FPGA IMPLEMENTATION OF DEBLOCKING FILTER CUSTOM INSTRUCTION HARDWARE ON NIOS-II BASED SOC

Bolla Leela Naresh[1] N.V.Narayana Rao [2] and Addanki Purna Ramesh[3]

Department of ECE, Sri Vasavi Engg College, Tadepalligudem, West Godavari (dt), Andhra Pradesh, India
leelanaresh.b@gmail.com
hyma_369@yahoo.com
purnarameshaddanki@gmail.com

ABSTRACT

This paper presents a frame work for hardware acceleration for post video processing system implemented on FPGA. The deblocking filter algorithms ported on SOC having Altera NIOS-II soft core processor.SOC designed with the help of SOPC builder .Custom instructions are chosen by identifying the most frequently used tasks in the algorithm and the instruction set of NIOS-II processor has been extended. Deblocking filter new instruction added to the processor that are implemented in hardware and interfaced to the NIOS-II processor. New instruction added to the processor to boost the performance of the deblocking filter algorithm. Use of custom instructions the implemented tasks have been accelerated by 5.88%. The benefit of the speed is obtained at the cost of very small hardware resources.

KEYWORDS

Deblocking filter, SOC, NIOS-II soft processor, FPGA

1. INTRODUCTION

Video broadcasting over the internet to handheld devices and mobile phones is becoming increasingly popular in the recent years. Also High Definition TVs are becoming common. But due to the limitation in the transmission bandwidth the videos will generally be encoded using video coding techniques which uses DCT and quantization that brings blockiness in the decoded video. In such scenario a smoothing filter to remove the blockiness is crucial which is called as deblocking filter. This deblocking filter algorithm requires high amount of processing requirement with the increase of the resolution of the images like High Definition Resolution. For example a HD image that has 6.23 million pixels in it needs 187 million pixels to be processed in one second to achieve widely accepted frame rate of 30 fps (frames per second). This means the video post processing system running at 300 MHz frequency (The high frequency systems will suffer from the high power consuming there by less battery life) gets just 1.6 cycles to process one pixel. Hardwired architectures which especially designed for particular algorithms can achieve this kind of processing requirements but they are lack of flexibility and can't be changed with the change in algorithms. Whereas processor based System on Chips gives enough flexibility to cope up with the change in algorithms but their programmable nature will not give the performance requirements like above. Therefore identifying application specific custom

DOI : 10.5121/vlsic.2011.2403

instruction blocks to accelerate the ported image post processing algorithm onto the processor based SOC will be a good choice. This approach is called as Fine Grain Acceleration. Fine grain acceleration of algorithms is a classic approach to accelerate the algorithms that requires high processing requirement. In fine grain acceleration technique the small parts of the algorithm which will be executed multiple times there by contributing good amount of final processing requirement as identified and these algorithm sections are accelerated by keeping dedicated hardware. This hardware will be activated by extended instructions of the processor. These instructions added to the instruction set (ISA) of the processor are called custom instructions. To come up the custom instructions required to achieve the frame rate mentioned above, we have chosen a framework with RISC processor and custom instruction addition capability. The Altera FPGA tools can be used to build such platform. Tools likes Altera SOPC (System on Programming Chip) builder, NIOS-II RISC processor and NIOS-II software development IDE eases to build the required platform to come up with the right custom instruction to speed up the deblocking filter. The SOPC builder technology provided simple way of adding the custom instructions to the NIOS-II soft processor and the NIOS-II IDE provides easy way of providing the software interface to the custom instruction added to the NIOS-II processor. In the next sections of the processor we give the details of the deblocking filter algorithm and the details of the developed system. Then we propose the custom instructions to accelerate the deblocking filter algorithm. And in the last section of the document we give the cycle count numbers and comparison with the latest hardware implementation of the deblocking filter are given.

2. DEBLOCKING FILTER ALGORITHM

There are two building blocks within many video coding standards like H.264, MPEG-4 or VC-1 which can be a source of blocking artifacts. The most significant one is the block-based integer/fractional discrete cosine transforms (DCTs) in intra and inter frame prediction error coding. Coarse quantization of the transform coefficients can cause visually disturbing discontinuities at the block boundaries. The second source of blocking artifacts is motion compensated prediction. Motions compensated blocks are generated by copying interpolated pixel data from different locations of possibly different reference frames. Since there is almost never a perfect fit for this data, discontinuities on the edges of the copied blocks of data typically arise. To reduce such blocking artifacts and improve the quality of the decoded video deblocking filter is necessary. The deblocking filter is a smoothing filter which will be applied on the edge of the block. Since we are targeting for a generic deblocking filter which should be used for videos encoded by any standard, we consider that the blocking artifacts can come at the minimum at 4X4 level and we apply the filtering at that level on whole video frame. Since the deblocking filter should not smoothens the real edge information present in the video frame, the filtering process considers the absolute differences between the pixels on the either side of the edge and if this difference crosses certain programmable thresholds called as *alpha* and *beta* then only the filtering is applied. The equations of the smoothing filter are taken from H.264 standard and are given below. The p0, p1, p2 and p3 and q0, q1, q2 and q3 given in the equations are pixel values on either side of an edge between two 1X4 blocks present in a 4x4 block as shown in the Figure 1.

Left 4x4 block				Right 4x4 block			
p3	p2	p1	p0	q0	q1	q2	q3
p3	p2	p1	p0	q0	q1	q2	q3
p3	p2	p1	p0	q0	q1	q2	q3
p3	p2	p1	p0	q0	q1	q2	q3

Figure 1: An edge between two 4X4 blocks

The equations of the absolute differences to determine the filter condition are:

$$ap = |p2 - p0|$$
$$aq = |q2 - q0|$$
$$app1 = |p1 - p0|$$
$$aqq1 - |q1 - q0|$$
$$apq0 = |p0 - p0|$$

The equations of the deblocking filter are:

$$if \left(ap < beta\ and\ apq0 < \left((alpha \gg 2) + 2\right)\right)$$

$$fp0 = (p2 + 2p1 + 2p0 + 2q0 + q1 + 4) >> 3$$
$$fp1 = (p2 \mid p1 \mid p0 \mid q0 \mid 2) >> 2$$
$$fp2 = (2p3 + 3p2 + p1 + p0 + q0 + 4) >> 3$$

$$else$$

$$fp0 = (2p1 + p0 + q1 + 2) >> 2$$
$$fp1 = p1$$
$$fp2 = p2$$

$$if \left(aq < beta\ and\ apq0 < \left((alpha \gg 2) + 2\right)\right)$$

$$fq0 = (q2 + 2q1 + 2q0 + 2p0 + p1 + 4) >> 3$$
$$fq1 = (q2 + q1 + q0 + p0 + 2) >> 2$$
$$fq2 = (2q3 + 3q2 + q1 + q0 + p0 + 4) >> 3$$

$$else$$

$$fq0 = (2q1 + q0 + p1 + 2) >> 2$$

$$fq1 = q1$$
$$fq2 = q2$$

The fp0, fp1, fp2 and fp3 and fq0, fq1, fq2 and fq3 given in the equations are pixel values after filtering. The above filter equations has to be applied on each row of the edge between current 4x4 block and the left 4x4 block and then on each column of the edge between the current 4X4 block with the top 4x4 block.

3. SYSTEM ARCHITECTURE

In this section of the paper, we are giving the details of the developed system to find out the best custom instructions to accelerate the deblocking filter process. The **Error! Reference source not found.** shows the system. The NIOS-II is the soft embedded processor that is used in the system which performs controlling of all the modules the system. The program to be executed by the NIOS-II processor resides inside SDRAM. The NIOS-II processor fetches the instructions using its instruction bus which is connected to the AVLON system bus. The SDRAM controller which is connected to the system bus fetches the data from SDRAM and presents on the system bus. The processor fetches the data using its data bus again connected to the system bus. The input images to be processed also lie in the SDRAM. The SRAM is used as frame buffer. The processor fetches the input video frames to be processed from SDRAM and applies deblocking filter using the deblocking filter custom instructions and writes the results to the frame buffer maintained in SRAM. From this frame buffer the final filtered pixels will be continuously read by the display controller which gives them to the video DAC for display. The display controller also generates the HSYNC and VSYNC signals to the Video DAC. The display process should not interrupted therefore the final images to be displayed are kept in SRAM which has low latency. The switches are used to RESET the signal.

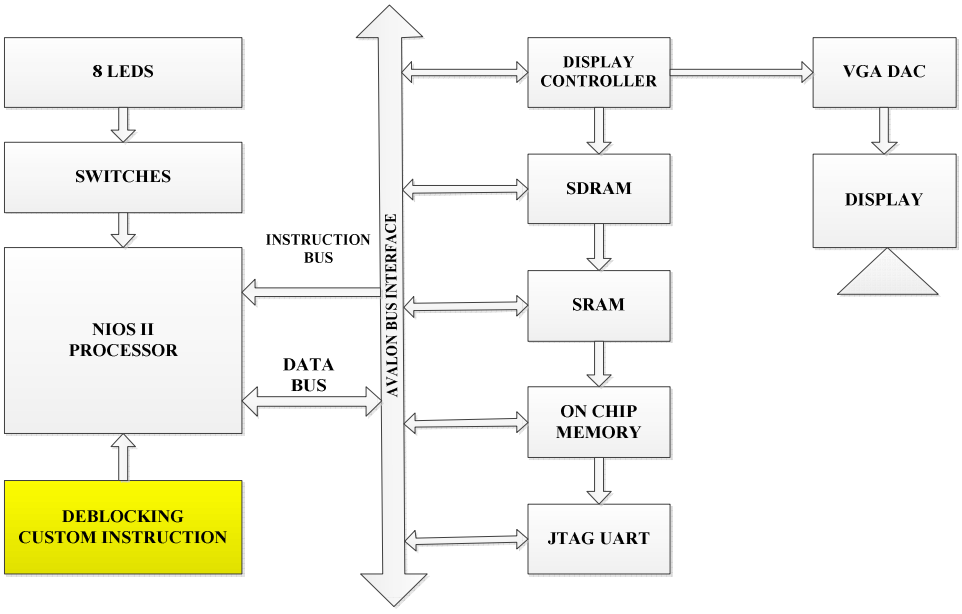

Figure 2: NIOS- II based SOC to derive the custom instructions for deblocking filter

4. DEBLOCKING FILTER CUSTOM INSTRUCTIONS

Custom instructions are custom logic blocks adjacent to the ALU in the processor data path and within the processor boundary, extending the CPU instruction set to accelerate time-critical software. By addition custom instructions to the processor we can reduce a complex sequence of standard instructions to a single instruction implemented in hardware [11]. For example a processor without multiplier in its ALU needs the multiplication routine to be written in assembly using several add and shift instructions. For this processor a multiplier can be added as a custom logic to speed up the process. The NIOS-II CPU configuration wizard provides a facility to add up to 256 custom instructions to the processor. The custom instruction logic connects directly to the NIOS II processor ALU logic as shown in **Error! Reference source not found.**.

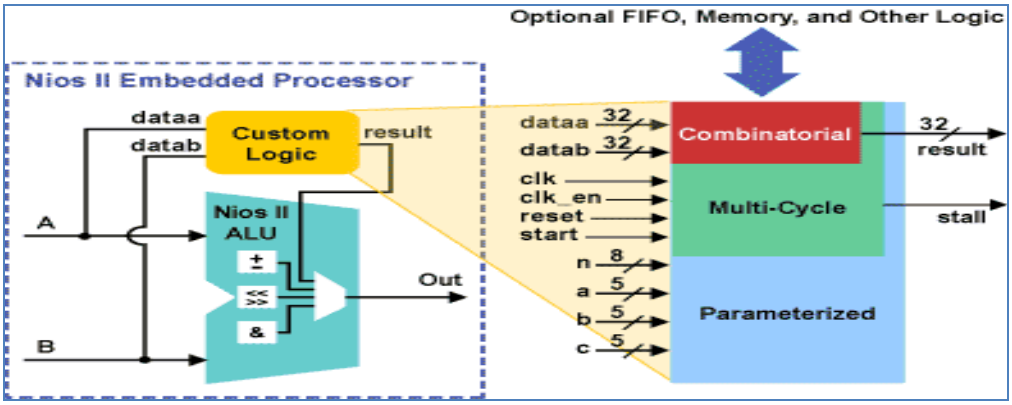

Figure 2: NIOS II Custom instruction logic

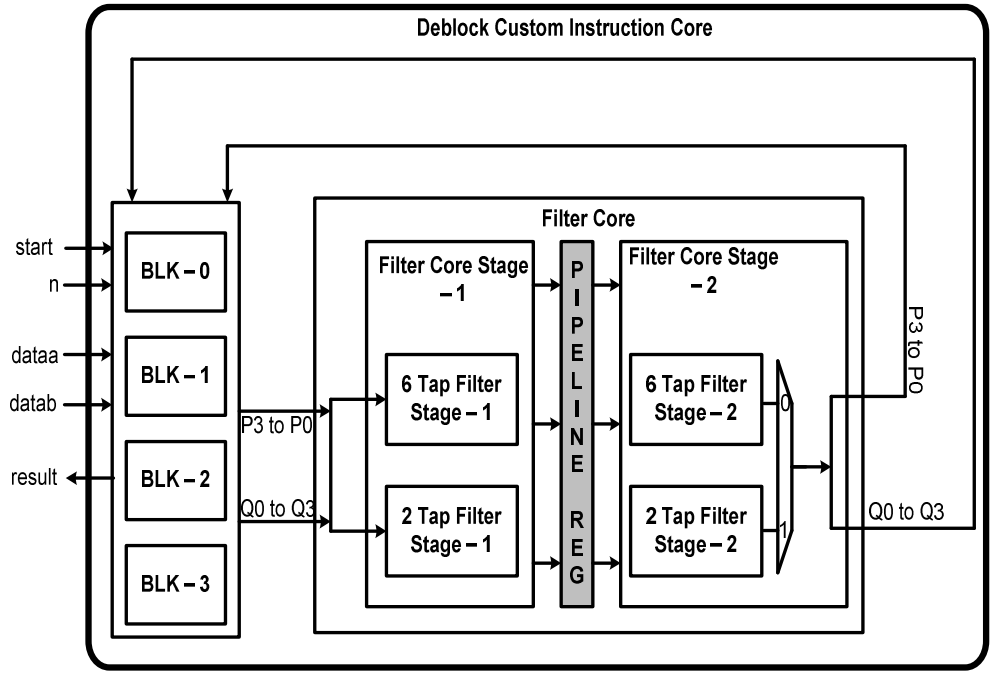

Figure 4: Block Diagram of deblocking filter custom instruction core

To speed up the deblocking filtering process implemented originally in software on NIOS II processor, custom hardware is designed and with a novel architecture and to activate this custom hardware 5 custom instructions are added to the ISA of the processor. The block diagram of the deblocking filter custom hardware is shown in **Error! Reference source not found.**. The description of I/Os of the custom block are given in Table 1.

Table 1: I/O Details of the custom block

Pin	Direction	Details
start	Input	Enables execution of a custom instruction
n	Input	The number of the custom instruction to be executed
dataa	Input	Data A operand of the instruction
datab	Input	Data B operand of the instruction
result	output	Result of the instruction

The custom instruction hardware has four blocks of pixel storage in it and each block can store one 4x4 block of pixels. These four blocks will be filled by the processor by sending 4 pixels at a time as instruction operand. The processor invokes the HW (hard ware) once to filter one row/column on the edge between two 4X4 blocks. The idea is whenever it calls the HW to do filtering it sends 4 pixels of the next block and it receives 4 filtered pixels back from the HW. Means to completed processing of 1 4x4 block it takes 4 instructions. The HW stores the right 4x4 block when its applying filtering on the edge between a two blocks located horizontally next to each other so that the filtered data need not fetched again when the edge between the top and that block has to be filtered. Similarly it stores the bottom 4x4 block in it so the filtered data need not fetched again when the edge between this block on the next horizontally located block has to be filtered. The custom instructions identified are given below.

FILL_BLK: This instruction fills one of the four 4X4 storage block.

SET_PARMS: This instruction sets the filtering parameters *alpha* and *beta*.

DBK_H: This instruction applies filtering on one row of the edge between two four 4X4 horizontally located blocks.

DBK_V: This instruction applies filtering on one column of the edge between two four 4X4 vertically located blocks.

NOP: This instruction is to introduce one wait cycle. Two wait cycles has to be inserted before starting a new edge which uses the filtered results of the previous edge. This is because of the 2 cycle latency of the HW.

With these custom instructions, the HW can process one 4X4 block in 6 cycles. (four instructions cycles + two wait cycles).

5. RESULTS

In this section we are presenting the simulation results of the deblocking filtering process of one edge between two 4x4 blocks. The simulation results showed in the
Figure **3** depicts that in 6 cycles the process is completed. The results are obtained using the Modelsim tool. Our deblocking filter process will be carried at 4x4 block level on the complete frame. For comparison we extend this results to one 16x16 block (called as Macro Block (MB)). A 16X16 block contains 32 4X4 level luma edges, and assuming 4:2:0 image coding format, 16 4x4 level chroma edges. The total edges in the 16X16 Macro Block are 48. Our algorithm takes 48x6 = 288 cycles to process. This is "18 clock cycles" less than the architecture proposed in [1]. With this processing speed we can achieve a processing rate of 1.125 pixels/clk. Table 2 shows the comparison of proposed work with reference paper [1]

Table 2: Comparison of proposed work with reference paper [1]

Sl. No	Reference paper(1)	proposed
Required clock cycles per Macro block	306	288

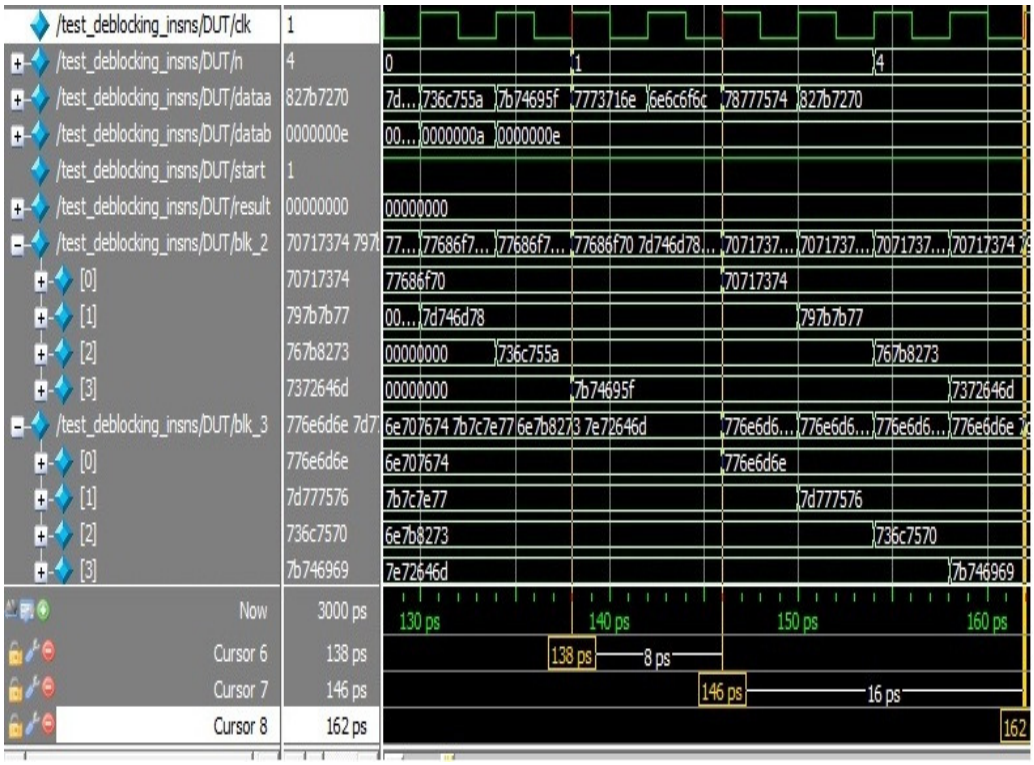

Figure 3: Simulation Results of filtering of 1 edge

The blocked image shown in Figure6 is taken as input after processing on NIOS-II processor which is implemented on CYCLONE II FPGA and the output image is shown in Figure7 is deblocked image.

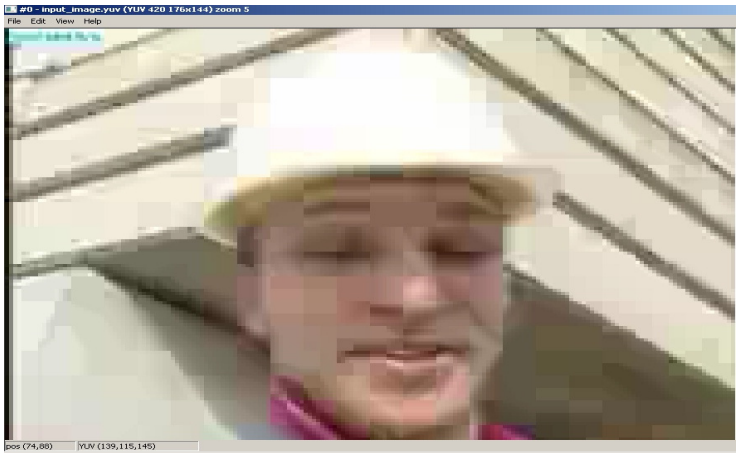

Figure 6: Blocked image as input

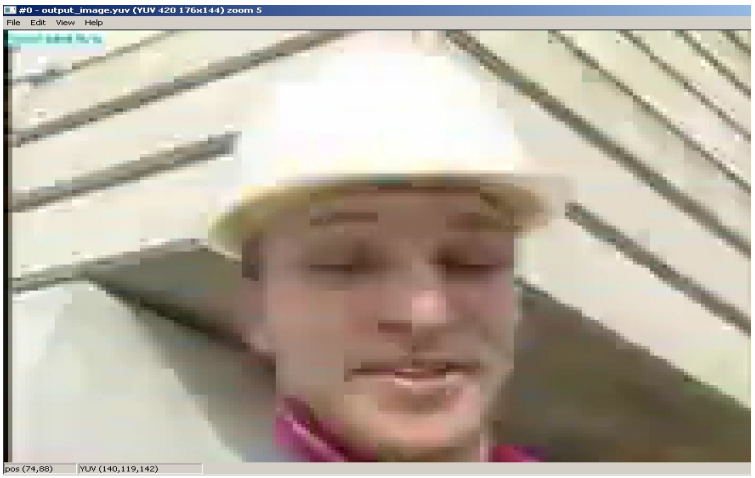

Figure 7: Deblocked image as output

The proposed architecture is implemented on CYCLONE-II FPGA (EP2C20F484C7) and the area numbers are given in Table 3.

Table 3: Area Results

Resource Type	Area Count
Total Logic Elements	4137
Logic Registers	3244

6. CONCLUSIONS

This research paper presents a framework for identifying custom instructions to accelerate a generic deblocking filter algorithm that is used for removing the blocking artifacts introduced by video encoders. A novel architecture for the deblocking filter is proposed which takes 18 clock cycles less then the architecture proposed in reference paper [1]. RTL code has been written to the deblocking filter architecture in Verilog and simulated using Modelsim. The hardware designed is connected to the NIOS-II processor custom instruction bus and the configured onto CYCLONE-II FPGA.

REFERENCES

[1] Wen Jia, Leibo Liu, Shouyi Yin, Min Zhu, Zhihua Wang "A Fast Complete Deblocking Filter on a Coarse- Grained Reconfigurable Processor Supporting H.264 High Profile Decoding" by NNSF of China, 2010, pp-221-224

[2] K.S.Chaitanya, P.Muralidhar, C.B.Rama Rao "Implementation of Reconfigurable Adaptive Filtering Algorithms" National Institute of Technology Warangal, India 2009 International Conference on Signal Processing Systems.

[3] Jung-Ah Choi and Yo-Sung Ho "Deblocking Filter Algorithm with Low Complexity for H.264 Video Coding", Gwangju Institute of Science and Technology (GIST) ,2008 pp. 138–147.

[4] C. Arbelo1, A. Kanstein, S. Lopez, J.F. Lopez, M. Berekovic, R. Sarmiento1 and J.-Y. Mignolet "Mapping Control-Intensive Video Kernels onto a Coarse-Grain Reconfigurable Architecture: the H.264/AVC Deblocking Filter" Research Institute for Applied Microelectronics Spain. (IUMA).2007 EDAA

[5] Chung-Ming Chen Chung-Ho Chen "An efficient architecture deblocking filter in H.264/AVC video coding" international conference on computer graphics and designing.2005.

[6] TU-T Recommendation H.264 "Advanced video coding for generic audiovisual services" was approved on 1 March 2005 by ITU-T Study Group 16 (2005-2008)

[7] Iain E G Richardson "H.264 / MPEG-4 Part 10 White Paper Reconstruction Filter" 30/04/03

[8] Iain E. G. Richardson "H.264 and MPEG-4 Video Compression Video Coding for Next-generation Multimedia" The Robert Gordon University, Aberdeen, UK

[9] Iain E. G. Richardson www.vcodex.com

[10] http://www.altera.com/devices/processor/nios2/benefits/performance/ni2-acceleration.html

[11] ALTERA NIOS–II Processor Custom instruction user guide