

# IP Integration Overhead Analysis in System-on-Chip Video Encoder

Antti Rasmus, Ari Kulmala, Erno Salminen, and Timo D. Hämäläinen  
Tampere University of Technology, Institute of Digital and Computer Systems,  
P.O. Box 553, Korkeakoulunkatu 1, FI-33101 Tampere, Finland  
{antti.rasmus, ari.kulmala}@tut.fi

**Abstract**—Current system-on-chip implementations integrate IP blocks from different vendors. Typical problems are incompatibility and integration overheads. This paper presents a case study of integrating two black-box hardware accelerators into highly scalable and modular multiprocessor system-on-chip architecture. The integration was implemented by creating two wrapper components that adapt the interfaces of the hardware accelerators for the used architecture and on-chip network. The benefit of the accelerators was measured in three different configurations and especially the execution time overheads caused by the software, data delivery, and shared resource contention were extracted and analyzed in MPEG-4 encoder. The overheads increase the function runtime up to 20x compared to the ideal acceleration. In addition, the accelerator that seemed to be more efficient performed worse in practice. As a conclusion, it is pointed out that the integration induces great overhead to the execution time, rendering a-few-clock-cycle optimizations within the accelerator meaningless.

## I. INTRODUCTION

Typical system on chip (SoC) contains several intellectual property (IP) components such as general-purpose processors, on-chip memories, dedicated hardware components like hardware accelerators, an on-chip communication network, and external interfaces [1]. Hardware accelerators are required when higher throughput, lower latency, area or power consumption is required. This paper presents a case study of integrating two hardware accelerators in order to reduce processor load and improve existing embedded MPEG-4 video encoder application performance on FPGA.

For IP block integration, Wagner *et al.* [2] proposed three main strategies: standard based, IP derivation, and automated communication synthesis. However, creating a wrapper component manually is still popular and often the only possible choice. Unfortunately, wrappers increase the final SoC area and decrease system performance [3]. However, most papers do not concentrate on quantitative analysis of the associated performance overheads. These include the software overheads, data delivery delays, and shared resource contentions. Together they are here called *integration overhead* that is added to the pure hardware acceleration execution time. The actual software function execution time, pure hardware accelerator execution time, and integration overhead components are depicted in Fig. 1. The figure is simplified for clarity.

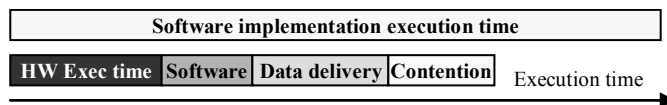


Fig. 1. Introduction to the integration overhead.

It is common that the IP block based hardware implementations of different applications are optimized to reach smaller silicon area, lower latency, or execution time, for example in [4] and [5]. In the figure, this would reduce the HW execution time. As other components remain constant, it reduces the benefit of such optimizations.

In this paper, the integration of the black-box hardware accelerators is implemented by manually creating two wrapper components. The speedup is measured in MPEG-4 SP encoder. In addition, the integration overhead components are separated and analyzed.

The structure of the paper is as follows. In Section II, the system under study is described. The hardware accelerators and the created wrapper components are introduced in the Section III. In Section IV, we describe the methods for measuring the performance that is followed by results and analysis in Section V. Section VI concludes the paper.

## II. THE SOC ARCHITECTURE

The parallel SoC architecture under study was first introduced by Kulmala *et al.* [6]. There were originally three different types of IP components: a master CPU, a slave CPU, and a synchronous dynamic random access memory (SDRAM) controller. These components are interconnected by an on-chip communication network, Heterogeneous IP Block Interconnection (HIBI) [7]. HIBI is configured as a bus with 32-bit data width. It consists of components called HIBI wrappers that implement the simple FIFO interface to IP components.

The architecture is modular and scalable, as the number of processors can be altered effortlessly and rapidly. The master processor controls the external IO and the slave CPUs, whereas the slave CPUs perform the actual computation. The architecture utilizes Altera's Nios II [8] processors. The external SDRAM is used as a primary data memory and it is accessed via the 32-bit on-chip SDRAM controller.

The resource manager (RM) implements centralized resource access control. It was introduced to the system to manage the use of hardware accelerator components among the parallel working processors. The RM is in charge of granting the processors turns to access the hardware accelerators.

### III. THE VIDEO ENCODER APPLICATION AND THE HARDWARE ACCELERATORS

The video encoder application was profiled [9] and it was discovered that motion estimation (ME) and discrete cosine transform (DCT), quantization, inverse quantization, and inverse DCT consume a notable part of the execution time. The combination of the four latter operations is referred to as DCT-Q-IDCT in this paper. In a repeated study, we found that the ME and DCT-Q-IDCT execution times are 21% and 16% of the whole CPU time respectively. Thus, these functions were the best choices to be accelerated. In addition, processors compute other encoding tasks in parallel while the hardware accelerators are running.

ME hardware accelerator performs the full-pixel motion estimation where a macroblock is compared to a 3-by-3 macroblock search area, and the direction of the motion is calculated. The accelerator features 128-bit wide buses in and out. It requires 2560 bytes (10 macroblocks) and outputs 262 bytes (1 best matching macroblock, a SAD value, and motion vectors) of data. The DCT-Q-IDCT accelerator has 9-bit wide bus in and 8 bits out, and transfers 385 and 768 bytes of data in and out respectively.

The accelerators have proprietary interfaces and they are originally optimized for speed, not for integration. Neither of the interfaces matches with HIBI wrapper. The dataflow to the accelerators also differ. The CPU directly sends and receives the data to and from the DCT-Q-IDCT, whereas only a command is sent to ME to fetch data from SDRAM. The results of the ME are sent directly back to the CPU. Thus, ME wrapper has to implement the protocol to access the SDRAM. In addition, the ME wrapper permutes the results, since ME gives the result data in different order than the CPUs require. These differences in the interfaces and functionality forced to create two separate wrappers: a simpler one for DCT-Q-IDCT and a more complex one for the ME. Fig. 2 shows the architecture including the RM, the new accelerators, and the wrapper components.

### IV. THE CONFIGURATIONS FOR MEASURING EXECUTION TIME

Our objective was to study performance, which is inversely proportional to the execution time. Clock cycles were measured using processors' timers and hardware monitor.

The timers were used to measure the encoding frame rate, which reflects the actual hardware acceleration benefit to the application. As the architecture is designed to be scalable by alternating the number of encoding slave CPUs, the encoding

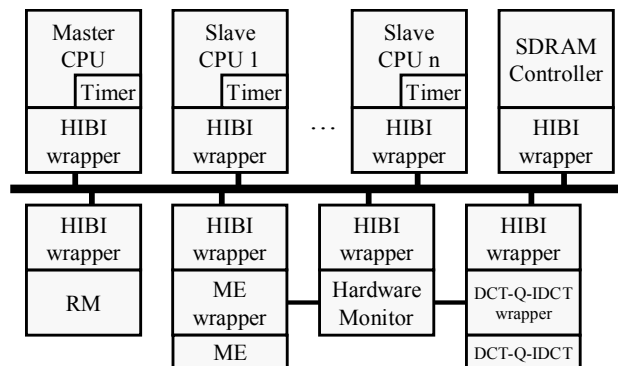


Fig. 2. The used SoC architecture.

frame rate was measured using one, two, and three encoding slave CPUs using either none, one, or both of the hardware accelerators.

The timers of the processors were also used to measure clock cycles spent on DCT-Q-IDCT and ME functions from the processor's point of view, when they are accessed from the software. These timers measure clock cycles between the driver function call that initiates the accelerated task and the interrupt request upon the arrival of the results back to the CPU. Hence, the time includes all overheads. In addition, the execution time of the original software function was measured as a comparison.

A synthesizable hardware monitor collects more detailed results on the accelerators and their wrappers. It is designed to distinguish the overheads, caused by the software, shared resource contention, and data delivery, for the maximum accuracy. Hardware acceleration execution time was measured in three different configurations: simulation, simple FPGA test, and encoder.

Simulation is the ideal reference situation where hardware accelerator is attached to the simulation test bench. It measures pure hardware accelerator execution time, since wrappers are not used at this point and data are always available to the hardware accelerator.

In simple FPGA test, the wrapper component is introduced and attached between the hardware accelerator and HIBI wrapper. The test architecture also contains Nios II CPU and the SDRAM controller and is the minimum that is required to run the accelerators on FPGA with a processor. In this limited system, the CPU runs a test program, which is dedicated to run the accelerators consecutively. All the resources are available when required, since there is no parallel processing in the system. This configuration is used to measure data delivery expenses on chip and software overheads.

The last configuration, video encoder, is the final application with two encoding slave CPUs, both of the hardware accelerators, the RM, and SDRAM controller. With this configuration, the contention of shared resources such as bus, SDRAM, and accelerators can be extracted. In addition, the delay of using the RM is counted in the contention section.

In addition, same kind of configuration with slave processor count from one to three is used to measure the video encoding speed as well.

## V. RESULTS AND ANALYSES

All represented results are obtained by using averages of three different 90-frame-long standard QCIF-sized video sequences called *Carphone*, *Foreman*, and *Akiyo*. The system frequency is 50 MHz and sequences were run 3 times.

### A. The video encoding speed

The encoding speed is shown in Fig. 3, where the number of processors indicate the numbers of master and slave processors. For example, 1+2 denotes one master and two slaves. The video encoding speed scales up flexibly with all four combinations of hardware accelerators. Alone, the DCT-Q-IDCT provides slightly better performance gain than the ME. Starting from the minimum configuration with 1+1 CPUs and no acceleration, one can see that it is more beneficial to add a single CPU than any acceleration. However, the case with the both accelerators and 1+2 CPUs is faster than 1+3 CPUs without acceleration. DCT-Q-IDCT provides 20-21% speed-up to the frame rate, ME 15-17% and both together 40-47%.

### B. The DCT-Q-IDCT execution time

Fig. 4 illustrates the results of the DCT-Q-IDCT function execution times. Firstly, the difference in cycle count between simulation and simple FPGA test is 50% due to the software overhead, wrapper component, and introduced HIBI communication. Secondly, the difference between the simple FPGA test and HW in encoder is 49%. Here, the increase is due to the contention of communication and the RM access that also includes the waiting time for getting access to the accelerator if it is being used at the time of query. Hence, the average access time increases, which consequently adds delay to the DCT-Q-IDCT transactions. Overall, the difference between simulation results and the actual encoding is 122%.

### C. The ME execution time

In ME measurements, the results between the three test configurations vary considerably as shown in Fig. 5. When compared to the simulation, execution time shows in simple FPGA test 890% and in encoder 2160% increment.

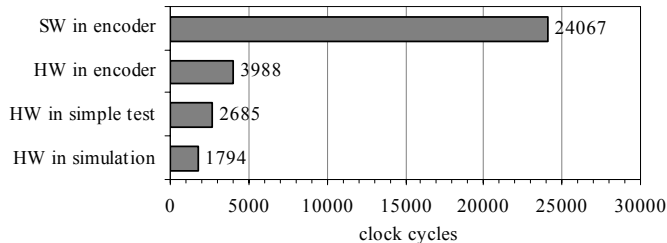


Fig. 4. Execution times of DCT-Q-IDCT function with different configurations measured in clock cycles with timers, except the simulation.

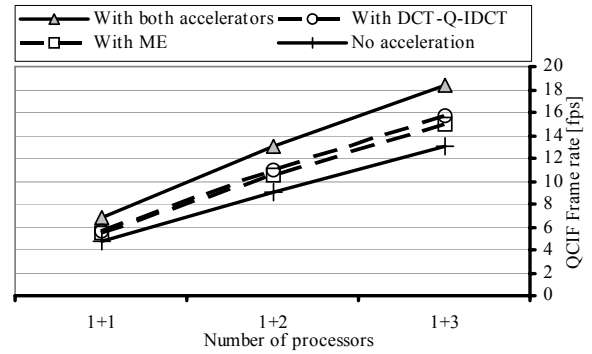


Fig. 3. The encoding speed with different accelerations options.

In simple FPGA test and accelerated encoder, the wrapper has to do the permutation operation to the result macroblock that causes extra latency. In addition, ME hardware has to wait for data since it has 128-bit input and SDRAM is only 32 bits wide.

In simple test configuration, the software and data delivery increase the execution time like in the case of DCT-Q-IDCT. In encoder, large amount of data is transferred to and from the SDRAM controller over the on-chip network, and the HIBI bus utilization grows when multiple parallel operations are running in the system. Here, the SDRAM data fetching operation slows down dramatically. This explains the more than doubled execution time in encoder when compared to simple FPGA test situation. Using wider SDRAM would be beneficial but was not possible in the current prototype, as the development board does not support one.

### D. The integration overhead components

The breakdown of the integration overhead is shown in Fig. 6 based on the results of HW monitor and CPU timers. The proportions of integration overhead are 96% and 55% with ME and DCT-Q-IDCT, respectively. Thus, the pure ME hardware accelerator execution time plays minor role. The software expense is tolerable, but data delivery delay and contention of shared resources are immense. These are due to the SDRAM and bus width issues.

In contrast to the ME, the hardware execution time of the DCT-Q-IDCT is in the major role. Because of the small bus widths and well-parallelized computation and communication, the DCT-Q-IDCT does not consume time on waiting data on

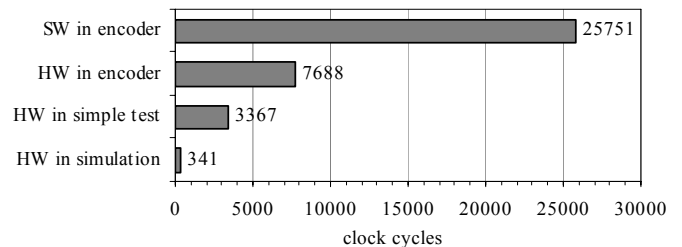


Fig. 5. Execution times of ME function with different configurations. Measured in clock cycles with CPU timers, except the simulation.

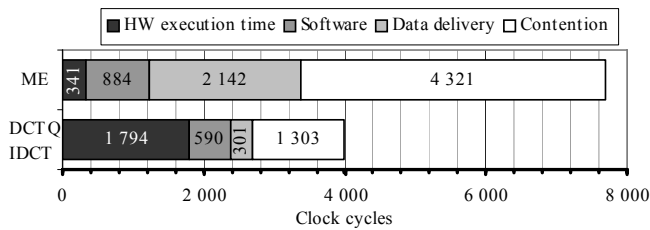


Fig. 6. The components of execution times on video encoder.

simple test. This, however, is the case in encoder, if the interconnection is somewhat utilized. In addition, the RM query delay contributes clock cycles in the contention bar.

The contention is relatively bigger with the ME because the ME has longer data transfers, and instead of only competing for HIBI like the DCT-Q-IDCT, the ME has to wait for the access to SDRAM as well.

ME has clearly shorter pure hardware execution time. However, the overall time needed for ME during the encoding is larger due to bigger overheads. Hence, comparing the accelerators based on their ideal times in testbench proves to be misleading. Similarly, running the accelerator with higher frequency does not offer notable speed-up since the overheads remain the same.

#### E. The area in FPGA

The area usage is obtained from the report of Altera's Quartus II tool after completion of synthesis, placement, and route operations on Altera's Stratix 1S40 FPGA chip. The areas of hardware accelerators, their wrappers, the resource manager, memory controller, hardware monitor, a HIBI wrapper, and Nios processor are depicted in the Fig. 7. Nios II processor area includes timer, 64 KB data memory, 8 KB instruction cache, and Nios-to-HIBI DMA controller. The DCT-Q-IDCT and ME wrapper sizes are 27% and 65% from the accelerator areas respectively. The permutation logic greatly increase the logic cell usage of the ME wrapper. However, as the ME hardware accelerator utilizes on-chip memory, the areas of the wrapper and the accelerator are not fully comparable.

### VI. CONCLUSIONS

The purpose of this paper was to illustrate the performance expenses of IP block integration in general with two real-life examples. In our case, the proportions of integration overheads for the execution time were 96% and 55%. They depend on the compatibility of the component and the architecture. In addition, the actual performance gain of the accelerators differs from that the background data indicate: the accelerator that seemed to be more efficient performed worse in our system.

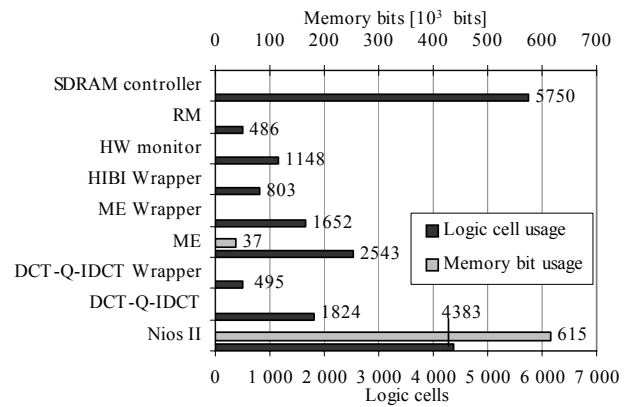


Fig. 7. The logic cell and memory bit usages of components on FPGA.

Such high variation in execution times complicates SoC design, and should not be ignored in design time especially in real-time systems. In addition, this finding renders few clock cycle optimizations in computation marginal, when the integration overhead contributes huge performance decrement due to interface incompatibility. This finding highlights the importance of interface and communication design in IP blocks and architectures.

### REFERENCES

- [1] A. Jerraya and W. Wolf, "Multiprocessor system-on-chips," London: Morgan Kaufman, 2004, pp. 1-18
- [2] F. Wagner, W. Cesário, L. Carro, and A. Jerraya, "Strategies for the integration of hardware and software IP components in embedded systems-on-chip," *Integration, the VLSI Journal*, Sep. 2004, vol. 37, Issue 4, pp. 223-252
- [3] P. Coussy, D. Gnaedig, A. Nafkha, A. Baganne, E. Boutillon, E. Martin, "A methodology for IP integration into DSP SoC: A case study of a map algorithm for turbo decoder," in *Proc. ICASSP*, 2004, pp. 45-48.
- [4] D. Guevorkian, A. Launainen, P. Liuha, V. Lappalainen, "Architectures for the sum of absolute differences operation," *Signal processing Systems*, Oct. 2002, pp. 57-62
- [5] J. Lee, N. Vijaykrishnan, and M. J. Irwin, "Inverse Discrete Cosine Transform Architecture Exploiting Sparseness and Symmetry Properties," *Circuits and Systems for Video Technology, IEEE Transactions on*, May 2006, Vol. 16, Issue 5, pp. 655-662
- [6] A. Kulmala, O. Lehtoranta, T. D. Hämäläinen, and M. Hännikäinen, "Scalable MPEG-4 Encoder on FPGA Multiprocessor SoC," *EURASIP Journal on Embedded Systems*, Vol. 2006, Hindawi Publishing Corporation
- [7] E. Salminen, T. Kangas, J. Riihimäki, V. Lahtinen, K. Kuusilinna, and T. Hämäläinen, "HIBI Communication Network for System-on-Chip," *Journal of VLSI Signal Processing*, June 1, 2006, Vol. 43, Issue 2-3, pp. 185-205, Springer-Verlag
- [8] Altera Corporation, "Nios II Processor Reference Handbook," version NII5V1-6.1, November 2006.
- [9] O. Lehtoranta, T. D. Hämäläinen, and V. Lappalainen, "Parallel implementation of video encoder on quad DSP system", *Microprocessors and Microsystems*, February 25, 2002, Vol. 26, Issue 1, pp. 1-15, Elsevier.