Implementation and Analysis of Architectures for the 4x4 2-D Forward Hadamard Transform of H.264/AVC

Daniel Palomino, Guilherme Corrêa, Robson Dornelles, Felipe Sampaio, Diego Noble, Luciano Agostini

Group of Architectures and Integrated Circuits (GACI), Department of Informatics
Federal University of Pelotas, Pelotas-RS, Brazil
{danielp.ifm, gcorrea_ifm, rdornelles.ifm, fsampaio.ifm, dnoble.ifm, agostini}@ufpel.edu.br

Abstract. This paper presents the design and investigation of low latency and high throughput dedicated architectures for the 4x4 Forward Hadamard transform of the H.264/AVC standard. The architectures were designed focusing on a good balancing between throughput and low latency, enabling its integration with the Intra Prediction coder module. Eight versions of this transform were defined, described in VHDL and synthesized to Altera Stratix II FPGAs, using the Quartus II software. The architectures which best fits the goal of this work present throughputs of 2.6 and 4.3 billions of input samples per second, with latencies of 1 and 2 clock cycles, respectively.

Keywords: H.264/AVC, video compression, Hadamard transform, intra prediction, FPGA based design.

1 Introduction

The H.264/AVC is the newest video compression standard. It was developed by experts from ITU-T and ISO/IEC aiming to double the compression rate achieved by the previous standards [1], [2]. The research on hardware architectures for the H.264/AVC standard is part of the Brazilian researchers' effort to develop the Brazilian System of Digital Television, the SBTVD [3].

Fig. 1 shows the encoder diagram block, which is composed by: motion estimation (ME in Fig. 1), motion compensation (MC in Fig. 1), intra prediction, loop filter, entropy coder, forward and inverse quantization (Q and Q^{-1} in Fig. 1) and forward and inverse transform (T and T^{-1} in Fig. 1).

T, T^{-1}, Q and Q^{-1} modules belong to the critical path of intra prediction in the encoder, since the samples used in the intra prediction module are reconstructed based on the current frame. It means that the predicted image blocks must be processed by the T, Q, Q^{-1} and T^{-1} modules before being used as references to process the next image block of the current frame. This loop requires an amount of time in which the intra prediction block must wait to the reconstructed data. Then, the mentioned loop design is critical, since there is a relationship between the performance of the T, T^{-1}, Q and Q^{-1} modules and the time in which the intra prediction module stays stopped. A

good solution for this situation is to design a dedicated loop of transforms and quantization targeting the Intra Prediction and other loop for the other coder modules.

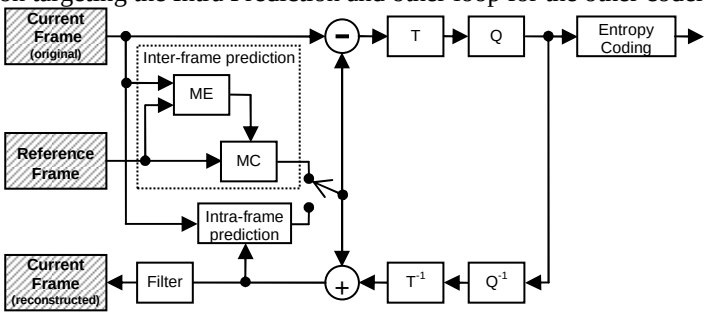

Fig. 1. Encoder diagram block of H.264/AVC.

The goal of this work is to investigate architectural alternatives for the 4x4 forward Hadamard transform. Eight different architectures were implemented in FPGAs and compared in terms of hardware resources use, critical delay and latency. This work aims at finding the architecture that better fits in the Intra Prediction requirements.

The rest of this paper is organized as follows. Section 2 explains the transforms of the H.264/AVC standard. Section 3 describes all the designed architectures. Section 4 presents the synthesis results and a discussion about them. Section 5 shows related works. Finally, section 6 concludes this work.

2 Transforms in the H.264/AVC Standard

The T module in Fig. 1 contains the forward transforms: 4x4 FDCT (Forward Discrete Cosine Transform) and the 4x4 and 2x2 Forward Hadamard transforms. Integer arithmetic is used in all transforms in order to avoid differences between inverse and forward transforms and also to allow a simpler hardware design [3].

The inputs of the T module are 4x4 blocks of residual information generated in the prediction step. The 2-D FDCT is the first transform applied and it is applied for all the input samples, including luminance and chrominance samples. If the samples are of luminance and were predicted using the intra 16x16 mode or if the samples are of chrominance, the Forward Hadamard transform is applied in the DC coefficients resulting from the 2-D FDCT. The DC coefficients of the luminance blocks are processed by the 4x4 Forward Hadamard and the DC coefficients of chrominance blocks are processed by the 2x2 Forward Hadamard [4].

The 4x4 Forward Hadamard transform defined in the H.264/AVC standard is presented in (1). The Hadamard transform does not process all information that pass through the T module so it stays idle during a period of time.

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} \\ W_D \\ \\ \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2 \qquad (1)$$

In order to implement this transform in an architectural level, its mathematical definition was decomposed in an algorithm that is shown in Table 1.

Table 1. Algorithm used in the computation of the 4x4 Hadamard transform.

$a_0 = W_0 + W_4$	$b_0 = a_0 + a_1$	$c_0 = b_0 + b_1$	$S_0 = (c_0 + c_1)/2$
$a_1 = W_8 + W_{12}$	$b_1 = a_2 + a_3$	$c_1 = b_2 + b_3$	$S_1 = (c_0 - c_1)/2$
$a_2 = W_1 + W_5$	$b_2 = a_4 + a_5$	$c_2 = b_0 - b_1$	$S_2 = (c_2 - c_3)/2$
$a_3 = W_9 + W_{13}$	$b_3 = a_6 + a_7$	$c_3 = b_2 - b_3$	$S_3 = (c_2 + c_3)/2$
$a_4 = W_2 + W_6$	$b_4 = a_0 - a_1$	$c_4 = b_4 + b_5$	$S_4 = (c_4 + c_5)/2$
$a_5 = W_{10} + W_{14}$	$b_5 = a_2 - a_3$	$c_5 = b_6 + b_7$	$S_5 = (c_4 - c_5)/2$
$a_6 = W_3 + W_7$	$b_6 = a_4 - a_5$	$c_6 = b_4 - b_5$	$S_6 = (c_6 - c_7)/2$
$a_7 = W_{11} + W_{15}$	$b_7 = a_6 - a_7$	$c_7 = b_6 - b_7$	$S_7 = (c_6 + c_7)/2$
$a_8 = W_0 - W_4$	$b_8 = a_8 - a_9$	$c_8 = b_8 + b_9$	$S_8 = (c_8 + c_9)/2$
$a_9 = W_8 - W_{12}$	$b_9 = a_{10} - a_{11}$	$c_9 = b_{10} + b_{11}$	$S_9 = (c_8 - c_9)/2$
$a_{10} = W_1 - W_5$	$b_{10} = a_{12} - a_{13}$	$c_{10} = b_8 - b_9$	$S_{10} = (c_{10} - c_{11})/2$
$a_{11} = W_9 - W_{13}$	$b_{11} = a_{14} - a_{15}$	$c_{11} = b_{10} - b_{11}$	$S_{11} = (c_{10} + c_{11})/2$
$a_{12} = W_2 - W_6$	$b_{12} = a_8 + a_9$	$c_{12} = b_{12} + b_{13}$	$S_{12} = (c_{12} + c_{13})/2$
$a_{13} = W_{10} - W_{14}$	$b_{13} = a_{10} + a_{11}$	$c_{13} = b_{14} + b_{15}$	$S_{13} = (c_{12} - c_{13})/2$
$a_{14} = W_3 - W_7$	$b_{14} = a_{12} + a_{13}$	$c_{14} = b_{12} - b_{13}$	$S_{14} = (c_{14} - c_{15})/2$
$a_{15} = W_{11} - W_{15}$	$b_{15} = a_{14} + a_{15}$	$c_{15} = b_{14} - b_{15}$	$S_{15} = (c_{14} + c_{15})/2$

3 Designed Architectures

Several architectures for the 4x4 2-D Forward Hadamard transform could be developed as long as the algorithm showed in Table 1 is respected. In this work, we present the implementation and analysis of eight different architectures with no separability exploration [4]. This means that the architectures do not compute each dimension of the 2-D transform separately in one dimension. Instead, they compute the two dimensions together. This allows a reduction of data dependencies and a generation of architectures with different levels of parallelism.

3.1 1-Stage Pipeline Architecture with 16 Input Samples per Cycle (1P16S)

The architecture with one pipeline stage and with a parallelism level of 16 samples per cycle (1P16S) is the simplest one implemented in this work since it does not make use of any register barrier for a pipeline execution. This solution is presented in Fig 2 and it includes all the 4x4 Hadamard operations inside only one pipeline stage. In each cycle, sixteen input samples are processed by the architecture. Considering the fact that there are no registers in this architecture, its implementation requires less hardware resources than the pipelined versions. . However, a higher critical delay is expected because the critical path of the circuit is composed by four adders.

3.2 4-Stage Pipeline Architecture with 16 Input Samples per Cycle (4P16S)

The 4-stage pipeline architecture with 16 input samples per cycle (4P16S) is a completely parallel version of the Hadarmard architecture which consumes 16 input

samples per cycle. In each cycle, 16 samples are processed in each pipeline stage and the results are stored in the corresponding pipeline registers.

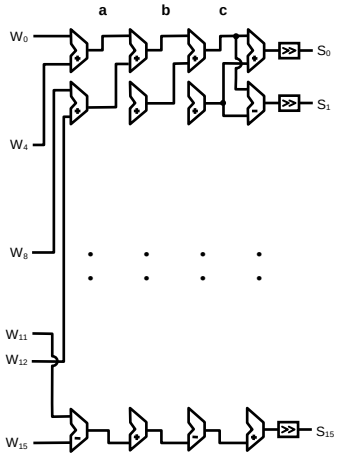

Fig. 2. 1-stage pipeline architecture with 16 input samples per cycle.

In this solution, presented in Fig. 3, only one cycle is demanded to perform the additions in each stage. This way, the throughput of the architecture is 16 samples per clock cycle and its latency is only 4 clock cycles. Based on Table 1, it is relatively easy to obtain the diagram shown in Fig. 3. Each column of the table corresponds to a pipeline stage and each line corresponds to an adder of the Hadamard architecture.

Although this architecture presents a great throughput, two problems are found in this solution. The first one is related to the high use of resources, since 16 adders and 16 registers are used in each stage. The second problem is related to routing due to the large number of wires used in the input and in the output of this architecture.

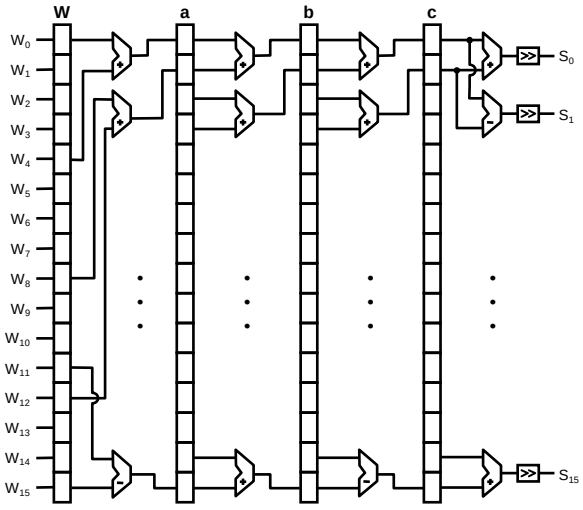

Fig. 3. 4-stage pipeline architecture with 16 input samples per cycle.

3.3 2-Stage Pipeline Architecture with 16 Input Samples per Cycle (2P16S)

The architecture with 2-stage pipeline and with 16 input samples per cycle (2P16S) is based on the version described in section 3.2. However, in this solution only 2 pipeline stages are used, which leads to a 2 clock cycles latency. Another advantage is the lower use of hardware resources since only two register barriers are used.

The critical path of this version is composed by two adders implying in a lower frequency when compared to the architecture 4P16S, presented in section 3.2.

3.4 4-Stage Pipeline Architecture with 4 Input Samples per Cycle (4P4S)

The architecture with a pipeline of four stages and with a parallelism level of four inputs per cycle (4P4S) produces and consumes 4 input samples at each cycle. This way, 4 ping-pong registers [5] (with 4 slots) and 4 adders are needed in each pipeline stage, as shown in Fig. 4. The ping-pong registers are used in order to provide synchronized storage resources for the input samples, since not all the data are available at the same time in the input of the Hadamard transform. When the ping-pong registers are filled with data, the computation can be performed by the adders and the results can be stored in the next pipeline barrier. This solution needs 20 cycles to completely process the first 16 input samples and 4 more cycles to process the subsequent 16 samples. This solution presents a latency of 16 clock cycles.

The huge number of registers is the main disadvantage presented in this version. On the other hand, in each stage only 4 adders are required, instead of the 16 adders presented in the previous architectures: 1P16S, 2P16S and 4P16S.

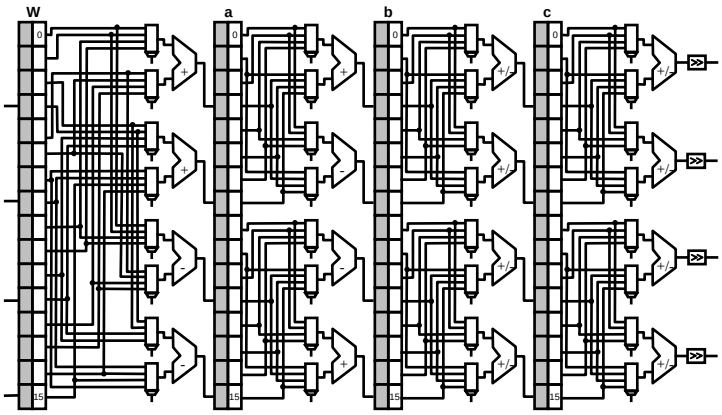

Fig. 4. 4-stage pipeline architecture with 4 input samples per cycle.

3.5 2-Stage Pipeline Architecture with 4 Input Samples per Cycle (2P4S)

The 2P4S architecture is organized in a pipeline with two stages and consuming four samples per clock cycle. The difference between this solution and the previous one (4P4S) is in the number of used registers. The 2-stage pipeline architecture demands 2

register blocks instead of the 4 used in the 4P4S version. This architecture was implemented in three different alternatives, as described below.

In the first one (2P4S-2+2), each pipeline stage contains two adders and it is limited by 4 ping-pong registers (with 4 slots). In this case, the second and the forth register barriers presented in the version 2P4S were removed.

In the second alternative (2P4S-3+1), the second and the third register barriers presented in the version 2P4S were removed. Thus, the first pipeline stage is composed by three serial adders and the second stage by just one.

In the third solution (2P4S-1+3), the third and the forth barriers presented in the version 2P4S were taken off in such a way that the first pipeline stage critical path has only one adder and the second is composed by three adders.

The three architectures present the same latency of 8 clock cycles. However, each one shows different results in terms of frequency due to its specific critical path. This three architectures present as advantage a lower number of hardware resources used when compared to the version 4P4S, described in section 3.4. Nevertheless, this characteristic implies in a longer critical path, which leads to a lower operation frequency.

3.6 2-Stage Pipeline Architecture with 2 Input Samples per Cycle (2P2S)

The 2P2S implementation has a pipeline with 2 stages and with a parallelism level of two samples per cycle and it aims to provide a low cost architecture since it uses four adders per pipeline stage. As the architecture has only two pipeline stages, only eight adders are used in the whole project. In this case, the latency is 16 clock cycles which is high when compared to the other versions.

Also, as the input of this architecture is composed by two samples, each ping-pong register must be built with 8 slots instead of the 4 slots of the 2P4S version.

4 Synthesis Results

All the architectures presented in the previous section were described in VHDL, synthesized and validated for the EP2S15F672C3 device from the Altera Stratix II FPGA family using the Altera Quartus II 7.2 software [6]. Also, the timing analysis for each architecture was performed by the Altera Classic Timing Analyzer tool [6].

Table 2 shows the results in terms of frequency and used resources for the mentioned FPGA device. In Table 2, the acronyms presented in each architecture description were used to identify the architectures. The first number of the acronym represents the number of pipeline stages and the second number refers to the number of input samples processed in parallel for each version. For example, the architecture 2P16S was designed with 2 pipeline stages and processing 16 input samples per clock cycle.

The architecture 1P16S presented, as already expected, a low frequency when compared to the other 16-input samples solutions. Also, as in all others 16-input samples architectures, a high number of ALUTs is required because of the number of

used adders. Although the architectures 4P16S and 2P16S use more registers because of the pipeline stages, a higher operation frequency is achieved. The 4-input samples architectures use less parallelism, which leads to a lower number of used ALUTs when compared to the full-parallel versions (1P16S, 2P16S and 4P16S). However, in order to implement the pipeline, these architectures use ping-pong registers barriers, which causes an elevated use of registers (especially high in the 4P4S-2+2 version). The implementation of different 2P4S solutions was made to evaluate the possible results for different pipeline configurations. The 2P2S architecture was developed to reduce the use of ALUTs in relation to the 2P4S versions. However, as shown in Table 2, the proposed version did not reach the expected results and then, this solution will be discarded in the next design efforts.

Table 2. Operation frequency, number of used ALUTs and number of used dedicated logic registers (DLR) of each solution for the Stratix II EP2S15F672C3 device.

Architecture	Frequency (MHz)	#ALUTs	#DLR
1P16S	162.02	681	304
4P16S	448.03	672	756
2P16S	268.31	680	452
4P4S	250.63	320	1052
2P4S-2+2	202.51	510	534
2P4S-3+1	171.00	360	555
2P4S-1+3	198.26	299	506
2P2S	193.27	472	547

Tab. 3 shows the processing rates for all the designed architectures. The throughput data presented in Table 3 was calculated based on the frequency and the input samples per cycle of each architecture. The frames per second values were calculated based on the number of samples of the worst case, which happens when all input blocks were provided by the 16x16 intra prediction. This way, there are 491,520 samples to be processed by the 4x4 2-D Forward Hadamard transform in one QHDTV frame (3840x2048 pixels). The very high throughput presented in Table 3 allows a very high processing rate of QHDTV frames. Then, all presented architectures are able to easily reach real time (30 frames per second) when very high definition videos (like QHDTV) are being processed, which means that all the presented architectures could be used in the T module. Other important information presented in Table 3 is the minimum operation frequency that is necessary to allow the processing of 30 frames per second for each architecture. All presented frequencies are very low and this question is important, since these architectures may be used in designs with other requirements than those presented in this work. These solutions are able to be inserted in a design focusing low power, for example, since they reach real time even running in a very low operation frequency.

Table 3. Latency, throughput, frames per second and minimum frequency to reach 30 frames per second for QHDTV frames of each solution for the Stratix II EP2S15F672C3 device.

Architecture	Latency (cycles)	Throughput (Msamples/s)	QHDTV Frames per second	Min. Freq. to QHDTV @ 30fps (MHz)
1P16S	1	2,592	5,274	0,92
4P16S	4	7,168	14,584	0,92
2P16S	2	4,292	8,734	0,92
4P4S	16	1,002	2,039	3,69
2P4S-2+2	8	810	1,648	3,69
2P4S-3+1	8	684	1,391	3,69
2P4S-1+3	8	793	1,613	3,69
2P2S	16	386	786	7,38

Since this work focuses in high throughput and low latency, to allow real time when Intra Prediction and transforms will be joined, there is a limitation in the choosing of the architecture to be used concerning its latency. Latencies of all solutions are presented in Table 3. As this design space exploration was focused in the Intra Prediction restrictions, the T module must be designed to support the burst of data that is generated by the Intra Prediction module. In function of the data dependencies already mentioned between the Intra Prediction and the transforms and quantization modules, the Intra Prediction must wait for the transforms and quantization to process the next block and the transforms and quantization must wait when this next block is processed by the Intra Prediction. While one 4x4 block is produced by the Hadamard transform, the Intra Prediction coder waits for the reconstruction of this block to generate the next one. This way, the chosen architecture should not present a high latency since there is not a continuous flow of information in the Hadamard transform input and it is also important that the architecture presents a high throughput in order to increase the speed of the computation.

Based on these Intra Prediction requirements and on the data presented in Table 3, the architectures that better fit in the design of the T module are the 1P16S and 2P16S. Both architectures present an acceptable equilibrium between throughput and latency, processing an elevated number of frames per second. Although the version 4P16S can process a higher number of frames per second, this architecture presents a high latency, what can hinder its application in the T module, as explained before.

5 Related Works

It is hard to find, in the literature, related works about this theme and where the architectures were implemented in the same technology, which limits the comparison with the architectures proposed in this work. Although the throughput depends on the technology in which the architecture was implemented, the latency can be analyzed since it is defined solely by the design. Table 4 presents a comparison between the

three most interesting versions designed in this work and six other solutions proposed by other authors. These three architectures were chosen because they are the best options to be integrated with the Intra Prediction coder.

Table 4. Comparison with related works.

Solution	Throughput (Msamples/s)	Latency	Paralelism Level	Technology
Our 4P16S	7,168	4	16	Stratix II
Our 2P16S	4,292	2	16	Stratix II
Agostini [7]	3,499	6	16	0.35µ
Our 1P16S	2,592	1	16	Stratix II
Agostini [7]	1,983	6	16	Stratix
Cheng [8]	800	2	8	0.35µ
Chen [9]	800	−	8	0.18µ
Wang [10]	320	4	4	0.35µ
Lin [11]	273	−	8	0.35µ
Agostini [12]	175	64	1	Virtex 2P

In [7], [8], [9] and [10] the solutions proposed are multitransform architectures, it means they can process the calculations related to all the 4x4 forward and inverse transforms. The solution [7] also performs the 2x2 Hadamard transform. The design proposed in [11] performs the 4x4 2-D Forward Hadamard transform. In addition, this architecture calculates the operations of its related quantization. The design presented in [12] is a dedicated forward transform architecture which is integrated in a complete T module.

The solutions 4P16S and 2P16S, designed in this work, present the highest throughputs when compared to the others published works and only one published work surpasses the throughput reached by 1P16S architecture. Then, the goal of high throughput was reached.

In addition, architectures 1P16S and 2P16S present the lowest latency among all related works (work [8] presents the same latency than 2P16S). Then, another goal of this work was reached: low latency.

A comparison about the relation between the reached throughput and the parallelism level of each solution presented in Table 4 was done and the three solutions designed in this paper present the best relation. This relation shows the high efficiency of these solutions in terms of use of parallelism to support the high throughput. Other important comparison is that the two first solutions presented in Table 4 and designed in this work (4P16S and 2P16S) present the best relation between throughput and latency among all presented solutions. This is an important metric, since this relation is the most important to design a T module which is able to respect the Intra Prediction constraints.

6 Conclusions and Future Work

This paper presented the design and evaluation of several implementations to the 4x4 2-D Forward Hadamard transform, looking for high throughput and low latency,

focusing in the Intra Prediction restrictions. Thus, eight different architectures were designed, using different levels of parallelism and different number of pipeline stages. All architectures were described in VHDL and synthesized targeting to Altera Stratix II FPGA. The architectures designed in this work may be used in the Intra Frame coder loop, since all the designed solutions presented a very high throughput and a low latency.

The best result in terms of latency was found in the 1P16S architecture. The best throughput was achieved by the 4P16S architecture and the lowest hardware cost was reached by the 2P4S-1+3 architecture. The best choice to be used in the Intra Frame coder loop must combine low latency and high throughput, so the best options would be the 1P16S or the 2P16S architecture.

As future work we plan to design the complete T module and the Intra Prediction coder loop.

References

1. Joint Video Team of ITU-T and ISO/IEC JTC 1: Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264) (2003).
2. Sullivan, G. and Wiegand, T.: Video Compression – From Concepts to the H.264/AVC Standard. Proceedings of the IEEE, v. 93, n. 1, pp. 18-31 (2005).
3. Forum SBTVD. Fórum do Sistema Brasileiro de TV Digital Terrestre. http://www.forumsbtvd.org.br (in Portuguese).
4. Richardson, I.: H.264/AVC and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia. Chichester, John Wiley and Sons (2003).
5. Altera Corporation: Altera - The Programmable Solutions Company, http://www.altera.com
6. Agostini, L. et al: Multiplierless and Fully Pipelined JPEG Compression Soft IP Directed to FPGAs. In: Journal Microprocessor and Microsystems. pp. 487-497 (2007).
7. Agostini, L. et al: High Throughput Multitransform and Multiparallelism IP for H.264/AVC Video Compression Standard. IEEE International Symposium on Circuits and Systems – ISCAS'06 (2006).
8. Cheng, Z. et al: High Throughput 2-D Transform Architectures for H.264 Advanced Video Coders. IEEE Asia-Pacific Conf. on Circuits and Systems, pp. 1141-1144 (2004)
9. Chen, K. et al: An Efficient Direct 2-D Transform Coding IP Design for MPEG-4 AVC/H.264. IEEE International Symp. on Circuits and Systems – ISCAS'05, pp. 4517-4520 (2005).
10. Wang, T. et al: Parallel 4x4 2D Transform and Inverse Transform Architecture for MPEG-4 AVC/H.264. IEEE International Symposium on Circuits and Systems, pp. 800-803 (2003).
11. Lin, H. et al: Combined 2-D Transform and Quantization Architectures for H.264 Video Coders. IEEE International Symp. on Circuits and Systems – ISCAS'05, pp. 1802-1805 (2005).
12. Agostini, L. et al: High Throughput Architecture for H.264/AVC Forward Transforms Block, ACM Great Lake Symposium on VLSI – GLSVLSI'06 (2006).