [4]. Valgrind is a tool suite that let system developers measure, evaluate, and target performance-related issues in their code. The main problem encountered is that Valgrind are a set of different standalone tools that require the constant attention of the developer, and they do not constitute an integrated automated environment where the designers can setup several profiling sessions.

The PySVCVal (PSV) is a program written and developed in Python intended to automate profiling sessions of Open SVC Decoder using the profiling tools provided by Valgrind. The PSV tool includes a friendly way to add the bit streams and manage the profiling sessions.

This paper explains the developing of PSV and its features designed to automate the profiling process. Section II describes the main features of PySVCVal. Section III analyses some profiling results obtained with PSV.

II. PYSVCVAL, A PROFILING TOOL FOR THE TRANSACTION DATA OF OPEN SVC DECODER

PySVCVal (PSV) is a script tool writted in Python using the Valgrind profiling tools under Linux, which helps in the profiling sessions of OSD.

The designer can add several bit-streams to automate the profiling sessions. The collected data results are stored in an Excel spreadsheet.

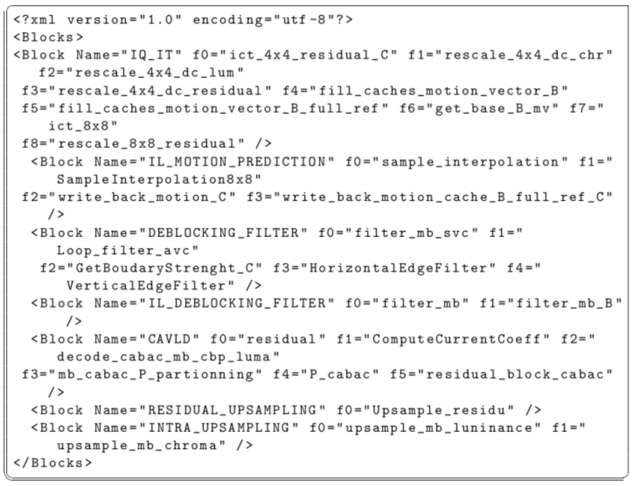
Besides Valgrind, PSV also makes use of several additional tools to automate the profiling sessions, as the "Bit Stream Extractor" from JSVM tools, and xlxsWriter Python library to generate the Excel output.

In summary, PySVCVal was developed under the following constrains and conditions:

- Python 2.7.3
- xlsxWriter Library
- Valgrind Profiling Tools
- BitStreamExtractorStatic.exe (JSVM Tool).

PSV can be only configurable via two external XML files, "bitstreams.xml" and "functions.xml".

The file “functions.xml” has one section named Blocks, and one element per each block named Block, with attributes Name, that contains the name of the block, and attributes fn that contains the n functions associated to the Block. An example of the file “functions.xml” can be seen in Fig. 5. A file called “functions.xml” has one section named Blocks, and one element per each block named Block, with attributes Name, that contains the name of the block, and attributes fn that contains the n functions associated to the Block. An example of the file “functions.xml” can be seen in Fig. 4.



```
<?xml version="1.0" encoding="utf-8"?>
<Blocks>
<Block Name="IQ_IT" f0="ict_4x4_residual_C" f1="rescale_4x4_dc_chr"
   f2="rescale_4x4_dc_lum"
 f3="rescale_4x4_dc_residual" f4="fill_caches_motion_vector_B"
 f5="fill_caches_motion_vector_B_full_ref" f6="get_base_B_mv" f7="
    ict_8x8"
 f8="rescale_8x8_residual" />
  <Block Name="IL_MOTION_PREDICTION" f0="sample_interpolation" f1="
     SampleInterpolation8x8"
 f2="write_back_motion_C" f3="write_back_motion_cache_B_full_ref_C"
    />
  <Block Name="DEBLOCKING_FILTER" f0="filter_mb_svc" f1="
     Loop_filter_avc"
   f2="GetBoudaryStrenght_C" f3="HorizontalEdgeFilter" f4="
      VerticalEdgeFilter" />
  <Block Name="IL_DEBLOCKING_FILTER" f0="filter_mb" f1="filter_mb_B"
      />
  <Block Name="CAVLD" f0="residual" f1="ComputeCurrentCoeff" f2="
     decode_cabac_mb_cbp_luma"
 f3="mb_cabac_P_partionning" f4="P_cabac" f5="residual_block_cabac"
    />
  <Block Name="RESIDUAL_UPSAMPLING" f0="Upsample_residu" />
  <Block Name="INTRA_UPSAMPLING" f0="upsample_mb_luninance" f1="
     upsample_mb_chroma" />
</Blocks>
```

Figure 4. Function.xml example

Normally, user needs to run some programs in the background, to check the scalability levels of a bit stream , select the scalability levels desired, and then call Valgrind with OSD to decode that bit stream. After the profiling session has stopped, sometimes is necessary to run another tool to analyze the results obtained, normally writing the results in a file.A diagram of the full process can be seen in Fig. 5.

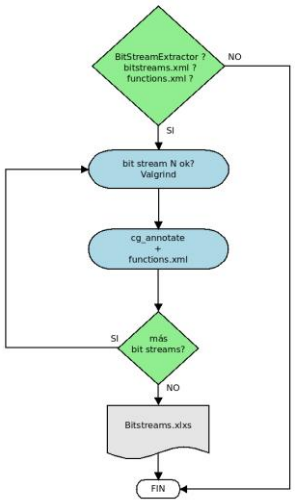


Figure 5. Process diagram in PySVCVal

PSV uses Cachegrind, the cache and branch-prediction profiler of Valgrind to analyze the bit streams selected in bitstreams.xml, once Cachegrind finish, it is recommended use other tool in Valgrind to get a detailed presentation of the profiling information. Cg_annotate can get only the information desired and show that information, or generate a file with selected information.

III. PREPARE YOUR PAPER BEFORE STYLING

To test PSV, several profiles using some of the common videos used in the video research community were performed. In Fig.6 and Fig 7 are shown some of the tests performed with PSV to check the transaction data load of each block defined in OSD as a function of the selected scalabilities.

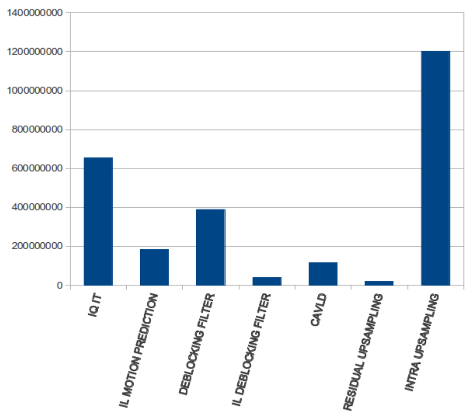


Figure 6. Transaction Data Load in OSD Blocks

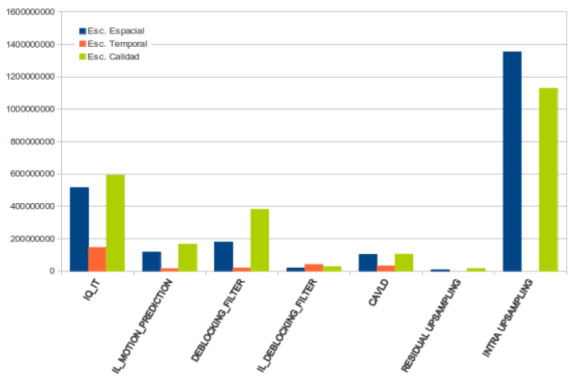


Figure 7. Total Transaction Data Load

REFERENCES

[1] vWiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A., “Overview ofthe H.264/AVC video coding standard,” Circuits and Systems for Video Technology, IEEE Transactions on , vol.13, no.7, pp.560-576, (2003).

[2] Mrak, M., Sprljan, N., Izquierdo, E., “An overview of basic techniquesbehind scalable video coding,” Electronics in Marine, 2004. Proceedings Elmar 2004. pp. 597- 602, (2004).

[3] M. Blestel and M. Raulet, “Open SVC decoder: a flexible SVC library,”Proceedings of the international conference on Multimedia (MM '10). ACM, New York, NY, USA, 1463-1466, (2010).

[4] Valgrind Home Page, http://www.valgrind.org , (2013).

[5] Maiti, S.N., Gupta, A., Piccinelli, E.M., and Saha, K., “Real-time SVC Decoder in Embedded System,” In Proceedings of SIGMAP. (2009).

[6] Joint Video Team, “Conformance testing,” http://wftp3.itu.int/av-arch/jvt-site/bitstream_exchange/SVC/, (2008).

[7] Heiko Hübert and Benno Stabernack, “Profiling-Based Hardware/Software Co-Exploration for the Design of Video Coding Architectures”, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 19, NO. 11, NOVEMBER 2009.