National Conference on Electronics and Communications (REV2013-KC01)

System-on-Chip Testbed for Validating the Hardware Design of H.264/AVC Encoder

VNU, University of Engineering and Technology
Hanoi, VietNam
Email:

***Abstract*— This paper presents an implementation of a LEON3-based System-on-Chip (SoC) testbed, which is aimed at experimentally evaluating and validating the H.264/AVC video encoding Integrated Circuit (IC) developed by SIS Laboratory at VNU University of Engineering and Technology. In addition, the paper also presents a methodology for verifying the design of H264/AVC video encoder in the Hardware/Software co-emulating fashion. The design is implemented on the DE2 development board from Altera Corporation. The testbed is able help us to evaluate effectively many aspects of the developed H.264/AVC video encoder.**

***Keywords*— SoC testbed, Harware/Software co-verification, LEON3 processor, H.264 encoder**

I. INTRODUCTION

Because of the high mask set cost for fabricating ASIC (Application-specific Integrated Circuit), it is necessary to verify and evaluate carefully the design at all design phases in order to ensure the fabricated chip is without bug. Prototyping an ASIC design, which has large integration level and high complexity, using FPGA (Field Programmable Gate Array) is indispensable in the design process.

ASIC design is more and more complex. The major challenge the designer must be confronted to design such an IC (Integrated Circuit) is verification. Generally, verification consumes at least 50%~80% of the design effort [1]. Verifying the design correctness is considered to be the key barrier against designing more complex VLSIs (Very Large Scale Integration), as well as exploiting leading-edge process technologies. There is not any single design tool that can solve the problem. Instead, a complex chain of tools and techniques, including classical simulation, directed and random verification, and formal techniques, etc., is required to reduce the number of design errors to an acceptable minimum. In this paper, we developed a LEON3-based SoC testbed and the platform-based verification method, which is aimed at experimentally evaluating and validating the H.264/AVC video encoding IC designed by SIS Laboratory at University of Engineering and Technology, Vietnam National University, Hanoi. This testbed can help us to evaluate effectively many aspects of the designed H.264/AVC video encoder.

The goal of verification is to ensure that the design meets the functional requirements as defined in the functional specification. In the top-down method for ASIC design and verification, the designer first develops a system-level model of the design from the functional specification. The system-level model is normally the high-level behavioral abstraction that is written in a high-level programming language such as C/C++. Alternatively, this model may also be created using the hardware description language (HDL) such as Verilog or VHDL. The behavioral model should be simulated in order to verify that it meets the required functionality completely and correctly. The behavioral model is then used as a reference to refine and create a synthesizable RTL (Register Transfer Level) model.

Before being synthesized to a structural model (or gate-level model), the RTL model is verified again to ensure that it exactly provides the required functionality and performance. The functional verification of the design at this step must be as complete and thorough as possible. This requires that the test vectors employed during simulation should provide the necessary coverage to ensure the design will meet specifications without bug. Unfortunately, the verification by simulation is difficult to test all cases. While the size of design increases, it might be unfeasible to run the full test-bench on a RTL model because of the huge simulating time. In this case, it is necessary to speed up the simulation using emulator, rapid prototype system, or hardware accelerators or to partition the design into several functional blocks. The modules are extracted from an abstract model of the design, and then individual modules can be verified independently with their associated test-bench. Afterwards, system-level emulation can run in a mixed mode where most modules are simulated with high-level abstract models, while one or some modules are substituted by hardware accelerator(s).

The rest of the paper is organized as follows. The hardware architecture of H.264/AVC video encoder is firstly introduced in Section II. Next, the design and implementation of the SoC testbed are presented in Section III. Section IV presents the methodology for verifying a hardware design by using the proposed SoC testbed. The details of validating H.264/AVC video encoder and experimental results are presented and discussed in Section IV. In Section V, some conclusions are drawn.

II. INTRODUCTION TO THE H.264/AVC VIDEO ENCODER

A. Basic concepts of H.264/AVC video encoding

The H.264/AVC video encoding standard is known as an efficient video encoding standard providing high quality at a very low bitrate in comparison with the previous standards such as MPEG-2 and MPEG-4.

The general architecture of the H.264/AVC encoder, composed of different functional blocks, is depicted in Fig. 1.

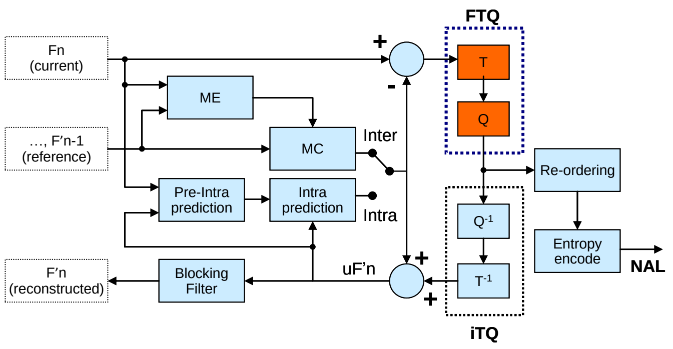


Fig. 1 Functional diagram of the H.264/AVC encoder.

In order to achieve high compression ratio, the H.264/AVC standard has adopted several advances in coding technology to remove spatial and temporal redundancies. These prominent techniques are as follows:

- A new way to handle the quantized transform coefficients has been proposed for trading-off between compression performance and video quality to meet the requirements of applications. Besides that, an efficient method called Context-Adaptive Variable Length Coding (CAVLC) is also used to encode residual data. In this coding technique, VLC tables are switched according to already transmitted syntax elements. Since these VLC tables are specifically designed to match the corresponding image statistic, the entropy coding performance is impressively improved in comparison to schemes using only a single VLC table [2];
- The H.264/AVC adopts variable block size prediction to provide more flexibility. The intra prediction can be applied either on 4×4 blocks individually or on entire 16×16 macroblocks (MBs). Nine different prediction modes exist for a 4×4 block while four modes are defined for a 16×16 block. After comparison among the cost functions of all possible modes, the best mode having the lowest cost is selected. The inter-prediction is based on a tree-structure where the motion vector and prediction can adopt various block sizes and partitions ranging from 16×16 MBs to 4×4-blocks. To identify these prediction modes, motion vectors, and partitions, the H.264/AVC specifies a very complex algorithm to derive them from their neighbors;
- The forward transform/inverse transform also operate on blocks of 4×4 pixels to match the smallest block size. The transform is still Discrete Cosine Transform (DCT) but with some fundamental differences compared to those in previous standards [3]. In [4], the transform unit is composed of both DCT and Walsh Hadamard transforms for all prediction processes;
- The in-loop deblocking filter in the H.264/AVC depends on the so-called Boundary Strength (BS) parameters to determine whether the current block edge should be filtered. The derivation of the BS is highly adaptive because it relies on the modes and coding conditions of the adjacent blocks.

B. *VENGME Hardware Architecture*

The "Video Encoder for the Next Generation Multimedia Equipment (VENGME)" project, supported by the Vietnam National University, Hanoi, aims at designing and implementing an H.264/AVC encoder targeting mobile platforms. The current design is optimized for CIF video; however, the architecture can be extended for larger resolutions by enlarging the reference memory and the search window.

One of the factors which affect both computational path and the power consumption is the workload of the system and the data dependencies among the pipeline stages. In H.264/AVC encoder, the most time consuming part is inter prediction including Integer Motion Estimation (IME), Fractional Motion Estimation (FME), and Motion Compensation (MC). The second time consuming module in the encoder is the entropy encoder (EC). Therefore, the architecture should be carefully selected to improve the coding throughput and the overall performance. Our proposed designs for Inter-Prediction, Entropy Encoder, and Forward Transformation and Quantization (FTQ) have been presented in [4]-[7].

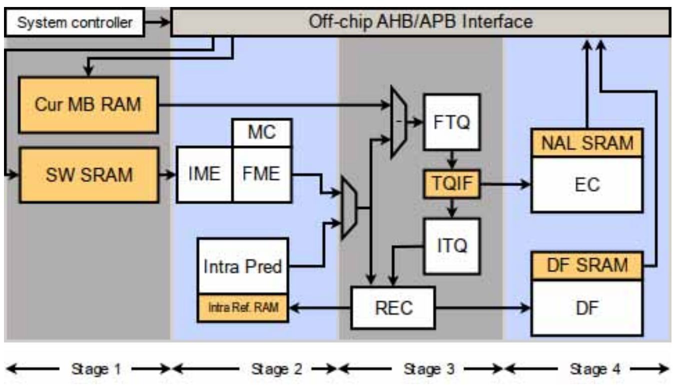


Fig. 2 VENGME H.264/AVC Encoder Architecture.

The complete architecture of VENGME H.264/AVC encoder uses the classical 4-stage pipeline scheme with some modifications, as illustrated in Fig. 2. The first stage is used to load the data needed for the prediction. The second stage includes intra- and inter-predictions. IME and FME are merged into the same stage because FME and MC can reuse the information from IME and the data from the search window SRAM. Inter-prediction and intra-prediction in the same stage can be executed in parallel or separately, thanks to the system controller decision. In the separate mode of execution, to save the power consumption, the intra- or inter-prediction can be switch off while the other in active state. In the mixed mode of execution, the intra prediction and inter prediction can be done in parallel, the intra prediction will finish first, and its results are stored in TQIF (TQ Interface) memory. After that, the intra module can be switched off to save power. Inter prediction and motion compensation continue to find the best predicted pixels. After having inter prediction results, TQIF memory can be invalidated to store new transformed results for inter module. The third stage and the final stage are the same as the classical 4-pipeline architecture.

III. Design and Implementation of a SoC Testbed

Top-level architecture of the SoC testbed is shown in Fig. 3. Altera DE2 development board is used as a prototype for this SoC testbed. The SoC testbed mainly consists of the blocks as follows. LEON3 processor [8] functions as the central processing unit (CPU) that takes charge of managing and scheduling all activities of the system. It receives the interrupt, stores data from input devices, processes data, and sets up operations for data transfer between memory and other devices. A real-time operating system (RTOS) (e.g. Linux) running on the processor may be responsible for performing all the above tasks. SDcard/SDRAM/FLASH/SRAM controllers provide the interface to external SDcard/SDRAM/FLASH/SRAM memories, respectively. SD card stores benchmark video sequences. SDRAM (Synchronous Dynamic Random Access Memory) stores input data (e.g. the encoding video frame) and intermediate data (e.g. reference frames and encoded frames). SRAM (Static RAM) buffers the temporary data during operating of the system. Flash memory stores the initialization and configuration information of the system, as well as holds the application program for CPU. The components communicate with each others by an AMBA bus, which is an on-chip bus architecture defined by ARM. The AMBA bus consists of three parts: AMBA High-performance bus (AHB) aims at connecting to high-bandwidth devices; AMBA peripheral bus (APB) targets at connecting to the devices that require a lower bandwidth; and a bridge joins AHB bus and APB bus together (AHB/APB Bridge). Some assistant functional modules such as interrupt controller (IRQ controller), UART, Timer, PS/2 and GPIO interface are connected to APB bus, whereas SDRAM/FLASH/SRAM controllers are connected to AHB bus.

User-defined IP (Intellectual Property) cores can be connected to AHB or APB bus for verifying. For example, considering the H.264/AVC encoder that organized into a number of modules (i.e. User-defined IP core in Fig. 3). Each IP core is specific to a particular function such as ME, DCT, etc. To increase flexibility, a wrapper is used that make the interface of IP cores compatible with the AHB or APB bus so that it can communication with other integrated components in the system. The wrapper integrates a PLL (Phase Lock Loop) and a DMA (Direct Memory Access) unit, which are reprogrammable at run-time by CPU. Here, PLL takes charge of synthesizing the clock signal that required by IP core, whereas DMA unit is responsible for getting and putting the data from and to SDRAM memory during IP core operation.

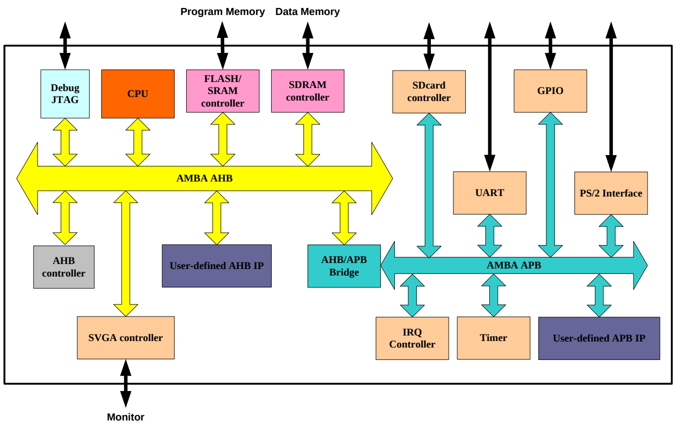


Fig. 3. Top-level architecture of SoC testbed.

IV. Validation Methodology

The system-on-a-chip (SoC) design and verification flow is shown in Fig. 4. In SoC design methodology, system-level design is implemented after the system specification was defined. A high-level description of application/algorithm is developed, which describes the architecture of the design by using the C language (so called C-Model) for simulating and analyzing different parameters of target system architecture, as well as verifying the design against the functional requirements.

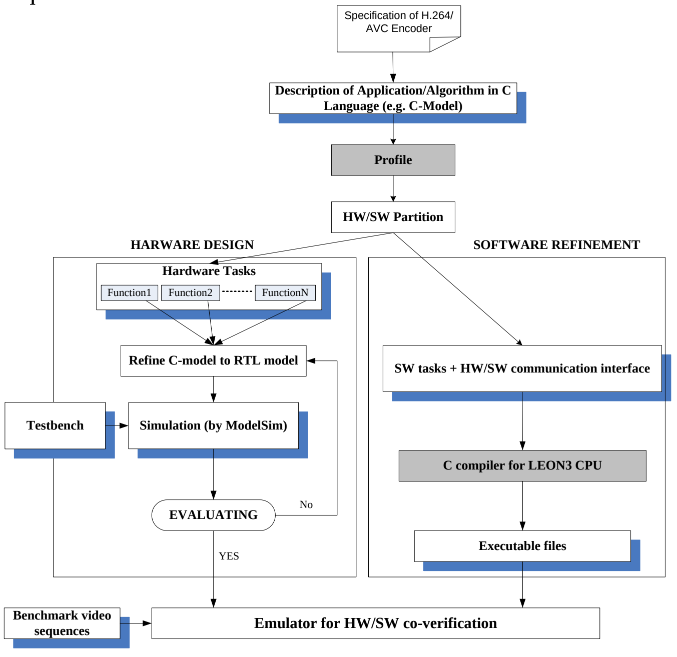

Specification of H.264/ AVC Encoder
Description of Application/Algorithm in C Language (e.g. C-Model)
Profile
HW/SW Partition
HARWARE DESIGN
SOFTWARE REFINEMENT
Hardware Tasks
Function1
Function2
FunctionN
Refine C-model to RTL model
Testbench
Simulation (by ModelSim)
EVALUATING
No
YES
SW tasks + HW/SW communication interface
C compiler for LEON3 CPU
Executable files
Benchmark video sequences
Emulator for HW/SW co-verification

Fig. 4. Design and verification Flow.

The C-Model then is analyzed for identifying the most time-consumption parts by a profiler (e.g. GNU profiler (gprof)). The code-level refinement can be required to modularize the C-Model. Based on the results from previous phases, HW/SW (Hardware/Software) partition phase will partition the overall computation of the algorithm into HW tasks and SW tasks. Each HW task, which is equivalent to a function in C-Model, will be extracted and refined to RTL model. The individual blocks can then be verified in isolation with a suitable test-bench. After RTL model of the module has been verified by simulation, it can be evaluated further through RTL/C co-emulation environment on the SoC testbed as shown in Fig. 5b. For that purpose, some refinement to C-model is also implemented, such as replacing the module, which needs to be verified at RTL level, with the communication interface that drives the corresponding IP core. The HW and SW tasks may communicate data with each other through the off-chip SDRAM. Finally, the software tasks and communication tasks will be compiled onto the LEON3 processor by Gaisler's compiler [9].

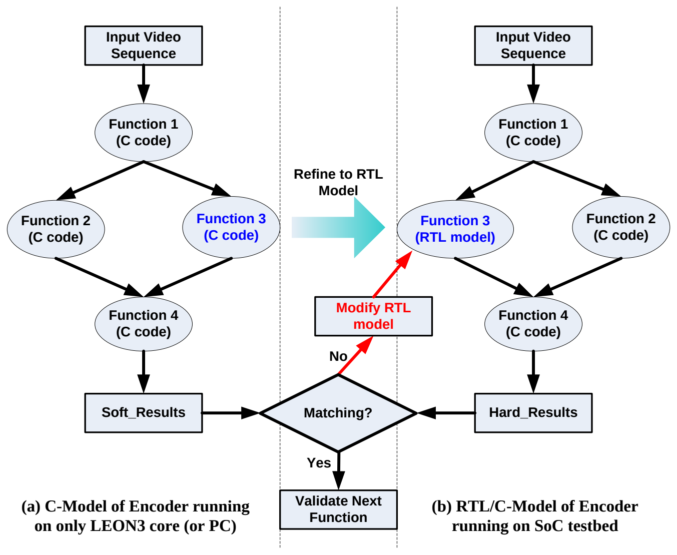


Fig. 5. Method for validating the RTL design on the SoC testbed.

V. APPLICATION TO H.264/AVC VIDEO ENCODER.

The H.264/AVC encoder is evaluated with CIF video sequences by both C/RTL simulation and emulation. Firstly, the encoder is divided into several functional modules, which relate to each other by a Control and Data Flow Graph (CDFG). Next, a C software model (C-Model as shown in Fig. 5a) of the encoder is built for untimed functional verification. The C-Model first run on only LEON3 processor of the SoC testbed for generating the result files that is called Soft_results as shown in Fig. 5a.

Once the design has been refined to the RTL model, and evaluated in terms of function and performance by RTL level simulation, it can be evaluated further through emulation environment on the SoC testbed. A RTL/C co-emulation model as shown in Fig. 5b is used for verifying the functionality and performance of the implementation of modules or the complete encoder. To verify one certain module (e.g. IME) at RTL-level, this functional module is refined independently to RTL model. At each time, only one function will be verified at the RTL level, the other functions are executed by the C-model running on the LEON3 core. The output result file from co-emulator (called Hard-result as shown in Fig. 5b) is compared with the result from the C-model to validate: if matching, it may continue to simulate and verify the implementation at the RTL-level for the other modules.

After all of functional modules are already validated, they are integrated together into a complete encoder, and continue to be verified on the emulator.

The process of emulating the encoder on the emulation board is described briefly as follows (Fig. 6). Firstly, Linux OS and C-model encoder is compiled and loaded into the Flash memory of the SoC testbed. Next, the benchmark video files are copied into an SDcard, which inserted in the SDcard slot afterwards. These source video files are in 'raw' YCbCr format at CIF resolution. After resetting the system, C-model encoder running on LEON3 core analyses the encoding parameters, gets the video frame one-by-one from FLASH memory and writes to SDRAM memory, and starts the encoding process. When software program executes to the location at where the module has been replaced by the communication interface, it transfers all of the necessary control parameters to DMA in wrapper and passes bus control to wrapper (i.e. phase (1) in Fig. 6). DMA will read data from SDRAM and trigger IP core operation (i.e. phase (2) in Fig. 6). DMA is also in charge of writing the result back to SDRAM. When the hardware IP core has done its task, it generates an interrupt signal to notice LEON3, and returns bus control to the CPU (i.e. phase (3) in Fig. 6).

The encoder creates a reconstructed video file, which is identical to decoded video file by a decoder. Therefore, it is able to display the reconstructed frames on a LCD monitor via SVGA controller for evaluating the visual quality of the decoded video file.

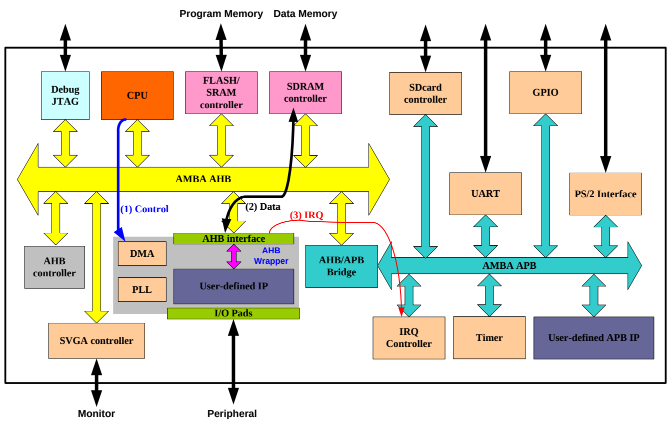
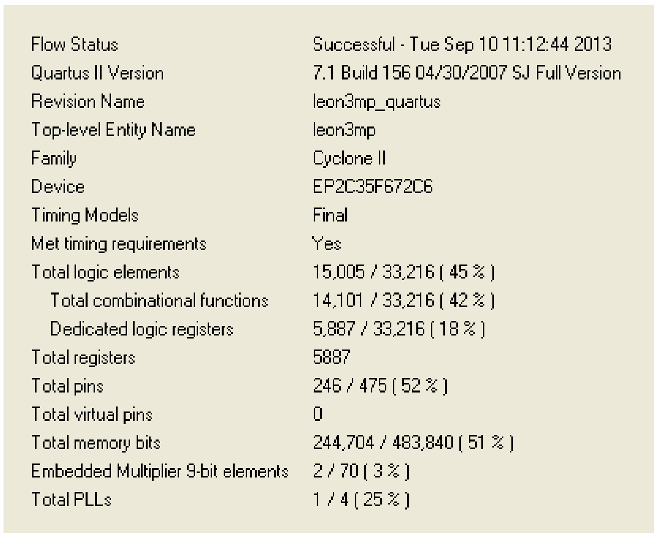


Fig. 6. The process of validating one IP core on the emulation board

Synthesis results of the SoC testbed are reported by Altera Quartus II as shown in Fig. 7.



Flow Status	Successful - Tue Sep 10 11:12:44 2013
Quartus II Version	7.1 Build 156 04/30/2007 SJ Full Version
Revision Name	leon3mp_quartus
Top-level Entity Name	leon3mp
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Met timing requirements	Yes
Total logic elements	15,005 / 33,216 (45 %)
Total combinational functions	14,101 / 33,216 (42 %)
Dedicated logic registers	5,887 / 33,216 (18 %)
Total registers	5887
Total pins	246 / 475 (52 %)
Total virtual pins	0
Total memory bits	244,704 / 483,840 (51 %)
Embedded Multiplier 9-bit elements	2 / 70 (3 %)
Total PLLs	1 / 4 (25 %)

Fig. 7. Compilation Report by Altera Quartus II.

VI. CONCLUSIONS

A SoC testbed and platform-based verification method for validating the hardware design of H.264/AVC video encoder has been presented in the paper. Hardware modules are connected to the system designed around LEON3 processor as custom hardware blocks for HW/SW co-emulation. The interface between the hardware module and SoC is done through the wrapper, so it is quite simple for application, and saves developing time. The experimental results prove that the SoC testbed is valuable to ASIC research and design.

REFERENCES

[1] Rashinkar P., System-On-a-Chip Verification Methodology and Techniques, USA Kluwer Academic, Publishers, 2001.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. “Overview of the H.264/AVC Video Coding Standard”, *IEEE Trans. on Circuits and Systems for Video Technology*, Vol.13, no. 7, pp. 560-576, 2003.

[3] I. E. G. Richardson. *H.264 and MPEG-4 Video Compression*. John Willey & Sons, New York, NY, USA, 2003.

[4] Xuan-Tu Tran, Van-Huan Tran, “An Efficient Architecture of Forward Transforms and Quantization for H.264/AVC Codecs”, *REV Journal on Electronics and Communications JEC*, Vol. 1, no. 2, pp. 122-129, 2011.

[5] Nam-Khanh Dang, Xuan-Tu Tran. A VLSI Implementation for Inter-Prediction Module in H.264/AVC Encoders. In Proceedings of *the 2013 IEICE International Conference on Integrated Circuits, Devices, and Verification (ICDV 2013)*, Ho Chi Minh city, Vietnam, November 2013.

[6] Ngoc-Mai Nguyen, Xuan-Tu Tran, Pascal Vivet, Suzanne Lesecq. An Efficient Context Adaptive Variable Length Coding Architecture for H.264/AVC Video Encoders. In *Proceedings of the 2012 International Conference on Advanced Technologies for Communications (ATC 2012)*, pp. 158-164, Hanoi, October 2012.

[7] Ngoc-Mai Nguyen, Edith Beigne, Suzanne Lesecq, Pascal Vivet, Duy-Hieu Bui, Xuan-Tu Tran. Hardware Implementation for Entropy Coding and Byte Stream Packing Engine in H.264/AVC. In Proceedings of *the 2013 International Conference on Advanced Technologies for Communications (ATC 2013)*, pp. 360-365, October 2013.

[8] Gaisler Research, “GRLIB IP Core User’s Manual”, Version 1.3.0-b4133, Aug. 2013.

[9] Gaisler Research, Bare-C Cross-Compiler System for LEON3 User’s Manual.